UNIVERZITA PARDUBICE

Dizertační práce

2019

Josef Brožek

UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Metodika budování distribuovaných simulačních modelů odrážejících provoz vybraných systémů s decentralizovaným řízením

Josef Brožek

Dizertační práce

2019

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 1. 6. 2019 Josef Brožek

ANOTACE

Dizertační práce se zabývá návrhem a ověřením metodiky využitelné pro budování počítačových distribuovaných simulačních modelů. Praktickým výsledkem práce je původní metodika nazvaná Metodis, která je navržena pro nasazení nezávisle od implementačních platforem. Metodika byl ověřena při zpracování relevantních případových studií.

KLÍČOVÁ SLOVA

Metodika, simulace, distribuovaná simulace, HLA, DIS.

TITLE

Methodology for building distributed simulation models reflecting the operation of selected systems with decentralized management

ANNOTATION

The dissertation deals with the design and verification of the methodology usable for building computer-based distributed simulation models. The practical result of this work is a original methodology called Metodis, which is designed to be deployed independently of implementation platforms. The methodology was verified during the processing of relevant case studies.

KEYWORDS

Methodology, computer simulation, distributed simulation, HLA, DIS.

Obsah

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK, ZNAČEK A TERMINOLOGIE

Termín	Význam
AAR	After Action Review. Systém pro zpětnou rekapitulaci průběhu simulačního experimentu. Využíván především v simulátorech typu trenažér.
Archimate	Standard pro dokumentaci softwaru
ASD	Adaptive Software Development. Adaptivní vývoj softwaru je agilní metodikou pro vývoj softwaru, vytvořen v roce 2000. Zjednodušuje vývoj softwaru zpětným učením a úpravou procesu vývoje.
Avatar	Reprezentace (nejčastěji vizuální) uživatele ve virtuálním prostředí.
BPM	Business Process Model
CORBA	Common Object Request Broker Architecture. Standard pro vzdálené volání metod. To znamená, že umožňuje volání metod fyzicky dislokované třídy.
Dekompozice logických procesů	Pojem zahrnuje dekompozici na logické procesy, jakož i dekompozici jednotlivých logických procesů.
DIS	Distributed Interactive Simulation. Distribuovaná interaktivní simulace dána standardem IEEE 1278. V minulosti široce využívaný standard pro tvorbu interaktivních simulátorů. Dnes je překonán.
Federace	Distribuovaná simulace složená z (potenciálně) heterogenních federátů.
Federát	Simulátor (potenciálně) kooperující v rámci distribuované simulace. Názvosloví převzato ze standardu HLA (IEEE1516:2010), ale pojem je užíván v obecném kontextu i pro simulátory nezaložené na výše uvedeném standardu.
FOM	Federation Object Model. Definice objektů v XML, které jsou dostupné v simulaci dle HLA.
GUI	Graphic User Interface. Grafické uživatelské rozhraní, zpravidla na počítači, nebo v jiném vizualizačním prostředku informačních technologií.
HIL	Hardware in the Loop. Systém provozu softwaru (vč. testů), kdy je plná kontrola nad daným universem svěřena autonomně hardwaru. To znamená, že systém je bez možnosti uživatelského zásahu.
HLA	High Level Architecture. Architektura pro návrh a provoz simulátorů daná standardem IEEE1516.
ICT	Information and Communication Technologies. Informační a komunikační technologie.
IIOP	Internet Inter-ORB Protocol. Protokol umožňující komunikovat aplikacím implementovaným v různých programovacích jazycích prostřednictvím internetu.

Interakce	Působení (nejčastěji uživatelské) na software či hardware s cílem získání odezvy, nebo změny stavu programu či dat.
Interaktivní simulátor	Simulátor, který umožňuje provádění interakcí během vlastního simulačního výpočtu. Umožňuje tak změny stavového prostoru, které (pokud nejsou korektně zaznamenány) zamezují opakovatelnost simulačního experimentu.
JSON	JavaScript Object Notation. Způsob zápisu dat (datový formát) nezávislý na počítačové platformě.
JVM	Java Virtual Machine. Virtuální počítač Java, který umožňuje spouštět univerzální Java programy nezávisle na počítačové platformě (JVM je sám o sobě platformově závislý, a tak touto formou závislost překlenuje).
Konstruktivní simulace	Počítačová simulace realizovaná na logicko-matematických modelech deterministického nebo stochastického charakteru využívaná k neindividuální (bezavatarové) interaktivní simulaci.
Logger	Specifická třída/funkcionalita programovacího jazyka, která umožňuje logování na programové úrovni.
Logický proces	Základní, logicky ucelená činnost či posloupnost činností, které jsou popsatelné sekvenčním způsobem pro potřeby tvorby simulace. Pojem vychází z disciplín ekonomických věd, kde logické procesy představují atomické součásti byznys procesů.
Logování	Proces, který dokumentuje rozvoj programu, jeho průběh a případně jeho chyby. Umožňuje reprodukovatelnost běhu programu.
LP	Logical process. Vizte Logický proces.
LVT	Local Virtual Time. Lokální virtuální čas. Časový údaj (často vč. datumového údaje), který je platný pro konkrétní virtuální výpočet/výpočetní uzel v daný, konkrétní okamžik.
Metodis	Vlastní původní metodika, která je výstupem této dizertační práce.
Middleware	Specifické programy, které poskytují dalším programům služby nad rámec služeb poskytovaných operačním systémem. Například JVM, RTI.
MIL	Man in The Loop. Paradigma využívané v simulaci a počítačových testech, které zdůrazňuje, že součástí vlastního prostředí je také člověk, resp. interakce od uživatele.
MVC	Model View Controller. Třívrstvý návrhový model, který je využitelný především pro standardní počítačovou platformu. Postupně zastarává ve prospěch vícevrstvých modelů.
OMT	Object Model Template. Definice dat v XML, které jsou nutnou běhovou podmínkou pro HLA simulace.
OOP	Object Oriented Programing. Objektově orientované programování založené na přidružení funkcí k datům a řadě dalších pravidel jako je dědičnost, zapouzdřednost, polymorfismus.

OOSP	Object Oriented Software Process. Agilní metodika pro návrh softwaru, která je silně objektově orientovaná.
Oponentský avatar	Avatar, který je ztělesněním neuživatelských vtělení. Často reprezentuje jednotlivé instance umělé inteligence. Cílem oponentských avatarů je zpravidla soupeření o zdroje s avatary.
ORB	Object Request Broker. Specifická rutina pro vzdálené volání metod/přístupu k instancím a jejím metodám.
PDU	Protocol Data Unit. Specifický souborový formát využívány v simulaci navržené a provozované v souladu s DIS. Vlastní PDU se dále skládá z hlavičky (header) a těla (body), přičemž PDU hlavička určuje zařazení do příslušné PDU kategorie (family).
Prostředí	Vlastní vizuální obsah interaktivního simulátoru, který není vymezen stavovým prostorem simulátoru a není klíčový pro fungování jednotlivých procesů. Slouží k dotváření uživatelského vizuálního vjemu. Obdobný pojem v oblasti herního průmyslu je Level Design.
Překážka	Část prostředí, která vytváří překážku pro pohyb entit, nebo určitých jen typů entit, nebo která filtruje určité fyzikální vlastnosti prostředí.
QoS	Quality of Services. Síťový protokol, který umožňuje zajištění prioritního řízení datového toku pro vybrané služby.
R&D	Research and Developement. Vizte heslo VaV.
R&D&I	Research and Developement and Inovations. Vizte heslo VaVaI.
RMI	Remote Method Invocation. Způsob IIOP specifický pro programovací jazyk Java.
RTI	Run-Time Infrastructure. Middleware umožňující provoz HLA simulací.
RUP	Rational Unified Process. Interativní vývoj softwaru založený na UML.
SCRUM	Jedna z nejpoužívanějších agilních metodik pro návrh softwaru.
Seat	Z anglické terminologie, přičemž pro pojem neexistuje vhodný překlad. Jde o právě jednu sadu uživatelských vstupně výstupních zařízení spřažených pro plnění konkrétní vstupně výstupní funkce. Například pro trenažér – simulátor stíhacího stroje je seat ovládací rozhraní pro právě jednoho pilota.
SOM	Simulation Object Model. Definice objektů v XML, které jsou dostupné v simulaci dle HLA.
Stream	Datový proud.
Task	Úloha, úkol. Může být atomický, nebo složený.
UX	User eXperience. Postup návrhu GUI založený na rekurzivním vytváření UI a získávání uživatelské zpětné vazby na toto UI.
Uživatelský avatar	Reprezentace (nejčastěji vizuální) uživatele ve virtuálním prostředí.

VaV	Výzkum a vývoj. Dle legislativy je to takový proces, který obsahuje 1) ocenitelný prvek novosti, 2) kterým dochází k vyjasňování výzkumné, nebo technické nejistoty a 3) je systematickou tvůrčí prací. Anglická zkratka pro totéž je R&D.
VaVaI	Věda a výzkum a inovace je rozšířením hesla VaV o přidání prvku Inovací. Inovace je proces, při kterém dochází k obnově zastaralých a zavádění nových: produktů, technických, sociálních, či kulturních schémat a procesů.
Wrapper	Specifická softwarová vrstva/knihovna, která umožňuje používat rutiny jiného softwaru/programovacího jazyka, jež by bez wrapperu nebyly použitelné/dosažitelné.
XML	Extensible Markup Language. Značkovací programovací jazyk pro uchování konzistentních dat nezávisle na počítačové platformě.

ÚVOD

1 Úvod

Dizertační práce nese název Metodika budování distribuovaných simulačních modelů odrážejících provoz vybraných systémů s decentralizovaným řízením. Tento název velmi věrně odráží obsah samotné práce i její zaměření.

Při pohledu shora je třeba konstatovat, že hlavním cílem práce bylo vytvoření metodiky. Metodika byla definována jako jeden z páteřních výstupů VaV činnosti v České republice. Standardně jsou metodiky budovány rozsáhlými tvůrčími týmy. Právě proto, že již zadání počítalo s tím, že metodiku vytvoří jednotlivec, šlo o velmi ambiciózní zadání dizertační práce. Metodika samotná je výstup komplexní, neboť musí být vždy zasazena do kontextu současného technického poznání. Metodika tak musí brát v potaz existenci dalších předpisů, standardů a postupů tak, aby je vhodně rozšiřovala, nebo specifikovala pro aplikaci v konkrétní aplikační doméně. Z výše uvedeného plyne, že v práci byl kladen důraz především na velmi obšírnou rešerši, která musela zmapovat všechny používané techniky, normy a doporučení, které jsou v oblasti používány. Důraz musel být kladen také na takové postupy, které nejsou dány žádnou normou, ale jde pouze o běžnou praxi v oblasti aplikovaného vývoje.

Po provedení obsáhlé rešerší činnosti bylo možné přistoupit k budování vlastní metodiky. Metodika musela být vybudována v souladu s požadavky na Certifikovanou metodiku (tak jak ji chápe dokument Definice druhů výsledků výzkumu, experimentálního vývoje a inovací, vydaný Úřadem vlády České republiky, sekcí Místopředsedy vlády pro vědu, výzkum a inovace, platné od 1. 1. 2014). Právě dokončená metodika (i ve fázi, před získáním certifikace) je hlavním výstupem dizertační práce, metodika byla nazvána *Metodis*.

Metodika samotná vznikla v roce 2014 a od té doby byla, v rámci dizertační práce, ověřována. Jednotlivé případové studie, které vznikly souběžně s metodikou, nebo práce, nebo simulátory, které vznikly výhradně jako prostředek pro ověření metodiky, jsou představeny jako součást této dizertační práce.

1.1 Struktura práce

Dizertační práce byla rozdělena na oddíly, které tematicky sdružují klíčové nejdůležitější části práce. Nehledě na strukturu oddílu byla práce strukturována do kapitol, které sdružují obsah na základě společné tématiky. Kapitoly jsou číslovány vzestupně arabskými čísly. Kapitoly jsou zařazeny do oddílů tak, aby byla práce přehlednější. Práce obsahuje následující oddíly:

1. Úvod
2. Rešerše problematiky
3. Metodika a postup jejího návrhu
4. Řešení ověřující metodiku
5. Závěr

Detailně pak struktura práce reflektuje výše zmíněné oddíly a to takto:

1. Úvodní část se zaměřuje na uvedení práce, zasazení do kontextu, hlubší analýzu názvu práce a předpokládaných výstupů (kapitola 1). Součástí úvodní části práce je vymezení oboru zájmu, o kterém pojednává kapitola 2, která (mimo jiné) specifikuje vlastní zadání výzkumného úkolu. Kapitola Výzkumný záměr se také vyjadřuje k použitým metodám a postupům (kapitola 3). Poslední kapitolou první oddílu (kapitola 4) je Vymezení základních pojmů. Tato kapitola je pro další práci klíčová, neboť v odborné komunitě je řada pojmů vymezena s určitým stupněm volnosti (nebo jasná definice konkrétního pojmu neexistuje, případně dochází k určitým sporům o přesném vymezení pojmů mezi jednotlivými vědeckými školami). Zároveň je třeba zdůraznit, že existují původní pojmy v anglickém jazyce, pro které neexistuje jednoznačný český překlad a jejich popis opisem není zcela triviální. Třetí kapitola tak vytváří odborný slovník a vymezuje konkrétní způsob práce s konkrétními termíny v rámci dizertace.
2. Oddíl nazvaný Rešerše problematiky sdružuje kapitoly, které se zabývají popisem rozsáhlého bádání o současném stavu problematiky. Již vytvoření přehledu technik a paradigmat využitelných a využívaných v oblasti počítačové simulace je významným tvůrčím výsledkem. Rešeršní činnost se zabývá logickým procesem a metodami jeho dekompozice (v kapitole 5), jako první, logickou úrovní, strukturování simulátoru během jeho tvorby. V další kapitole (kapitola 6) se rešerše zaměřuje na použité techniky a technologie, přičemž předmětem zkoumání jsou především HLA a DIS. Právě HLA a DIS se stávají klíčovými technologiemi, pro které je možné využít celou metodiku. Kapitola sedmá představuje rešerši v oblasti implementačních paradigmat využitelných na architektonické úrovni, resp. při vlastní programové implementaci simulátorů a to především s důrazem na taková řešení, která budou využívat super-skalárního nebo distribuovaného zpracování jednotlivých částí simulačního výpočtu. V kapitole osmé je kladen důraz na techniky využitelné při tvorbě takového simulátoru, který umožní uživatelské změny stavového prostoru simulátoru dynamicky během průběhu simulačního výpočtu (fakticky popisuje možné implementace interaktivních zásahů u simulátoru). Kapitoly devět a deset vytváří ucelený přehled v současných normách, doporučeních a metodikách, které jsou využitelné pro budování simulátorů, resp. obecněji, pro vývoj softwarů. Poslední kapitola oddílu (kapitola 10) vytváří přehled o vlastním nasazení simulátorů, používaných technikách a technologiích v době současné, nebo nedávné minulosti.

Velká část oddílu je kopií Písemné zprávy ke státní doktorské zkoušce autora (která dále vychází z vlastních publikací (Brožek a Jakeš, 2014) a (Brožek a Jakeš, 2017), ze jejichž české verze bylo čerpáno také v Jakešově (2014) diplomové práci.

3. Oddíl Metodika a postup jejího návrhu se zaměřuje na klíčovou část dizertační práce: metodiku samou. Oddíl postupně vysvětluje klíčové pojmy a postupy při tvorbě metodiky. První kapitola oddílu (kapitola 11) se věnuje pojmu metodika v kontextu výstupu VaV činnosti, specifikuje a formalizuje metodiku, kategorizuje typy metodik a vysvětluje proces schvalování metodik jako VaV výstupů. Kapitola se také věnuje problematice kontextu metodiky v rámci komplexního systému vědeckého poznání a úzce navazuje na rešerši technologií, když uvádí, proč je třeba metodiky důsledně podřizovat již zavedeným pojmům, postupům a paradigmatům. Kapitola dvanáctá vysvětluje, jak řešitelské týmy vytváří metodiky, přičemž z této kapitoly přímo vychází část, která obecné principy ukazuje na konkrétní metodice – totiž metodice *Metodis* předkládané jako tvůrčí výsledek této dizertační práce.
4. Klíčový oddíl nazvaný Řešení ověřující metodiku představuje vybrané etapové výsledky, které byly autorem samostatně, nebo ve spolupráci s týmem, předkládány, a které slouží pro demonstraci vývoje či aplikace *Metodis*. To znamená, že jednotlivé kapitoly slouží jako představení etapových výsledků, které byly publikovány a které slouží jako demonstrace faktu, že metodika je prakticky použitelná. Celý oddíl tak prokazuje to, že samostatná vědecká práce předkladatele dizertační práce, je korektní a aplikovatelná.
5. Poslední oddíl je nazván jako Závěr a obsahuje povinné údaje dizertační práce, jakými je přehled vlastní publikace, včetně komentáře, přehled pedagogických a dalších aktivit a nedílně také závěry vlastní práce.

1.2 Typografické konvence

Práce využívá typografických konvencí stanovených ve směrnici Univerzity Pardubice č. 9/2012 Pravidla pro zveřejňování závěrečných prací a jejich základní jednotnou formální úpravu a ze souvisejících standardů ČSN ISO 2145 Dokumentace – Číslování oddílů a pododdílů psaných dokumentů a ČSN ISO 01 6910 Úprava písemností psaných strojem nebo zpracovaných textovými editory.

Konkrétní odchylky a zpřesnění je třeba uvést pro číslování kapitol, kdy je úvodní kapitola číslována číslem 1, nikoli 0 a to z důvodu vnitřní struktury kapitoly. V práci je používána práce se zdroji v souladu s Harvardským citačním stylem. Důležitá fakta a klíčové součásti seznamů jsou značeny tučným písmem. Odkazy a terminologie, která je specifická pro tuto práci (tedy není běžně používána v odborné komunitě) je v textu značena kurzívou.

2 Základní údaje o práci

Tato kapitola vymezuje samotný obor zájmu, vyjadřuje se k motivaci volby tématu a přibližuje zaměření práce.

2.1 Vymezení oboru zájmu

Analýzou názvu dizertační práce Metodika budování distribuovaných simulačních modelů odrážejících provoz vybraných systémů s decentralizovaným řízením bylo možné vyslovit několik prvotních předpokladů, které vytvořily ohraničení samotné práce.

2.1.1 Metodika

Z názvu práce je jasně patrné, že klíčovým prvkem byla Metodika. Nutné tedy bylo prostudovat co vlastně metodika je a jak ji chápe česká legislativa. Bylo nutné provést kategorizaci metodik, zjistit, jakými procesy dochází k tvorbě metodik a jaké prostředky jsou pro tuto tvorbu používány.

Z praktického hlediska byl klíčový výběr třídy systémů, pro které bude metodika uplatnitelná a tvorba prototypů, které metodiku ověří. Bylo nutné také zohlednit dvě aplikační pravidla:

- Pokud je metodika tvořena obecně, bez znalosti příslušné aplikační domény, vždy bude existovat taková množina úloh, pro jejichž řešení není metodika vhodná.
- Pokud je metodika tvořena na míru konkrétním aplikačním doménám, tak bude její zobecnění nevhodné, nebo dokonce zcela nemožné.

Přestože je tvorba prototypu standardně krokem, který následuje až po tvorbě metodiky, neboť slouží k jejímu ověření, je žádoucí uvažovat o aplikační doméně (a prototypu) již při tvorbě metodiky. Tím dochází k paradoxu, neboť při návrhu dochází k řešení rekurzivního problému: Metodika by měla být dostatečně obecná, avšak vhodná pro danou aplikační doménu, čímž dochází k její restrikci, a proto je třeba ji zobecnit.

Obecně jde charakterizovat tvorbu metodiky tak, že se autor musí vždy snažit najít vhodný kompromis mezi dvěma pravidly, která byla zmíněna výše.

2.1.2 Budování systému

Z názvu dizertační práce plyne, že zaměření bude především na budování simulátorů. V rámci restrikce témat je tak možné částečně upozadit problematiku ověřování, provozu a vyhodnocování simulační studií. Tato restrikce není v rozporu s tím, co simulace je, nebo k čemu se užívá. Zaměření práce se problematice věnuje pouze v omezeném rámci – právě zmíněném budování simulátorů.

2.1.3 Distribuovaný simulační systém

Část názvu odrážející provoz distribuovaných systémů, lze interpretovat tak, že samotné modelované systémy mají distribuovaný charakter. Přestože neexistuje nutná ekvivalence v charakteru modelovaných a modelujících systémů, bylo pro potřeby této práce rozhodnuto, že i modelující systémy budou mít distribuovaných charakter. Tím, že řešení bude distribuované, dojde ke zjednodušení řady procesů.

V průběhu práce bylo nutné klasifikovat třídy úloh, pro které bylo vhodné využití simulací (na rozdíl od ostatních metod, například prostřednictvím heuristických a exaktních metod operačního výzkumu). Nad touto množinou úloh bylo nutné následně provést restrikci a vybrat ty z nich, které je výhodné řešit prostřednictvím distribuované simulace. Při zpracování textu bylo čerpáno a více informací lze najít v publikacích (Banks, 2010), (Pelánek, 2011).

2.1.4 Decentralizované řízení

Decentralizované řízení se jevilo jako jasně definovaný technický pojem. Avšak pokud byla problematika vnímána v širších souvislostech, bylo nutné si uvědomit, čeho se samotná decentralizace týkala, neboť bylo nutné důsledně rozlišovat jednotlivé typy decentralizace. Základní třídy decentralizace lze rozlišovat především na:

- úrovni modelovaného systému,
- logické úrovni modelujícího systému a
- fyzické úrovni modelujícího systému.

Tato klasifikace vychází z Fujimota (2000) a Craiga (1996) s přihlédnutím k možnosti aplikace v HLA (IEEE 1516, 2010).

2.1.5 Vymezení systémů

V rámci práce bylo třeba provést volbu typů systémů, pro které bylo vhodné aplikovat jednotlivá řešení. Řešení byla prezentována prostřednictvím prototypů, které měly dostatečný rozsah pro to, aby prokázaly výhodnost využití právě těch metodik návrhů, které byly vytvořeny v rámci této práce.

2.2 Motivace

Motivací pro vypracování této práce byl zejména zvyšující se potenciál simulačních výzkumů (a to především s využitím distribuovaných simulátorů), kdy rozvoj v hardwaru usnadňuje realizaci efektivních simulátorů. V budoucí praxi se tak dá předpokládat, že realizace simulačních studií bude pro vybranou problematiku stále častější.

2.3 Rešeršní činnosti

Rešeršní činnost byla prováděna standardními metodami (vyhledání v databázích, knihovnách, sbornících aj.), avšak také prostřednictvím pohovorů s kapacitami v oboru – díky nim bylo možné získat doporučení na konkrétní monografie, odborné články, nebo jen odkaz na zajímavé praktické aplikace. Součástí rešeršní činnosti bylo i několik exkurzí na renomovaných pracovištích (např. simulační centrum na letišti Pardubice, simulační centrum na Univerzitě Obrany, oddělení vývoje agentově orientovaných simulátorů na Žilinské Univerzitě), získávání informací z veletrhů, nebo předváděcích akcí v rámci konferencí (např. ukázka trenažéru řízení lodi a ukázka provozu trenažéru kontejnerového jeřábu na konferenci EMSS2015), nebo zahraniční stáž zaměřená na HLA (Lancaster University Management School).

Vzhledem k tomu, že bylo původní rešeršní šetření obsáhlé, neboť se snažilo pojmout všechna základní témata pro tvorbu metodiky a zároveň zmapovat metody, současné trendy a architektury, bylo seznámení s jeho výsledky rozloženo do několika samostatných kapitol.

3 Výzkumný záměr a použité metody

Dizertační práce se zaměřila na problematiku metodiky jako výstupu VaVaI, její tvorby, ověření a určení její uplatnitelnosti. Vlastní výzkumný záměr se tak zaměřil primárně na vybudování metodiky, vymezení klíčových pojmů a postupů v kontextu řešení práce. Seřazené procesy pak lze identifikovat tak, že bylo třeba:

1. Vymezit obor řešení, technologie řešení, aplikační doménu,
2. vytvořit vlastní metodiku,
3. ověřit vlastní metodiku,
4. realizovat podpůrné práce a činnosti,
5. sumarizovat řešení v dizertační práci.

3.1 Restrikce řešení dle oboru a technologie

Prvním nutným krokem bylo definovat si obor zájmu. Samotný obor simulací je extrémně rozsáhlý a vyžívá mnoho zcela rozdílných přístupů.

Klíčové bylo vytvořit si kvalitní znalostní základnu, díky níž bylo možné jasně specifikovat, jaké jsou trendy v současných simulačních vědách, jaké jsou užívány principy, paradigmata a architektury k budování simulátorů.

Na základě již zjištěných skutečností bylo žádoucí zohlednit především následující odvětví:

- systémy pro podporu rozhodování,
- aplikace mobilních technologií v simulaci,
- heterogenní systémy s různým typem dekompozice,
- netradiční řešení simulačního jádra na úrovni sekvenčního simulátoru (federátu),
- vstupně výstupní zařízení jako součást simulátorů,
- principy pro emulaci prostředí.

A následně tato simulační odvětví použít ve vhodné aplikační doméně, a to především v systémech odrážejících provoz v:

- pozemní dopravě,
- průmyslové výrobě, nebo
- obslužných a logistických systémech.

3.2 Záměr vlastních metodik pro návrhy

Proto, aby bylo možné vytvořit vlastní metodiku, bylo třeba vymezit řadu kritérií, a to především:

- použitou architekturu simulátorů,
- použitý systém dekompozice,
- použité řešení na softwarové úrovni,
- vymezení aplikačních domén,
- problematiku typů interakcí,
- problematiku použitelných programovacích jazyků,
- problematiku interaktivity.

Stanovením toho, která z výše uvedených kritérií se stala konstantami (tj. například prohlášením: „Metodika je použitelná výhradně pro simulátory odrážející provoz na železnici.“) a která kritéria se stala proměnnými metodiky, došlo k jasnému vymezení metodiky.

Ke stanovení jednotlivých postupů pak došlo na základě vlastní výzkumné činnosti.

3.3 Záměr tvorby simulačního modelu

Vlastní metodika je, z pohledu technického, jasně stanovena co se týče rozsahu i povinných náležitostí. Z pohledu VaVaI také existují jasně stanovené podmínky toho, co musí metodika splňovat. Avšak splněním formálních náležitostí není možné o metodice prohlásit, že je korektní. Proto v rámci dizertační práce vznikl, kromě záměru vytvořit vlastní metodiku, také záměr vytvořit jeden či více simulátorů, které metodiku ověří.

Záměrem tedy bylo vytvořit nejméně jeden simulační model, který bude splňovat následující výčet podmínek:

- na řešení simulátoru se bude podílet více než jeden řešitel,
- simulátor bude netriviálního rozsahu,
- půjde o distribuovaný simulátor,
- řešitelský tým bude postupovat v souladu s *Metodis*,
- řešitelský tým bude sestaven tak, aby bylo vysoce pravděpodobné, že bez použití *Metodis* by tento tým nebyl schopen dosáhnout stejného, nebo kvalitativně obdobného cíle,
- výsledky budou publikovány ve sbornících mezinárodních konferencí a ve vědeckém časopise.

3.4 Ostatní práce

Samotným navržením simulačního modelu samozřejmě práce nekončila. Bylo nutné provést jeho validaci a verifikaci. Tato činnost byla o to složitější, že se musely uvažovat nejen chyby v konceptuálním nebo fyzickém modelu, ale zároveň bylo nutné uvažovat chyby již v samotné metodice pro návrh simulátoru. Komplikace s validací, případně verifikací, bylo

nutné umocnit přítomností živé obsluhy, neboť simulátor třídy trenažér, který umožní většinu řízení přenechat na člověku, bylo velmi náročné podrobovat testům na stabilitu řešení.

3.5 Použité metody

Dizertační práce byla postavena na standardně popsaném iterativním cyklu VaV činnosti: pozorování, formulace problémů, logická indukce/logická dedukce, ověření. Přičemž, vzhledem k tomu, že řešený obor zájmu byl definován aplikačně, nedocházelo ke standardní formulaci hypotéz a jejich ověřování. Místo formulace hypotézy byla součástí práce formulace metodiky, která byla následně ověřována její aplikací.

V rámci dizertační práce byly použity především (ne však výhradně) následující vědecké metody:

- Empirická rešerše – pro sběr dat, vymezení problematiky a hranic problematiky,
- Abstrakce – pro vyhledání nosných problémů, které jednotlivé technologie/jednotliví řešitelé řeší odlišně, přestože klíčová část problému je totožná,
- Logická konkretizace – v kombinaci s rešerší pro vymezení technologií a technik,
- Analýza – proces faktického rozčlenění problému na dílčí podproblémy,
- Syntéza – pro kombinaci technologií, technik a paradigmat při řešení širších aplikačních a funkčních celků řešených v rámci metodiky (tj. řešení dílčích problému získaných analýzou a jejich zpětná syntéza pro vyřešení původního problému),
- Induktivní a deduktivní postupy použité při tvorbě metodiky, stejně jako využité při tvorbě jednotlivých simulátorů ověřujících metodiku.

4 Vymezení základních pojmů

Protože obor počítačové simulace a obecně informačních technologií je relativně široký, existuje řada odborných pojmů, které se překrývají (např. atribut jako součást objektu, atribut jako obecná vlastnost a atribut jako pojem z HLA simulace). Kapitola by měla vyjasnit a ucelit základní pojmy, které jsou v rámci práce použity.

Věc, systém a abstrakce

„V simulaci a modelování se studuje nějaká ***věc****, resp. možné varianty nějaké věci, přičemž slovo věc chápeme tak, jak jej chápou filozofové: je to nějaký objekt hmotného světa, a to buď objekt, který vskutku existuje (např. organismus konkrétní osoby, konkrétní továrna, krajina, škola atd.), nebo o kterém uvažujeme, že by existovat mohl (např. stroj, budova či výrobní provoz, který by měl být realizován, nebo nemocný organismus dané osoby, o jehož terapii se uvažuje, ale definitivní rozhodnutí dosud nebylo formulováno). Věc chápou filosofové v její úplné složitosti (pokud existuje) nebo spolu se všemi nejasnostmi její existence (pokud se uvažuje o možnosti věc realizovat) a chápou i to, že není v lidských silách celou věc racionálně, tj. rozumovými prostředky, pochopit a zvládnout. Tak to chápou i různé obory vědy, techniky a řízení společnosti, a proto zavádějí na zkoumaných věcech* ***abstrakce****, které zanedbávají některé aspekty těchto věcí; zanedbané aspekty jsou vybrány tak, že aspekty, které zbývají, jsou daným vědeckým, technickým či společenským oborem zvládnutelné: mimo jiné, mohou o nich racionálně komunikovat pracovníci odpovídající vědecké, technické či společenské profese. Takovou abstrakci budeme v modelování a simulaci nazývat* ***systémem*** *a podle charakteru profese, která systém na věci „vidí“, „zavádí“ či „definuje“, dostává systém i přívlastek: např. televizní přijímač je obvykle chápán jako elektronický systém, neboť i jeho běžný majitel ví, že si ho kupuje pro jeho elektronické vlastnosti.“* (Křivý a Kindler, 2001)

Model

„Slovo model se používalo v běžné řeči nejprve pro předlohu. V odborném jazyku doby před simulací a virtuální realitou zůstal z této praxe termín funkční model, a to pro první exemplář navrženého výrobku, který pracuje tak, jak by výrobek pracovat měl, přestože jiné vlastnosti výrobku (např. estetické) tento exemplář ještě nemá. Z této praxe vznikla i interpretace slova model pro něco zvláštního, nezvyklého či nákladného (např. hlavně před druhou světovou válkou používané termíny model klobouku, model automobilu apod.).“ (Křivý a Kindler, 2001)

V modelování a simulaci je termín model použit pro analogii mezi dvěma systémy (modelovaným a modelujícím). Pro určení konkrétního typu této vazby, případně diskusi o stupních volnosti, nebo úrovni abstrakce lze čerpat více informací z Křivého a Kindlera (2001), Kindlera (1980), nebo Bankse (2010). Příkladem vazby mezi modelovaným a modelujícím systémem je například mapa (modelující systém) a svět (modelovaný

systém), tak jak uvádí Kindler a Křivý (2001): *„Jednoduché příklady nabízí mapa (model části země na papíře) (…).“*

Statický a dynamický systém

„Abstrakce může nebo nemusí zanedbat význam času. Např. význam času v systémech železniční dopravy nelze běžně zanedbat, avšak konstruktér mapy železniční sítě České republiky k jízdnímu řádu roku 1997 zanedbává jak to, že se po jednotlivých tratích pohybují v čase vlaky, tak to, že se železniční síť může měnit před rokem 1997 i po něm. Systém, v němž se od významu času abstrahuje, se nazývá ***statickým systémem*** *(anglicky static system). Pokud se od významu času neabstrahuje, pak jen výjimečně se berou v úvahu i jeho vlastnosti, jak je poznává moderní kvantová fyzika. V drtivé většině oborů se čas chápe ”newtonovsky”, to jest jako v klasické fyzice, čili tak, že je smysluplné mluvit o tom, že dvě události nastaly v systému současně nebo jedna z nich nastala dříve než druhá. Systém, jehož čas se nezanedbává a je přitom chápán takto newtonovsky, se v modelování a simulaci nazývá* ***dynamickým systémem*** *(angl. dynamic system). Simulace se jinými než dynamickými systémy nezabývá.“* (Křivý a Kindler, 2001)

Modelování

„Podstatou modelování ve smyslu výzkumné techniky je náhrada zkoumaného systému jeho modelem (přesněji: systémem, který jej modeluje), jejímž cílem je získat pomocí pokusů s modelem informaci o původním zkoumaném systému.“ (Křivý a Kindler, 2001)

Simulace

Z různých zdrojů je možné získat širší škálu definic, proto jsou uvedeny jen ty definice, které se vzájemně doplňují, případně rozšiřují.

„Úkolem simulačního programu je zjistit, jak se bude simulovaný systém chovat pro zadaná vstupní data. Úkolem simulačního programu není provádět optimalizaci, tzn. hledat, pro která vstupní data dostaneme optimální řešení. Uživatel může provádět se simulačním programem opakované simulační experimenty s cílem zjistit očekávané výsledky pro různá vstupní data a nalézt tak optimální řešení problému.“ Překlad z Bankse (1998).

„Simulace je v matematice a kybernetice vědecká metoda, při které se zkoumají vlastnosti nějakého systému pomocí experimentů s jeho matematickým modelem. Počítačové simulace se staly užitečnou součástí matematického modelování mnoha přirozených systémů ve fyzice, chemii a biologii, psychologii a získaly nový pohled do fungování těchto systémů.“ (Kindler, 1980)

„Simulace je výzkumná technika, jejíž podstatou je náhrada zkoumaného dyn. systému jeho simulátorem s tím, že se se simulátorem experimentuje s cílem získat informace o původním zkoumaném dynamickém systému.“ (Křivý a Kindler, 2001)

Verifikace

V souvislosti s počítačovou simulací, je verifikace modelu proces potvrzení, že je simulace správně zavedena s ohledem na konceptuální model (tedy že odpovídá specifikaci a předpokladům pro daný účel použití). Během verifikace jsou testovány a opravovány implementační chyby v modelu. Účelem verifikace modelu je zajistit, že provádění modelu je koncepčně správné.

Existuje mnoho technik, které mohou být použity pro ověření modelu. Použitelnými metodami jsou například: kontrola modelu odborníkem, zkoumání výstupu modelu na přiměřenost pod různými nastaveními vstupních parametrů, nebo pomocí interaktivního debuggeru. Také platí, že mnoho inženýrských technik používaných pro verifikaci softwaru je použitelných pro verifikaci simulačních modelů. Zdoje: (Kindler, 1980), (Banks, 2010), (Fujimoto, 2000), (Pelánek, 2011), (Ulrych a kol., 2007).

Validace

Validační experimenty se zaměřují na to, jakým způsobem model zobrazuje reálný systém (pokud existuje). Nejde tak již o koncepční problémy ze strany zadání, ale koncepční problémy ze strany původních požadavků. Validací modelu se dle definice rozumí *„Ověření, že počítačový model v rámci své oblasti použitelnosti dosahuje uspokojivého rozsahu přesnosti v souladu s určeným použitím modelu.“* Překlad z Bankse (2010).

Již v roce 1967 bylo definováno, že validace by se měla seskládat ze tří hlavních kroků, kterými jsou:

1. Vybudování věrného modelu (autoři ve své definici přibližují pojmu verifikovaného modelu).
2. Ověření předpokladů modelu (vstupních generátorů a jejich nastavení).
3. Porovnání výsledné transformace vstupů a výstupů pro model s totožnou transformací u skutečného systému (pokud existuje).

Pro výše uvedené bylo čerpáno zejména ze zdrojů: (Fujimoto, 2000), (Quinn a kol., 2006), (Missile Defense Agency, 2008), (IEEE1012, 2012).

Paralelní a distribuovaná simulace

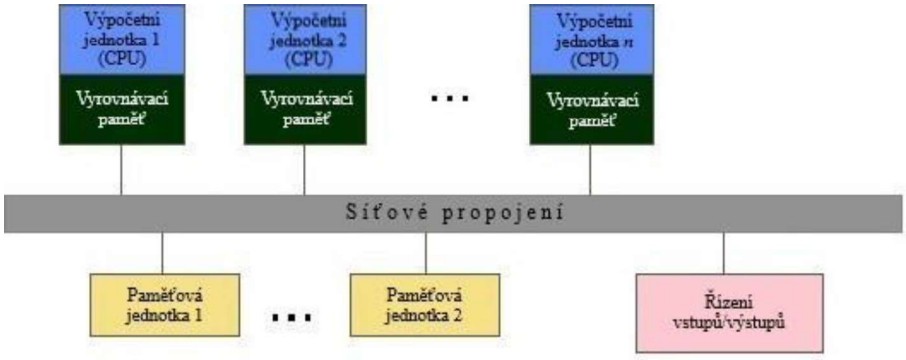
Tato kapitola je přebrána z Ulrycha a kol. (2007), což je překlad vybraných pasáží z Fujimota (2000).

Počítačová simulace představuje výpočet, který modeluje chování reálného či abstraktního systému. Paralelní a distribuovaná simulace umožňuje spuštění tohoto simulačního programu na více počítačových procesorech, které jsou vzájemně síťově propojeny. Rozdíl mezi těmito dvěma přístupy je založen na formě výpočetního systému, na němž simulace běží. Základní charakteristiky obou technologií jsou uvedeny v následující tabulce:

Tabulka 1: Paralelní a distribuovaná simulace[1]

	Paralelní simulace	Distribuovaná simulace
Fyzická poloha	Počítačová laboratoř	Budova, město, svět
Procesory	Homogenní	Většinou heterogenní
Komunikační síť	LAN	LAN/WAN
Komunikační latence	Několik (max. desítky) mikrosekund	Stovky mikrosekund, sekunda

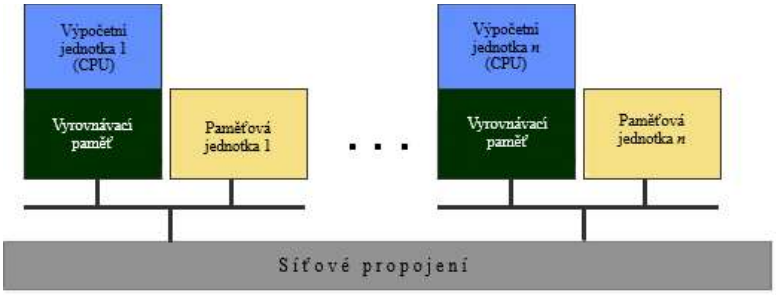
Principem paralelní simulace je realizace výpočtu a běhu simulačních programů na bázi multiprocesorových výpočetních platforem. Simulace probíhá na skupině počítačů jednoho výpočetního centra. Schematicky je podstata paralelní simulace znázorněna na následujícím obrázku 1.



Obrázek 1: Schématické uspořádání komponent paralelní simulace[2]

Oproti tomu podstatu distribuované simulace představuje výpočet prováděný na geograficky vzdálených, různě rozmístěných počítačích (výpočetní střediska mohou být rozmístěna po celém světě). Ty jsou vzájemně propojeny prostřednictvím lokální či světové sítě, znázorněno na obrázku 2.

[1] Zdroj: (Ulrych a kol., 2007), originální obrázek (Fujimoto, 2000)
[2] Zdroj: (Ulrych a kol., 2007), originální obrázek (Fujimoto, 2000)



Obrázek 2: Schématické uspořádání komponent distribuované simulace[3]

V obou případech je výpočet jednotlivých simulačních modelů (třeba tvořených z několika simulačních programů či podprogramů) distribuován na několik procesorů. Existuje mnoho důvodů proč využít distribuce simulačního výpočtu a jeho spuštění na více počítačových systémech, a to například:

- Zkrácení výpočetních časů – rozčleněním rozsáhlých simulačních modelů do množství dílčích výpočtů, které jsou řešeny souběžně, je možné zredukovat čas výpočtu úměrně počtu využitých procesorů.
- Geografické rozmístění – spuštění simulačních programů na množině zeměpisně zcela odlišně umístěných počítačů, umožňuje vytvářet jeden virtuální svět za přispění mnoha účastníků (ti mohou být také fyzicky rozmístěni po celém světě). Tento způsob uspořádání výpočetního systému snižuje cestovní náklady, časové prodlevy, usnadňuje komunikaci a v konečném důsledku značně zvyšuje efektivitu.
- Integrace modelů, jež jsou počítány na strojích různých výrobců – lze uvést jednoduchý příklad praktického využití: letecké simulátory pro různé typy letadel (či pro jednotlivá letecká odvětví) byly vyvíjeny zcela odděleně různými výrobci. Mnohem jednodušší a nákladově efektivnější způsob než složitý porting těchto programů na jeden počítač (násilné svázání jednotlivých simulátorů dohromady), je tvorba nového virtuálního prostředí, ve kterém je každý výpočet prováděn na jiném počítači (ty jsou však virtuálně propojeny) s možností sjednotit komunikační architekturu.
- Tolerance k chybám – dalším významným přínosem multiprocesorových výpočetních technik je rostoucí odolnost vůči chybám. Pokud totiž v průběhu simulace jeden procesor tak zvaně vypadne, ostatní mohou dále pokračovat ve výpočtu, aniž by byl přerušen simulační běh. Výpadek jednoho stroje tedy neznamená výpadek celé simulace. Je však nutné posoudit, zda tento výpadek nezpůsobí zkreslení výpočtu. Celá problematika tolerance k chybám však rozhodně není triviální, v této práci lze nalézt pouze nástin dané problematiky.

[3] Zdroj: (Ulrych a kol., 2007), originální obrázek (Fujimoto, 2000)

Logický proces

Pojem logický proces je v komunitě matematiků i informatiků zabývajících se simulací velmi často používán. Velkým problémem však je formální zavedení tohoto pojmu, neboť i světová literatura (Banks, 1998), (Fujimoto, 2000) se často exaktní definici elegantně vyhýbá. Obecné pojetí, které říká, že význam pojmu logický proces může být z odlišných zdrojů odlišný, není pro tuto práci vyhovující.

Jedním z nejstarších pohledů na logický proces je parafrázovaná definice, která říká, že logický proces je takový proces, který je možné řešit pomocí sekvenčního automatu. Podle takové definice by však mnoho simulačních modelů vytvořených v některém ze simulačních nástrojů (např. Rockwell Arena) už ze sekvenčního principu simulačního nástroje, bylo logickým procesem, a to by platilo i pro modely velkého rozsahu. Logický proces by však rozhodně neměl být sekvenční simulátor velkého rozsahu.

Další fakt, který vylučuje tuto definici logického procesu, vychází od Fujimota (2000), který se zabývá rozsahem logického procesu v paralelních simulacích. Pokud by definice logického procesu byla taková, jak je uvedeno výše, jednotlivé paralelní simulace by odpovídaly definici logického procesu. Prakticky by tak nebylo třeba provádět další dekompozici na logické procesy, nebo dekompozici logických procesů.

Pro potřeby této práce je proto pojem logický proces zaveden následujícím způsobem: Logický proces je označení pro proces změny stavového prostoru nebo entity, anebo pro soustavu těchto procesů, pokud mezi nimi existuje příčinná souvislost, kterou je možné logicky snadno odvodit.

Pravidlo logičnosti je z pohledu simulačního architekta (resp. analytika) klíčové, neboť právě logicky dovozené celky příčin a následků je možné modelovat.

Emulace

Emulátor je v informatice druh softwaru umožňující běh počítačových programů na jiné platformě (architektuře, operačním systému), než pro kterou byly původně vytvořeny a kterou samy od sebe podporují. Typickým příkladem emulátoru je program umožňující běh videoher známých z herních konzolí na běžném PC pod Microsoft Windows či Linuxem. Emulace je speciálním případem virtualizace (někdy je však chápána jako silnější pojem). Text vychází z (Wikipedia: Emulátor, 2001) a ověřen je z pramenů (Muira, 2007) a (van der Hoeven a kol., 2007).

V informačně-teoretickém smyslu lze podle Church-Turingovy teze libovolné výpočetní prostředí emulovat na libovolném jiném. Tato teoretická vlastnost však v praxi naráží na dvě hlavní překážky: 1) Příslušná emulace může být pro praktické účely příliš pomalá, avšak 2) nejčastější překážkou je špatná dostupnost informací o původní architektuře (platí pro proprietární software). Při tvorbě emulátoru je pak potřeba používat metody reverzního inženýrství (Muira, 2007) a (van der Hoeven a kol., 2007).

Virtuální realita

„Virtuální realita (VR) (nebo virtuální prostředí) je technologie umožňující uživateli interagovat se simulovaným prostředím. Technologie virtuální reality vytvářejí iluzi skutečného světa (např. při výcviku boje, pilotování, lékařství), nebo fiktivního světa počítačových her.“ Přeloženo z Rheingolda (2000).

Jde o vytváření vizuálního zážitku zobrazovaného na obrazovce počítače, nebo speciální audiovizuální helmy, popř. oblečení snímající pohyb a stimulující hmat (Rheingold, 2000).

Z jiného zdroje: *„Odpověď na otázku "Co je virtuální realita": z technického hlediska je to přímočarý termín používaný k popisu trojrozměrného, počítačem generovaného, umělého prostředí, které může být zkoumáno ve styku s člověkem. Takováto osoba se stává součástí tohoto virtuálního světa, nebo se ponoří do tohoto prostředí, ve kterém je schopná manipulovat s objekty, nebo provést řadu akcí.“* Přeloženo z Rheingolda (2000).

Důrazně je třeba rozlišovat mezi pojmy Virtuální realita (což je pojem, kterým se tato práce zabývá) a pojmem Rozšířená realita, který je odlišným prostředkem informačních technologií ve skutečné realitě. V zahraniční literatuře se občas stává, že se pojmy chybně, překrývají, nebo nahrazují. To je způsobeno především tím, že pojmy mají komplikovaný kontext svého vzniku, vzájemného vývoje a odlišení.

REŠERŠE PROBLEMATIKY

5 Logický proces a jeho dekompozice

Součástí různých technologií, které byly analyzovány v rámci rešeršní činnosti, je odlišný přístup k logickým procesům (LP) a jejich potenciální dekompozici. V obecné rovině je možné prohlásit, že logické procesy je možné dekomponovat podle různých paradigmat. Vzhledem k tomu, že cílem práce bylo navržení vhodné metodiky pro budování distribuovaných simulátorů, bylo nutné zohlednit přinejmenším nejčastěji využívaná paradigmata pro dekompozici logických procesů.

Jednotlivé dekompoziční metody se zpravidla liší v závislosti na využívané implementační, nebo běhové technologii. Je žádoucí uvést výčet základních metodik či přístupů, které jsou při budování distribuovaných simulátorů využívány. Jsou to tedy především:

- tvorba simulátoru bez dekompozice logických procesů,
- dekompozice na agenty,
- funkční dekompozice logických procesů,
- specializační dekompozice logických procesů,
- dekompozice na paralelní procesy,
- dekompozice na mikroprocesy,
- tvorba kompozitního modelu.

Sestavení kategorizace je výsledkem vlastní rešeršní činnosti, jednotlivé techniky byly získány z publikací (Balmell a kol., 2006), (Logan a Theodoropoulos, 2005), (Mastaglio a Callahan, 1995), (Franklin a Graesser, 1997). Pro úplnost je vhodné doplnit, že logický proces je definován v řadě vědeckých disciplín – LP je charakterizován v disciplínách ekonomických vědních oborů (Smith, 2000), přičemž přesnější vymezení LP v simulaci a vztah LP a sekvenčního simulátoru provedl Kuhl a kol. (2000).

5.1 Tvorba simulátoru bez dekompozice logických procesů

Základní metoda pro tvorbu distribuovaného simulátoru je tvorba rozkladem, tedy separace různých dále nedělitelných logických procesů, nebo i širších, logicky pospojovaných celků a jejich rozmístění do různých výpočetních uzlů.

Tuto metodiku si lze velmi snadno představit na příkladu simulace provozu na segmentu dálnice se dvěma odpočívadly. V případě, že bychom model rozdělili na prostou dálnici, odpočívadlo 1, odpočívadlo 2 a každý takto vzniklý simulátor provozovali na jiném výpočetním uzlu, získáme tak distribuovaný simulátor vytvořený prostým rozkladem (bez dekompozice logických procesů).

Tato metoda má velkou výhodu v tom, že je snadno odvoditelná, transparentní a většina simulátorů využívá stejné entity. Verifikace a validace těchto systémů také patří k těm jednodušším.

Dekompozice rozkladem (bez dekompozice LP) je však použitelná i v situaci, kdy nedochází k užívání stejných entit. Například pro potřeby trenažéru bojového vozidla je možné provést dekompozici tak, jak je nastíněno na následujícím obrázku 3. Dojde tak k separaci části simulátoru pro řidiče a části simulátoru pro zbraňového specialistu. Tvorba rozkladem se tak stává nejjednodušším analytickým způsobem, jak vytvořit distribuovaný simulátor. Tato metoda je však použitelná jen v některých případech.

Obrázek 3: Dekompozice bojového vozidla rozkladem[4]

Další řešení prostřednictvím tohoto způsobu dekompozice je možné nalézt například v (Brožek a kol., 2017c), (Brožek a kol., 2014b), (Bruzzone a kol., 2010).

5.2 Dekompozice logických procesů na agenty

Jedna z nejčastěji používaných metodik pro dekompozici logických procesů je dekompozice na agenty. Tento způsob řešení je obecně známý, avšak existuje širší spektrum metod, jimiž lze dekomponovat logický proces na agenty. Řešení se kategorizují dle hierarchie, úrovně komunikace atp.

Přestože samotná disciplína agentových (resp. multiagentových) simulátorů je velmi rozsáhlá, tak samotné principy, které je nutné uplatnit pro tento typ simulace (pokud zanedbáme agenty) patří mezi principy jednodušší. Prakticky tak můžeme abstrahovat od složitosti agenta a prohlásit, že zajištění komunikace mezi jednotlivými agenty je nejvýše právě tak složité, jako zajištění komunikace v případě jiného typu dekompozice. Právě jasně definovaný problém, snadné komunikace a závislost klíčových (pod)problémů pouze na vnitřní struktuře konkrétního agenta je klíčovou výhodou agentových (resp.

[4] Zdroj: (Brožek, 2017c)

multiagentových) simulací, a proto je lze využívat i pro nejkomplexnější modelované systémy (jako modelování socio-ekonomický systémů, nebo fyzikálních systémů).

Protože problematika je velice obsáhlá, je možné pro více informací nahlédnout například do Kavičky, Klimy a Adamka (2005).

5.3 Funkční dekompozice logických procesů

Funkční dekompozicí logických procesů rozumíme takový princip, který rozloží simulující systém vertikálně, to jest dle jeho funkčních vlastností.

Tedy pokud máme několik železničních stanic propojených tratěmi, standardním rozkladem bychom došli k několika stanicím, které jsou co do vnitřní logiky typově stejné (například v nich probíhají stejné technologické procesy), fakticky tak dosáhneme delimitace některých funkcí. U složitějších modelů je však konzistence modelu analyticky řešeného rozkladem velmi obtížně udržovatelná.

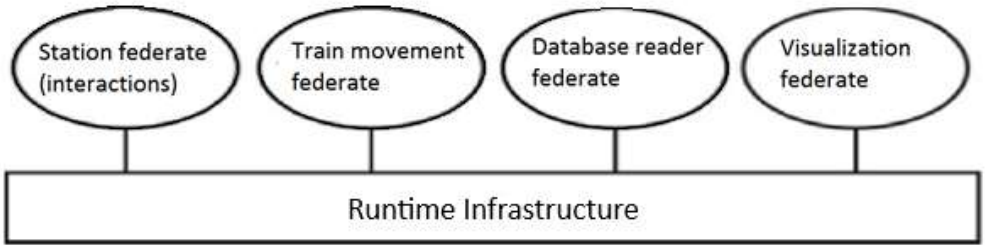
Funkční dekompozice může takovýto model rozdělit pragmatičtěji na správce kolejí, správce vlaků, správce stanic a například vizualizátor. Výhodou tohoto způsobu dekompozice je mnohem snazší údržba, neboť nedochází k duplicitnímu řešení problémů v rámci jednotlivých simulátorů. Nevýhodou je složitější obsažení dekompozice běžnou logikou a tedy nižší intuitivnost řešení.

Samotná funkční dekompozice se často používá právě pro ladění a testování simulátorů, jejich modifikací atp.

5.4 Kombinace rozkladu a funkční dekompozice logických procesů

Za určitých okolností je výhodné použít kombinaci více metod. Kombinace metody dekompozice rozkladem a následná aplikace funkční dekompozice logických procesů patří k často používaným variantám. Tento model je navíc následně převeditelný i na jiné způsoby dekompozice (např. další dekompozicí lze dosáhnout i dekompozice na agenty). Právě kombinace rozkladu a funkční dekompozice se používá při konstrukci většiny moderních simulátorů.

Vezmeme-li předešlý příklad železniční stanice, tak jeho prakticky navrženou a ověřenou dekompozici můžeme vidět na obrázku 4. Správa pro pohyb vlaků, databázový zapisovač (a verifikátor) a vizualizace jsou odděleny metodou funkční dekompozice.



Obrázek 4: Rozklad a funkční dekompozice[5]

5.5 Dekompozice na paralelní procesy

Dekompozice na paralelní procesy je způsob dekompozice, jež využívá principů dekompozice rozkladem k tomu, aby bylo možné jednotlivé části výpočtu provádět paralelně.

Výhodou tohoto způsobu dekompozice je takový simulační výpočet, jehož některé části jsou extrémně složité. Zjistíme-li, že v našem simulačním výpočtu existuje nějaké úzké hrdlo (z pohledu výpočetního výkonu), můžeme ho odstranit právě dekompozicí na paralelní procesy.

Důsledek rozkladu je takový, že proces, který byl úzkým hrdlem, paralelizujeme a umožníme tak provádění několika jeho částečných výpočtů současně, znázorněno na obrázku 5a.

Samotný princip je na první pohled jednoduchý a je nutné podotknout, že jeho nasazení je velmi efektivní. Na druhou stranu, aby byl aplikovatelný, musí proces a entity dosahovat nejvyššího stupně volnosti (tedy výsledek procesu nesmí mít vliv na entity, které do procesu chtějí vstoupit) a musí existovat způsob, jak definovat kauzální vazby u paralelně zpracovávaných entit.

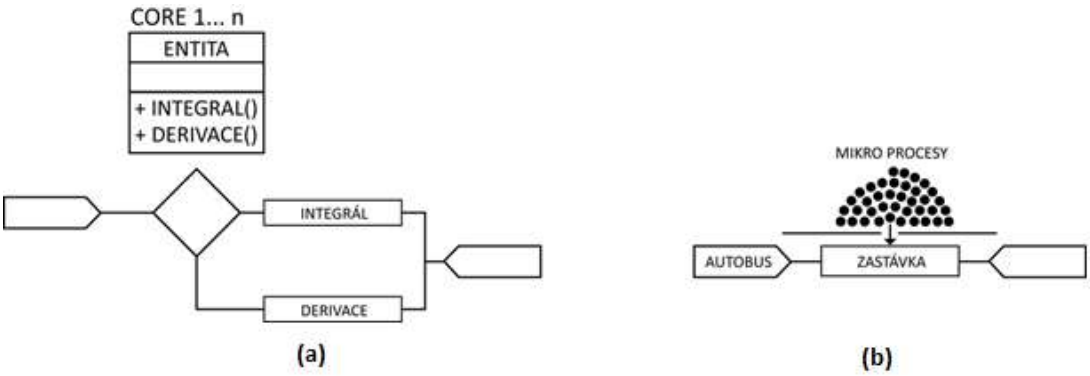
5.6 Dekompozice na mikroprocesy

Dekompozice na mikroprocesy je použitelná tam, kde existuje jeden proces, který je výpočetně netriviální a má zpracovávat velké množství entit.

Samotná dekompozice pak spočívá v obrácení celého systému tak, že funkčními celky se místo procesů stanou entity, přičemž zároveň jde o záměnu tzv. hloupé entity (enity, které jsou zpracovávány procesy) za tzv. inteligentní entity (entity, které mají vlastní logiku), tak jak je ve svých pracích chápe například Kindler a Křivý (2001).

[5] Zdroj: (Brožek a kol., 2014)

Nejjednodušší bude zřejmě vysvětlení na příkladu, tedy: Jeden proces, který by měl řídit pohyb desítek entit na zastávce autobusů, by byl velmi komplexní a pro zvyšující se množství entit také výpočetně velmi náročný. Na druhou stranu, pokud se stane funkčním celkem každá jednotlivá entita, tak jedinou odpovědností takovéto entity bude interakce s nejbližšími entitami prostřednictvím vlastního mikroprocesu. Proces čekání na zastávce se tak zjednoduší a hlavně jeho rozvoj (výpočet) lze velice efektivně provádět paralelně. Problematiku je možné vidět na obrázku 5b.



Obrázek 5: Dekompozice na paralelní procesy a mikroprocesy[6]

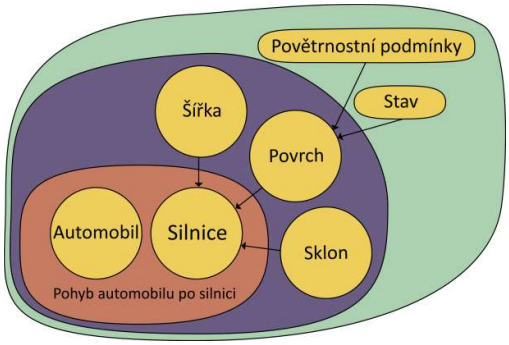
5.7 Tvorba kompozitního simulátoru

Kompozitní simulátory jsou postaveny tak, aby byly postupně rozšiřitelné. Je tak možné velice snadno škálovat jednotlivé výstupní parametry na základě výsledků předchozích experimentů.

Jednoduchý příklad je na obrázku 6: Základní tezí analytické struktury simulátoru jednoduchého kompozitního modelu pohybu automobilu po silnici je rekurzivní modularita. Stěžejní jsou nejprve pouze dva objekty (silnice a automobil), přičemž výpočet je rekurzivně prováděn do hloubky a pokud rozšíření nelze najít, je použit implicitní výpočet daný objektem. Systém funguje na principu: Pokud nemáš žádné rozšiřující komponenty, použij výpočet, který je implementován přímo v tobě. Čímž dochází k tomu, že základní model může kalkulovat například s jednoduchým vzorcem pro výpočet dráhy: $s = v \cdot t$. Při vysoké míře abstrakce je takovýto model korektní. Postupným rozšiřováním je však možné jednotlivé parametry zlepšovat. Na obrázku 6 je červeně vyznačen základní model, modře snížení abstrakce první úrovně, zeleně snížení abstrakce druhé úrovně. Tato snížení však byla realizována vždy jen nad jedním objektem. Základní parametry silnice je možné rozšířit o sklon, typ povrchu a šířku silnice. Výpočet pak nemusí proběhnout v objektu silnice, ale může být realizován prostřednictvím tří samostatných sub výpočtů. Při simulačním výpočtu

[6] Zdroj: Vlastní

pak bude bráno v potaz mnohem více parametrů a celý model může dosáhnout nižší úrovně abstrakce.



Obrázek 6: Kompozitní simulační systém[7]

Kompozitní architektury simulátoru fyzicky využívající především velice komplexní simulační řešení, pro které je žádoucí vlastnost, která garantuje, že zvyšující se komplexnost modelu snižuje úroveň abstrakce a to bez podstatného vlivu na architekturu simulátoru. Celý systém postupného zpřesňování je možné zachytit také hierarchickými diagramy, například k-cestnými stromy. Struktura příkladu by však reflektovala strukturu v obrázku 6. Při zpracování kapitoly bylo čerpáno a více informací lze najít v (Friedman, 2002) či (Tamegaya, 1997).

[7] Zdroj: Vlastní

6 Použité technologie a architektury

Vytvoření přehledu používaných technologií a dalších prostředků bylo klíčové proto, aby bylo možné vysvětlit další souvislosti práce. Účelem kapitoly je především:

1. Nastínit názvosloví a popsat principy a paradigmata používaná v HLA jako v architektuře, která byla hlavním aspirantem na implementační technologii dizertační práce.
2. Zdůraznit problematiku logické decentralizace kombinované s fyzickou centralizací jako jevu, ke kterému dochází v HLA.
3. Popsat další technologie a architektury využitelné pro implementaci distribuovaných simulátorů.

6.1 Architektura HLA

V této kapitole bylo čerpáno především ze standardu IEEE1516:2010, několika monografií (Knigth, 2001), (Simulation Interoperability Standards Organization, 2001), (Rabelo a kol., 2013) a pak také z praktického tutoriálu od společnosti Pitch Technologies A. B. (Pitch Technologies AB, 1993), která je autorem běhového prostředí (konkrétní implementace běhového rozhraní RTI) a jež se na školícím pracovišti využívá pro provoz HLA.

6.1.1 Úvod do HLA

High Level Architecture (dále jen HLA) je velmi rozsáhlá architektura pro vytváření a provoz distribuovaných simulací. Norma neomezuje aplikační doménu, způsob dekompozice, použité programovací jazyky, provozní hardware atp. Na nižších úrovních HLA specifikuje způsob komunikace mezi jednotlivými uzly distribuované simulace, přičemž požadavky klade především na samotný přenosový formát (jde o komunikaci prostřednictvím XML). Díky tomu je možné vytvářet softwarově i hardwarově heterogenní simulační systémy (např. část v jazyce Java, část v jazyce C). Pomocí konverze vstupně výstupních dat do XML je možné zapojovat již existující simulátory.

Široké spektrum možností využití, otevřenost normy, definice procesů na mnoha různých úrovních, definice pravidel a množství dalších předpisů samotné normy a jejích částí (jednotlivými součástmi standardu jsou relativně rozsáhlé dokumenty: IEEE1516:2010, IEEE1516.1:2010, IEEE1516.2:2010, IEEE1516.3:2010, IEEE1516.4:2010) znesnadňuje zvládnutí celé této metodiky.

6.1.2 Základní pojmy

Federace

Federací se v HLA rozumí celý distribuovaný simulátor (resp. komplexní distribuovaný simulační systém). Pokud je model korektně navržen, je během jednoho simulačního experimentu provozována právě jedna federace.

Federát

Obecně je federátem softwarová aplikace, která je v souladu s normou HLA a může se podílet na simulačním experimentu. Během svého běhu je součástí federace. Může jít o logický proces, dekomponovanou část logického procesu, ale také například vstupní čidlo nebo panel, zobrazovací jednotku, systém pro načítání historických či databázových dat atp.

Objekty

Objektem v HLA rozumíme sadu vlastností, které jsou význačné pro konkrétní entitu. Entita je s vlastnostmi pevně spojena a je definována již v OMT (pojem je zaveden později, neboť k jeho definici je třeba definovat objekt).

V HLA rozlišujeme dva základní typy objektů, a to **vnitřní objekty**, přístupné jen v konkrétním federátu, definovány v SOM části OMT. Druhým typem jsou **interakční objekty**, jejichž definice musí být známa také federaci tak, aby mohlo docházet k jejich změnám. Definovány jsou ve FOM části definičního souboru OMT. Pojmy OMT, FOM i SOM jsou zavedeny níže.

Jednotlivé datové složky objektu se nazývají atributy, což odpovídá standardnímu názvosloví oborů softwarových technologií.

Atributy

Atribut je datová složka definovaného datového typu (základní datové typy i atributy jsou definovány v OMT, o kterém bude pojednáno níže), přičemž tato datová složka může náležet buď k objektu (pak jde o atribut objektu), nebo federátu (potom jde o atribut federátu). Na rozdíl od standardního pojetí atributů z OOP zde neexistuje zapouzdření. Důraz je naopak kladen na transparentnost všech definovaných objektů a jejich atributů.

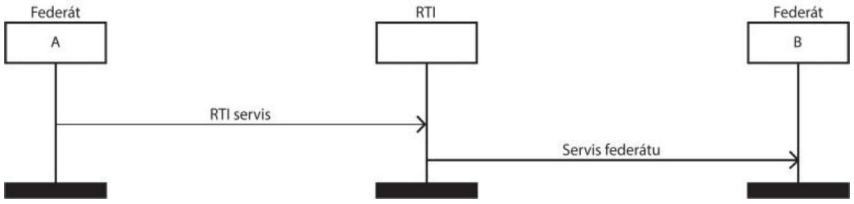
Interakce

Veškerá komunikace v rámci federace probíhá ve formě interakcí. Interakce musí být předem definovány. Každá interakce může mít jeden nebo více parametrů (těmi jsou již zmíněné interakční objekty, nebo atributy interakčních objektů). Protože HLA specifikuje velké množství přípustných typů interakcí, došlo uvnitř standardu ke kategorizaci interakcí (v originálním názvosloví se tyto kategorie jmenují management, např. Federation Management). Základní kategorie (managementy), které HLA specifikuje, jsou následující:

- federace,
- federát,
- objekt,
- vlastnictví,
- čas (resp. časové báze),
- datová distribuce a
- podpůrné funkce.

Interakci je možné provádět přímo prostřednictvím interakčních tříd, například předání objektu dalšímu federátu atp.

Interakce mezi sebou federáty nikdy nerealizují přímo. Vždy musí o provádění interakcí žádat RTI (pojem je zaveden později), která interakci provede. Tento postup je schematicky znázorněn na obrázku 7. Potvrzení je nezávislou službou (anglicky service), která by fungovala stejně jako příklad z obrázku 7, avšak směřovala od federátu B k federátu A. Výjimkou by byl stav, kdy by RTI zjistila, že federát B není pro akci připravený/způsobilý. V takovém případě by vyrozumění o nedokončení podalo přímo RTI.



Obrázek 7: Interakce v HLA[8]

OMT

V HLA je nutné veškeré elementy (v tomto odstavci rozumějme objekty, atributy, interakce a parametry) definovat v takzvaném OMT (Object Model Template). Samotné OMT je strukturováno do dvou částí: FOM a SOM.

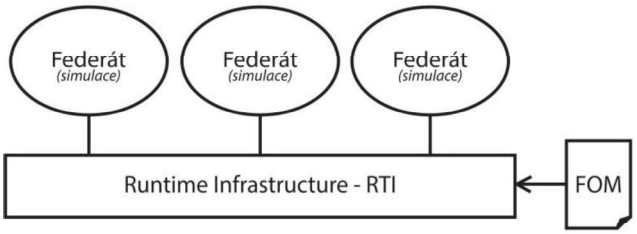
Elementy mají platnost dle svého zařazení do FOM, nebo SOM. Pakliže jsou definovány v části FOM (Federation Object Model) jde o interakční elementy, které prochází simulací a mohou procházet RTI, čili mohou migrovat mezi jednotlivými federáty (vyskytují se tak ve federaci a mohou být subjektem interakce mezi federáty). Elementy definované v SOM (Simulation Object Model) jsou pak dostupné pouze lokálně uvnitř federátu (tj. v simulátoru, nebo sadě simulátorů stejného druhu, které se podílí na celém distribuovaném výpočtu) a nemigrují tak mezi federáty. Důvodem, proč jsou i vnitřní objekty konkrétních simulátorů definovány v SOM je to, že je nutné zachovávat absolutní transparentnost a řiditelnost. Důsledkem tohoto kroku je, že kromě usnadnění řízení a ladění je také mnohem snazší implementace optimistických synchronizačních metod.

Samotné OMT má formu XML dokumentu, je proto relativně snadno čitelné. Avšak tvorba OMT je rozsáhlou záležitostí předepsanou specifikací IEEE1516.2:2010.

[8] Zdroj: (Brožek a kol., 2017)

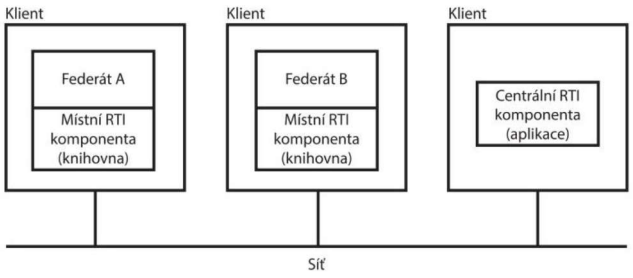
RTI

RTI (run-time infrastructure) je softwarová vrstva, která zajišťuje veškeré komunikační služby federátům. Je definována přímo standardem (funkcionalita, rozhraní, dostupnost aj.), je platformově a jazykově nezávislá.



Obrázek 8: Logické schéma HLA[9]

Pro běh ve federaci potřebuje OMT, přičemž RTI pak může provádět jen základní definované funkce (vytvoření federace, vytvoření federátu, připojení federátu do federace aj.) a operace odvozené od OMT (např. je-li v OMT definována konkrétní interakce, je možné její provedení požadovat od RTI).



Obrázek 9: RTI a jeho fyzická realizace[10]

Při charakterizování architektury HLA je nutné mít na paměti, že její nejvytíženější komunikační částí je právě RTI, což je patrné z obrázku 8. Veškerá komunikace mezi federáty směřuje právě přes RTI. Faktem však je, že RTI není pasivní sběrnice, jak se může na logické úrovni zdát, ale musí jít o aktivní aplikaci, která poskytuje jednotlivým federátům své služby. V důsledku toho se tak distribuovaná simulace stává simulací s centralizovaným, řídicím uzlem, který má potenciál stát se úzkým hrdlem systému. Běžné značení a logické fungování RTI je možné vidět na obrázku 8. Fyzická realizace RTI je znázorněna na obrázku

[9] Zdroj: (Brožek a kol., 2017)
[10] Zdroj: (Brožek a kol., 2017)

9. Pohled na obě výše zmíněná znázornění by měl dostatečně osvětlit rozdíl mezi logickým a fyzickým pojetím RTI. Při porovnání obou obrázků je také možné vysledovat, že přestože je HLA definována jako distribuovaná simulační architektura bez centrálního řízení, na fyzické úrovni je centrálním fyzickým prvkem právě RTI. Je proto třeba důsledně rozlišovat úroveň pohledu na celou architekturu.

Na fyzické úrovni má RTI dvě důležité části (toto je možné vidět na obrázku 9). První z nich je vlastní aplikace RTI (centrální RTI komponenta), která často běží na vlastním výpočetním uzlu a je to právě ta část, která se aktivně stará o komunikaci, připojování federátů k federaci, spravuje objekty atp. Jde o nejvytíženější komunikační část simulátorů[11]. Druhou částí je lokální RTI komponenta. Jde o knihovnu, kterou je nutné přilinkovat k vlastnímu softwarovému řešení, nebo aplikačnímu wrapperu[12]. Knihovny jsou standardně dodávány výrobcem RTI a jsou zpravidla použitelné jen pro implementaci RTI od daného dodavatele. Knihovna funguje jako rozhraní pro volání a zpětná volání mezi centrální RTI komponentou a vlastní logikou každého federátu.

6.2 Architektura DIS

DIS, čili distribuovaná interaktivní simulace (původně, anglicky Distributed Interactive Simulation), je technologie, která vznikla historicky dříve než HLA, ale zároveň je dlouhodobě vyvíjena paralelně s HLA. Historicky šlo dokonce o úspěšnější technologii, neboť je na rozdíl od HLA definována na daleko nižší úrovni a pro méně výkonné hardwarové konfigurace je tak daleko výhodnější. Naopak v současné době (cca od roku 2010) se do popředí dostává HLA, neboť vývoj pro ni je jednodušší a rychlejší.

DIS je definována normou IEEE1278 (2006) a historicky také několika armádními normami (například STANAG[13] 4484ED, SIMPLE[14] STANAG 5602), a to v USA i Evropě, kde byla hojně používána, než byla nahrazena právě HLA.

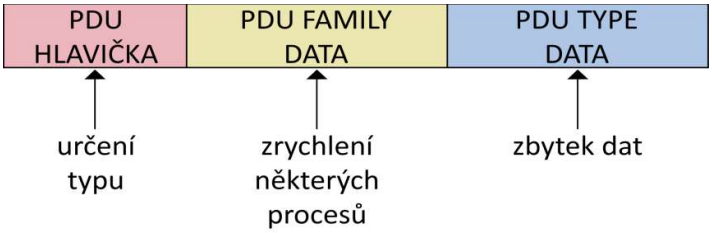
Základní určení DIS je zaměřeno pro tvorbu prostředí pro kooperující simulátory v rámci distribuované simulace. Předpis standardu je skutečně minimalistický a jde de facto jen o síťový protokol, který definuje jednotlivé části packetů, princip práce aj.

Základním pojmem DIS je tak PDU (protocol data unit). Jednoduše řečeno jde o konceptuální tvar packetu, který cestuje po síti. Samotné PDU packety a jejich třídy jsou, z pohledu svých typů, stromově rozděleny podle třídy využití. V jedné PDU rodině (PDU

[11] Avšak je nezbytné upozornit, že je možné vytvářet i jiné, než pouze lineární nehierarchické simulační systémy, čímž je možné v jedné simulaci využívat více centrálních RTI komponent.
[12] Wrapper je zvláštní typ programu, který zabezpečuje spojení mezi pasivním rozhraním a aktivní bránou (Gate). Obecně je za wrapper považována softwarová komponenta, která dokáže zařídit jednomu programu přístup k jinému programu jako k běžně dostupné službě, nebo k softwarovému démonu.
[13] NATO's Standardization Agreement
[14] Standard Interface for Multiple Platform Link Evaluation

Family) jsou společně například všechny operace pro žádosti o vstup do procesu, výstup z procesu, inicializace aktivity a jiné[15]. PDU se šíří po síti broadcastem, nebo multicastem, což však klade velké nároky na síť. Právě šíření packetů broadcastem (resp. multicastem) je významnou nevýhodou DIS.

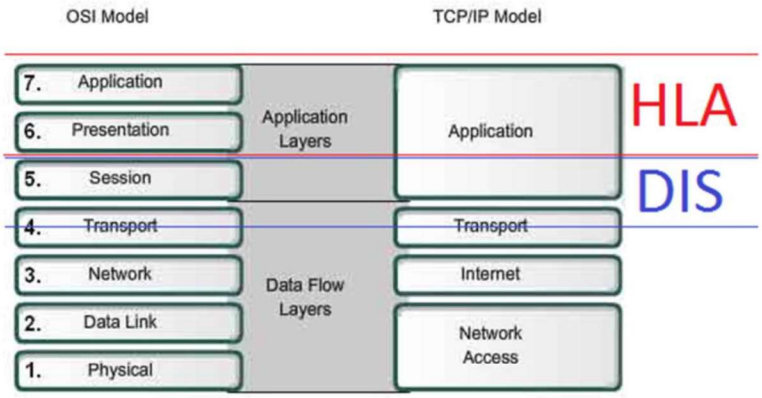
Aby došlo ke zvýšené efektivitě, má předpis pro PDU packet tři části. První částí je PDU hlavička, která identifikuje typ packetu, adresáta a odesílatele a umožňuje tak zahazovat ty PDU packety, které nejsou určeny pro daný simulátor. Další dvě části jsou si podobné a obsahují vlastní data. Druhá ze tří částí, PDU Family, nese data, která jsou společná pro všechna data dané PDU rodiny a slouží ke zrychlení zpracování PDU packetů. Opět slouží k rozlišení, která část simulátoru data potřebuje a často také nese většinu důležitých dat. Poslední třetí část nese data konkrétního PDU rámce. Struktura PDU je uvedena na obrázku 10.



Obrázek 10: PDU Packet[16]

Protože jde o nízkoúrovňové řešení, bylo nahrazeno v praktických použitích právě HLA architekturou. Na obrázku 11 je možné vidět srovnání architektur HLA a DIS z pohledu síťového ISO/OSI modelu a z pohledu síťového TCP/IP modelu. Přestože je tento model důležitý zejména pro softwarové inženýry a pro logiku řešení nemá žádný zásadní vliv, je třeba ho brát v potaz, neboť vysvětluje odlišné náklady na implementaci řešení (nízkoúrovňová řešení za využití DIS jsou pro kodéry pracnější a časové náročnosti jsou až násobně vyšší, než je tomu při využití HLA).

[15] Jako další příklad můžeme zmínit PDU Family Entity, která se dále větví na informace o přesunu, kolizi, stavu, směru aj.
[16] Zdroj: Vlastní



Obrázek 11: Srovnání DIS a HLA z pohledu ISO/OSI[17]

Obrázek 11 znázorňuje vztah referenčního modelu ISO/OSI a TCP/IP modelu vůči technologiím HLA a DIS.

6.3 Programovací přístupy a paradigmata

Mezi způsoby, jak budovat distribuované simulátory na nejnižší (implementační) úrovni, patří také softwarová rozhraní pro vzdálené volání metod a funkcí.

Výhodou těchto rozhraní je jejich jednoduchost. Jako příklad je možné uvést Java RMI. Při zpracování problematiky bylo čerpáno z Reillyho (2015). Aplikační příklady je možné najít v mnoha pracích. Při zpracování rešerše pro potřeby této práce bylo však nutné důsledně rozlišovat mezi případy korektního a nekorektního použití technologie. Špatný přístup je bohužel častým nedostatkem aplikace programovacích technik a technologií.

Řešení slouží k přístupu ke vzdáleným objektům tak, jako by se jednalo o místní objekty. Tedy pokud existuje u objektu X metoda Y na vzdáleném počítači, výše zmíněná rozhraní umožní uživateli pracovat s metodou Y, jako by byla na místním počítači.

Tento přístup má řadu výhod. Protože se celý distribuovaný systém více výpočetních uzlů může chovat jako jedna aplikace, je možné postavit taková softwarová řešení, která budou odpovídat řešení na jednom počítači. Jinak řečeno, pokud by existovalo simulační jádro určené pro jeden výpočetní uzel (tj. nedistribuovaný simulační výpočet), jsme prostřednictvím výše zmíněných technologií schopni provádět takovýto výpočet distribuovaně například pomocí metody dekompozice na logické mikroprocesy, nebo dekompozicí na paralelní procesy. Podstatné však je, že by pro takovýto distribuovaný simulační výpočet nebylo nezbytně nutné řešit problematiku synchronizace mezi jednotlivými uzly simulace.

[17] Zdroj: Vlastní, původní obrázek síťových architektur převzat z Bankse (1998)

Zároveň je však nutné zohlednit fakt, že je prostřednictvím těchto technik možné vytvořit distribuovaný simulátor, který se bude skládat z rovnocenných logických procesů, jež budou mezi sebou řádně synchronizovány. Tento přístup má však již ze svého principu velkou nevýhodu v režii, která je nutná pro běh simulačního výpočtu.

6.3.1 Přímé programování

Metoda přímého programování je alternativou k řešení prostřednictvím existujících simulačních nástrojů. Na nejnižší úrovni, která umožňuje přenos dat mezi jednotlivými logickými procesy distribuované simulace, je možné provést libovolnou implementaci, což může vyhovovat v prostředí, kde je vytvářen specifický simulátor.

Direct programming (vlastně přímé programování) bylo původním základem pro DIS. V situacích, kdy je žádoucí uvolnit vlastní řešení od pevně stanovené normy, je použití direct programmingu vhodné. Avšak tento požadavek je zpravidla výjimečný.

Základními metodami, kterými je možné navazovat spojení, jsou především:

- Řetězcové komunikace (pomalé, datově náročné, jednoduché, intuitivní),
- datové streamy (možnost šifrování, velmi rychlé, logicky složité, nutné mít vlastní stream R/W při použití heterogenních systémů),
- definované datové packety (kromě vlastností předchozí možnosti dochází k dalšímu zrychlení zpracování a je zde možnost zasílat větší množství dat; tato technika slouží jako základ pro DIS),
- XML přenosy (středně rychlé zpracování, intuitivní, využívá se v HLA).

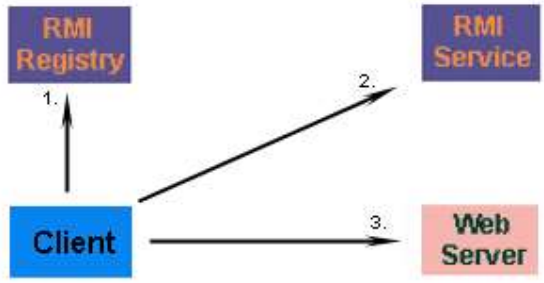
Oproti ostatním zmíněným metodám jsou metody přímého programování výpočetně relativně efektivní a dosahují nejlepších výsledků pro využití v síti (Manlig, 1999). Na druhou stranu je jejich konstrukce poměrně složitá a údržba u komplexnějších systémů velmi náročná. Sporadická bývá také znovupoužitelnost. Výhodou je možnost úpravy programovacích rozhraní a vytvoření bran (Gate) pro jiné technologie, neboť struktura dat je předem jasně dána.

6.3.2 Programování pomocí připravených knihoven

Nejsnazším způsobem, jak programovat distribuovaná řešení, je využití hotových knihoven. Princip těchto knihoven je vždy velice podobný. Knihovny pracují s objekty na vzdáleném výpočetním uzlu tak, jako kdyby šlo o objekty místní.

Jednou z těchto knihoven je Java RMI (Remote Method Invocation), která dokáže prostřednictvím jednotlivých Java JVM tvořit vazby mezi vzdálenými a místními objekty. Synchronizace je však velice pomalá a celé řešení při bližším pohledu složitější. Řešení bylo zkoumáno pro svůj potenciál v distribuovaném programování, avšak nakonec bylo vyhodnoceno jako nevhodné pro použití v distribuovaných simulacích. Topologii je možné vidět na obrázku 12, přičemž více informací o možnostech využití jednotlivých zobrazených

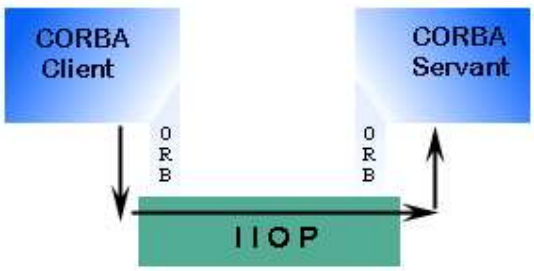
komponentů RMI je možné nalézt v Reillym (2015). Protože technologie byla ze zkoumání kvůli velkému množství slabých stránek vyřazena, není jí v této práci věnována další pozornost.



Obrázek 12: Princip Java RMI[18]

Další často používanou technologií je CORBA (Common Object Request Broker Architecture). Jde o multiplatformní obdobu Java RMI. Její vlastní režie je mnohem nižší než u Java RMI a použití jednodušší. Multiplatformnost navíc přináší zajímavé možnosti využití řešení. Fungování je však totožné – tvoří se vazby mezi vzdálenými a místními objekty a aplikace se chová tak, jako kdyby vzdálené objekty byly místními. CORBA zajišťuje, aby pro aplikaci bylo toto chování zajištěné a transparentní. O fungování CORBA a jeho relativní podobnosti s HLA referuje Rabelo a kol. (2013), přičemž z Reillyho (2015) je převzat obrázek 13, který ilustruje komunikační organizaci CORBA.

Princip fungování je velice blízký principu fungování HLA, avšak v případě rozhraní CORBA nedochází k aktivní režii na straně běhové sběrnice (IIOP). Rozhraní k běhové sběrnici je řešeno prostřednictvím ORB (Object Request Broker), což je middle-ware, který je nutné pro každé prostředí doinstalovat.



Obrázek 13: Princip CORBA[19]

[18] Zdroj: (Reilly, 2015)
[19] Zdroj: (Reilly, 2015)

Ani jedna z těchto technik se však příliš nehodí k tvorbě simulátorů, neboť ani jeden ze dvou základních přístupů, které je možné použít, není optimální. Základními přístupy jsou:

1. Využití vzdáleného ovládání prostředků k přímému řízení toku aplikace. Tato metoda je však režijně velice náročná. Běžně se nepoužívá, přestože je správná.
2. Využití vzdáleného ovládání prostředků k vlastní implementaci synchronizací, nebo předávání zpráv. Navzdory tomu, že se technologie v tomto režimu používá (i pro softwary tvořené na odborných pracovištích), je vhodnější použít první jmenovaný přístup.

7 Implementační a technologická paradigmata

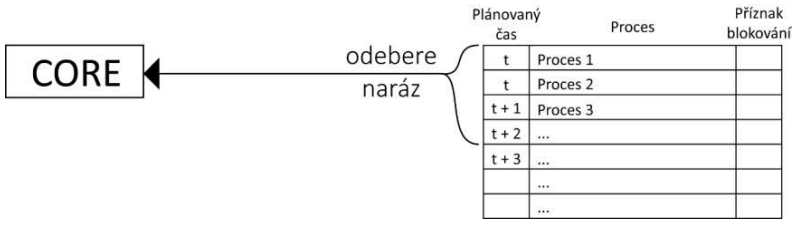
Dizertační práce se v rámci případové studie věnuje budování simulátorů i z pohledu zezdola, tedy vlastního programového kódu. Proto je nutné vytvořit pohled na to, která z programátorských paradigmat jsou v současné době využívána v distribuované simulaci.

7.1 Multiprocesoring

Mnoho simulačních nástrojů dneška je navrženo na základě principů, které jsou platné již několik desítek let (Huntsinger, 1988). Tato paradigmata však často nerespektují měnící se trend vývoje hardware. Ještě před dvaceti lety bylo hlavní cestou ke zvyšování výkonu prosté zvyšování pracovních frekvencí. Paralelismus byl ještě dlouho po svém příchodu doménou zejména zpracování obrazu (tj. grafických karet) (Chandy a Misra., 1981).

Moderní simulační nástroje se snaží však jít dál a překonat staré principy, neboť se stále častěji začíná využívat paralelních výpočtů.

Mezi základní způsoby, jak simulátoru umožnit zpracovávat paralelně několik procesů, patří využití superskalárního simulačního jádra. To může rozšiřovat oba základní způsoby implementace simulačního jádra (využívající buď kalendář událostí, nebo seznam aktivit). Jeho základní princip spočívá v tom, že pokud jsou procesy nezávislé (tedy výsledek jednoho nemůže mít dopad na výsledek jiného), je možné je zpracovávat paralelně. Z příslušné datové struktury se tak namísto jedné události (resp. aktivity) odebírá větší množství naráz. Odebíráním volných procesů v dávce je dosaženo nižších výpočetních časů. Popsaný princip je možné vidět na obrázku 14.

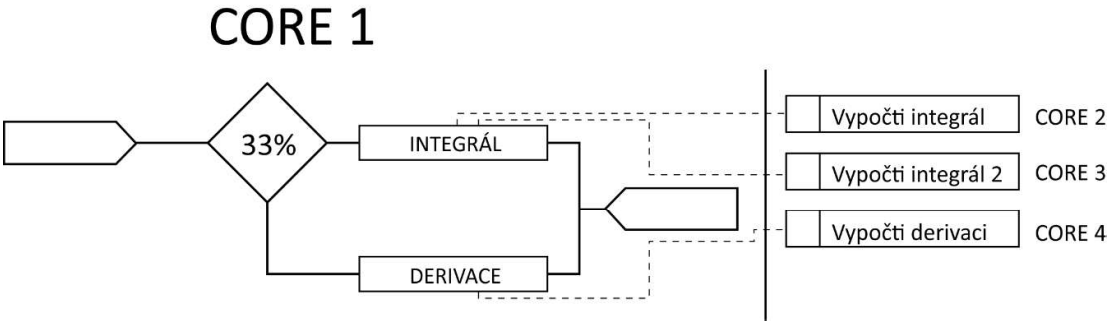


Obrázek 14: Superskalární simulační jádro[20]

Dalším základním principem pro paralelní zpracování dat je procesní servisní halda (Process Heap, nebo také Data Service Container Process Heap (Microsoft, 2010)). Její mechanismus je podobný principu vláknové haldy (Thread Pool (Oracle, 2015)) v moderních programovacích jazycích. Jednoduše je pouze určeno, které procesy je možné paralelizovat a zároveň, kolik maximálně vláken je možné využít. Vnitřní funkce programovacího jazyka (resp. virtuálního stroje JVM, nebo .NET) se pak samy postarají o to, které procesy budou běžet paralelně a kolik jich bude třeba. Tedy, využijí standardního chování při aplikaci

[20] Zdroj: Vlastní

principu poolingu. Optimalizace paralelního běhu je v takovémto případě relativně obtížně řiditelná, avšak algoritmy, které se o tento běh starají, jsou povětšinou velmi dobře realizované (například navíc umožňují práci s prioritou). Celý princip je znázorněn na obrázku 15, kde se jedno vlákno stará o standardní průchod entit simulátorem, avšak výpočetně velmi náročné procesy jsou prováděny v samotných vláknech.



Obrázek 15: Procesní servisní halda za běhu výpočtu[21]

Velmi častým a pro mnoho případů nejefektivnějším způsobem je paralelizace na úrovni mikroprocesů (úzce souvisí s tématem dekompozice na mikroprocesy, avšak tato technika je s výhodou využitelná také pro některé třídy agentových simulátorů). Tento způsob běhu je vhodný nejen pro procesory a cloud, ale také například pro evoluci výpočtu na grafických kartách. Aby byl běh efektivní, je třeba splnit především následující podmínky:

- Existuje malé množství tříd, ale velké množství instancí.
- Počet variací volaných metod je nízký, ale jsou volány velmi často.
- Mezi instancemi vzájemně nebo mezi nimi a prostředím dochází k častým interakcím.
- Nedochází k častým interakcím mezi instancemi entit zahrnutých do paralelních výpočtů a instancemi entit nezahrnutých do paralelních výpočtů.
- Větší část entit má vzájemně stanovenu volnost.

Splněním výše uvedených podmínek je možné zajistit superskalární zpracování dat v simulačním modelu, a to až do úrovně stovek či tisíců paralelně běžících výpočtů.

7.2 Migrace LP po uzlech

Proto, aby bylo možné dosáhnout migrace jednotlivých logických procesů po uzlech, je třeba nejprve provést jejich miniaturizace. Miniaturizace (nebo také atomizace) logických procesů (přesněji miniaturizace logických procesů nebo dekomponovaných částí logického procesu) úzce souvisí s tématem paralelního zpracování dat a případně s možností migrace těchto částí po různých výpočetních uzlech. Miniaturizace spočívá v tom, že množství režijních dat

[21] Zdroj: Vlastní

a informací musí být minimální. Ideálním způsobem miniaturizovaný logický proces je možné popsat například prostřednictvím funkcionálního programování, nebo jednoduchou matematickou funkcí. Tohoto stupně dekompozice není často možné dosáhnout, proto je standardní metrikou pro určení miniaturizace velikost dat, která je třeba přenést v případě, že by LP měl být prováděn na jiném uzlu.

Současné technologické trendy otevírají cestu k velmi specifickým způsobům, jak balancovat výkon distribuovaného simulátoru, a to je migrace logických procesů, agentů, nebo dekomponovaných částí logických procesů po výpočetních uzlech během simulačního výpočtu.

Samotná migrace vede velmi často k nutnosti po omezenou dobu zastavit výpočet. Výhodou však je možnost autonomního řízení vytížení jednotlivých výpočetních uzlů.

Techniky nutné pro tento způsob výpočtu jsou využívány už několik desítek let (první přibližně v roce 1993), tehdy však sloužily k distribuci výpočtu pro různé komplexní programy. Uživatel například mohl na svůj osobní počítač nainstalovat aplikaci, která propůjčovala část jeho výpočetního výkonu k dekódování lidského genomu, nebo k výpočtům astronomickým (už tehdy pro expertní systémy nebo simulace). Když se pak uživatel rozhodl svůj počítač využívat k výkonově náročným činnostem, tak část prací, které jeho stroj vykonával pro distribuovaný výpočet, prostě odmigrovaly na jiný fyzický stroj.

Pro potřeby simulací v měřítku do desítek fyzických strojů však dozrály techniky simulace až v poměrně nedávné době. Problematika úzce souvisí právě s možností paralelního zpracování dat (neboť migrující výpočet využívá velmi podobných technik) a metodami pro dekompozici.

V současné době je možnost migrujících a nemigrujících logických procesů spíše otázkou preference, neboť jejich nasazení je pouze otázkou přidání několika knihoven. Systém řízení je pak zcela autonomní.

Při zpracování textu bylo čerpáno a více informací uvádí Dubitzky a kol. (2012).

7.3 Clustery, cloud

Dostupnost velmi silných výpočetních uzlů je velkou šancí pro současné simulační techniky. V současné chvíli se předpokládá velký potenciál zejména pro zpracování mikroprocesových simulací, případně agentových simulací. Obecně lze říci, že každý výpočetně náročný distribuovaný simulátor je možné provozovat na heterogenních strojích. Klíčové a časově náročné výpočty tak mohou běžet v cloudu či na clusteru a zbytek pak dle prostředí a dle potřeby (např. interaktivně v emulovaném prostředí). Při zpracování textu bylo čerpáno a více informací uvádí Dubitzky a kol. (2012).

7.4 Pokročilé techniky vizualizace

V minulosti byly požadavky na grafické výstupy simulátorů kladeny spíše na ty interaktivní (a ani zde to nebylo podmínkou). Nyní však nároky z tohoto pohledu rostou. Interaktivní simulátory jsou ideální takové, které emulují realitu, nebo jejichž kvalita obrazu odpovídá de facto reálnému pohledu (tj. ovládání je stále na počítači, ale kvalita vizualizace se velmi blíží vizuálním vjemům reality).

Výpočetní požadavky na vizualizaci tak velmi často překonávají výpočetní požadavky na samotný simulátor. I toto je alternativa, se kterou je nutné počítat a zvolit vhodné řešení pro simulátor, který nakonec může výpočetně čekat na vizualizaci. Při zpracování textu bylo čerpáno a více informací uvádí Dubitzky a kol. (2012).

7.5 Interaktivní zásahy a trenažéry

Současné simulace jsou stále častěji používány v kontextu trenažérů, případně přímo v kontextu emulace reality. Právě proto je nutné vytvořit velice kvalitní softwarová řešení pro interaktivní zásahy, a to nejen z tradičních vstupně výstupních zařízení, ale de facto z libovolného zdroje (není výjimkou například analýza pozice na základě obrazu z kamery). Obvyklé užití interaktivních zásahů je možné najít ve specifických simulátorech nazývaných trenažéry.

Trenažéry jsou ve vývoji již mnoho desítek let. Demonstrátory simulátorů typu trenažér je relativně snadné nalézt na mnoha místech, a to od intuitivních řešení (např. simulátor řízení osobního automobilu) až po velmi komplexní strategicko-taktické trenažéry, které využívá armáda.

Podoba trenažérů se liší podle jejich účelu i rozsahu. Pokud bude však úhel pohledu směřován k distribuované simulaci, je nejklasičtějším zástupcem vojenský heterogenní taktický simulátor. Relativně zajímavá verze tohoto řešení je k dispozici v Centru simulačních a trenažérových technologií v Brně, na letišti v Pardubicích a v bojovém prostoru Vyškov. Armáda má k dispozici jednak simulační prostor pro generální štáby (pro definici strategií) a následně několik sad simulátorů pro bojová vozidla, tanky, helikoptéry a letadla. Propojením všech těchto zařízení do jedné simulace pak vniká trenažér pro velké množství uživatelů naráz.

Samotná implementace trenažéru je náročná zejména na vizualizaci a na řešení interaktivních zásahů. Zvláštní zřetel si u aplikací typu trenažér (tedy řešení v reálném nebo téměř reálném čase) zaslouží synchronizační mechanismy. Konzervativní synchronizační metody totiž mohou (při velkém počtu účastníků, při pomalé síti) způsobovat nepříjemná zamrzávání (Freezing, nesprávně nazýváno také jako Lagy/Lagování). Optimistické synchronizační metody (Bruzzone a kol. 2010), (Letizia a kol., 2014), (Letizia a kol. 2015) jsou komplikovanější na implementaci, ale výše zmíněným nedostatkem netrpí.

8 Techniky pro implementaci interaktivních zásahů

Pro vysvětlení technik pro implementaci interaktivních zásahů je třeba vymezit několik základních pojmů úzce specifických pro tuto oblast. Klíčovým pojem je parametr, resp. parametrizace. Parametrizace je proces přidělení parametru proměnné/entitě/procesu. Entita/proměnná/proces potřebuje parametry proto, aby se mohla chovat nekonstantně. Nekonstantní chování umožňuje simulátorům studovat různé situace, provádět evoluci výpočtu atp. Při zpracovávání kapitoly bylo čerpáno z: (Roubtsova, 2016), (Korn, 2011), (Letizia a kol., 2014) a především (Popovici a Mosterman, 2012).

Simulátor je možné parametrizovat různými způsoby, které je možné kombinovat. Proto, aby bylo možné vytvořit alespoň určitý ucelený pohled, je provedena kategorizace různých typů zásahů. Tato kategorizace je provedena na základě výše uvedené literatury – přičemž některé kategorie jsou v literatuře uvedeny explicitně a s některými je pracováno spíše implicitně.

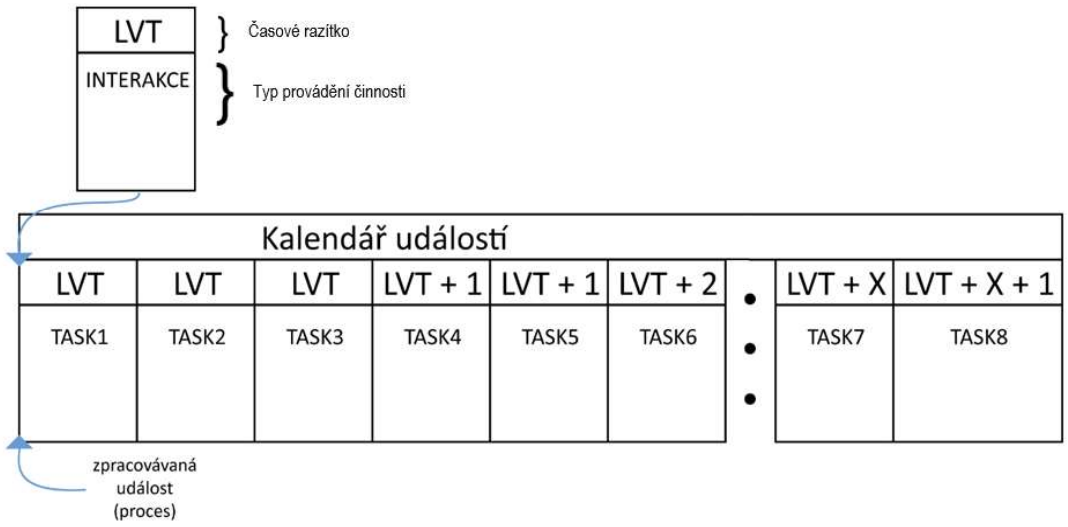
Simulační modely je možné parametrizovat **staticky** (před spuštěním simulačního výpočtu), nebo **dynamicky** (za běhu simulačního výpočtu). Dynamickou parametrizaci je možné dále rozlišovat dle použitých metod na parametrizaci **deterministickou** a **stochastickou**. Další, odlišná kategorizace (která sleduje jiné vlastnosti interakcí) se zaměřuje na způsob iniciace parametrizace. V případě, že je hodnota parametru zavedena autonomně softwarem, jde o **automatickou** parametrizaci, pokud je parametrizace provedena externím podnětem (zpravidla ručně uživatelem), jde o parametrizaci **interaktivní**. Parametry je dále možné odlišit podle místa iniciace parametrizace. V případě, že simulátor používá vnitřní data (generátory pseudonáhodných čísel, databázi s daty) jde o **offline** simulátor. Pokud je simulátor (resp. simulující systém) libovolným způsobem propojen na simulovaný systém, nebo na vstupně výstupní zařízení emulovaného prostředí, jde o **online** simulátor.

Samotná technika interaktivních zásahů se může odlišovat. Interaktivní systémy můžeme klasifikovat na systémy tvrdě interaktivní (hard realtime interactive) a měkce interaktivní (soft realtime interactive). Pro tvrdě interaktivní systémy existuje velmi pevná vazba mezi uživatelským vstupem a důsledky těchto vstupů. Literatura (Popovici a Mosterman, 2012) uvádí, že přijatelné latence uživatelských vstupů a projekce těchto vstupů do parametrů je 0,20 sekundy reálného času. Tedy, od okamžiku uživatelského zásahu do provedení projekce uživatelské interakce do stavového prostoru simulátoru nesmí uplynout více, než 0,20 sekundy. Tvrdě interaktivní systémy jsou charakteristické tím, že jejich simulační čas má pevnou vazbu s reálným časem, přičemž požadavek na tyto systémy jsou vazby odpovídající násobku rychlosti plynutí simulačního času dosahující hodnot 0,8 až 1,2 času reálného (tj. čas v simulaci může plynout nejméně rychlostí 0,8 a nejvíce 1,2 násobku času tak, jak ho vnímají lidé).

Měkce interaktivní systémy jsou takové interaktivní systémy, u nichž není nutné požadovat striktní dodržení podmínek pro tvrdě interaktivní systémy. Po definici systémů a požadavcích na jejich vazby je možné uvést jednotlivá implementační řešení.

8.1 Interaktivní zásahy s přerušením

Základní technikou tvrdé interaktivity je využití interaktivních zásahů s přerušením. V okamžiku uživatelského zásahu je okamžitě ukončena zpracovávaná činnost (nejčastěji proces), avšak vzhledem k obecnosti je v rámci kapitoly využíván původní pojem Task (Banks, 2010) resp. český pojem aktivita), přičemž dojde k navrácení vnitřního stavu systému před okamžik zpracování této činnosti, včetně nastavení kalendáře událostí tak, aby aktuálně zpracovaná činnost byla první v tomto kalendáři. Dalším krokem (oproti běžnému běhu programu) však není standardní zpracování další úlohy z kalendáře, nýbrž zavedení další úlohy do kalendáře, kterou je právě uživatelská interakce. Potom již dojde k běžnému zpracování první události z kalendáře událostí. Princip je ilustrován na obrázku 16.



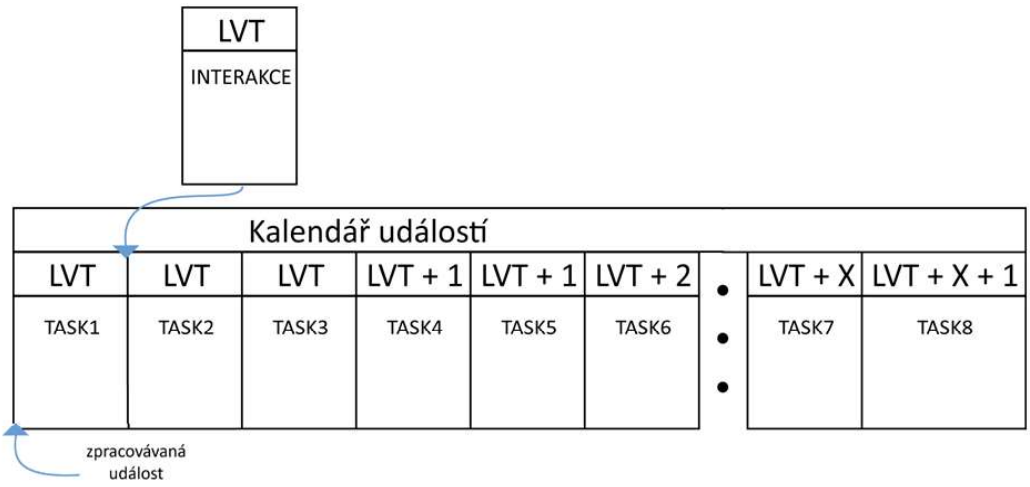
Obrázek 16: Interaktivní systém s přerušením[22]

Tato technika je výhodná především, pokud jsou zpracovávané činnosti výpočetně (tj. i časově) náročné a probíhají striktně sériově (tj. vždy je odebírán a zpracováván právě jeden proces). U simulátorů s paralelním zpracováním činností, nebo u simulátorů s krátkými procesy je výhodnější použít interaktivní systém bez přerušení.

[22] Zdroj: vlastní

8.2 Interaktivní zásahy bez přerušení

Nejjednodušší technikou je aplikace interaktivních zásahů bez přerušení běhu aktuálně simulované aktivity. Na rozdíl od systému s přerušením nedochází k násilným stavovým změnám, ani neočekávanému chování systému. Uživatelská interakce je v tomto případě zařazena do kalendáře událostí za aktuálně zpracovávanou úlohu (pokud je před zpracováním úlohy tato úloha z kalendáře událostí odebrána, potom se přidá přímo na vrchol kalendáře událostí). Princip zobrazen na obrázku 17. K řádnému pochopení obrázku je třeba uvědomění, že TASK1 je aktuálně zpracovávaná úloha v čase LVT. Uživatelská interakce je tak přidána za aktuálně prováděnou úlohu (a nepřeruší tak její vykonávání), avšak přidá se před všechny další úlohy, které měly být zpracovány v čase LVT.



Obrázek 17: Interaktivní zásah bez přerušení[23]

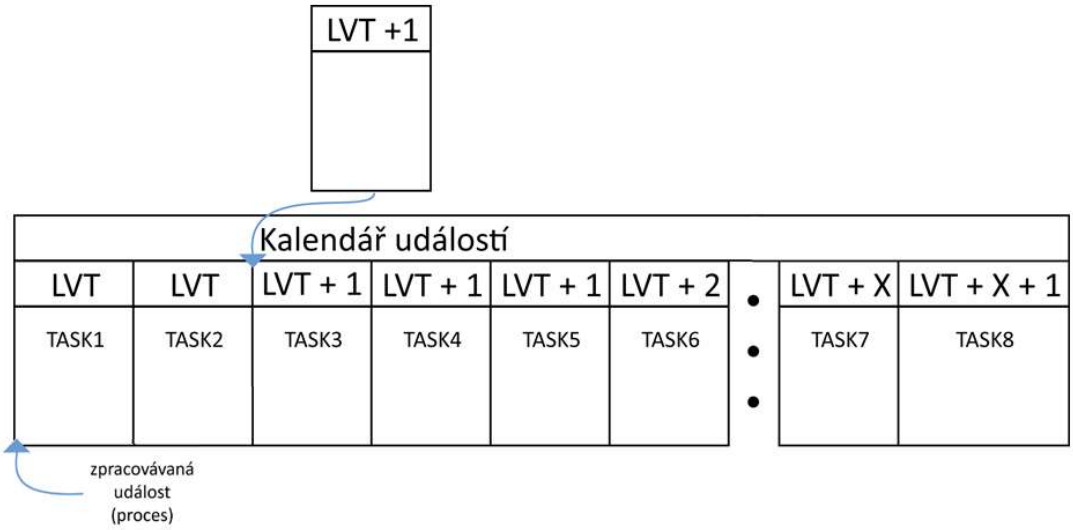
Na rozdíl od přecházející metody s přerušením se může jevit vysoké riziko toho, že zpracovávaná událost TASK1 změní stavový prostor jinak, než na co uživatel svou interakcí reagoval. Toto je skutečně přípustné – a při dodržení reakční doby 0,20 sekundy reálného času je riziko minimalizováno.

8.3 Interaktivní zásah s elementárním odstupem

Základní princip řešení interaktivních zásahů s elementárním odstupem rozšiřuje techniku interaktivních zásahů bez přerušení. Změna oproti této metodě je v tom, že dojde k dokončení všech operací pro dané časové razítko. Tento krok má velkou hodnotu při pozdější validaci řešení. Běžně není možné deterministicky určit, který z procesů se stejným časovým razítkem bude zpracován dříve, než ostatní procesy s tímtéž časovým razítkem.

[23] Zdroj: Vlastní

Pokud je přidán proces interaktivního zásahu, není možné určit, které procesy již byly zpracovány, a z kalendáře odstraněny a proto není snadné provést validaci, nebo proces after action review. Právě tento problém je vyřešen prostřednictvím algoritmu interaktivního zásahu s elementárním odstupem. Princip je vidět na obrázku 18.

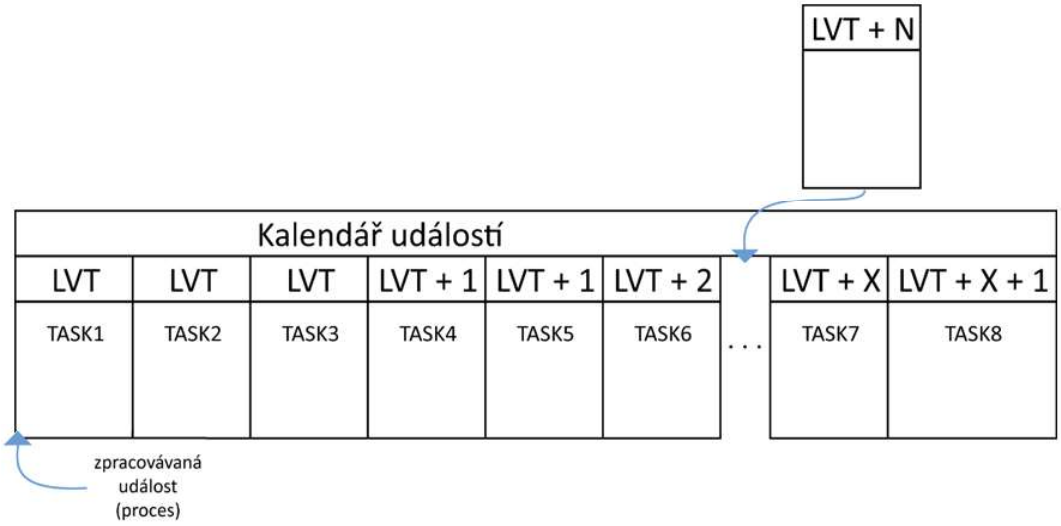


Obrázek 18: Interaktivní zásah s elementárním odstupem[24]

8.4 Interaktivní zásah s výhledem

Poslední ze základních technik interaktivních zásahů je plánování interaktivního zásahu s výhledem (rovněž bez přerušení). Jde o zvláštní způsob tvrdě interaktivního zásahu, neboť je splněna podmínka tvrdé interakce (tj. projekce do stavového prostoru simulace v krátkém čase), ale zároveň dochází k projevu této projekce až v čase pozdějším. Řešení je využitelné zvláště v případech, kdy simulovaný systém svou povahovou obsahuje takové procesy, které není fyzicky možné realizovat ihned. Jako příklad lze uvést letecký simulátor pro vzdušné souboje. Dvě letadla, pokud po sobě pálí za hranicí viditelnosti, nejsou sestřelena (resp. nejsou konfrontována s raketou) ihned, nýbrž až po uplynutí určitého času. V takovém případě je třeba tuto přímou konfrontaci plánovat s výhledem po provedení interaktivního zásahu. Princip plánování interaktivního zásahu posunutého o časový skok *n* je možné vidět na obrázku 19.

[24] Zdroj: Vlastní



Obrázek 19: Interaktivní zásah s výhledem[25]

Interaktivní systém s výhledem je možné výhodně použít v distribuovaných simulátorech, kde jsou interaktivní prvky online systémem nepřímo spojeny s vizualizačním zařízením. Dochází pak k iluzi volnější vazby, která však má veškeré výhody vazby těsné (snazší verifikace a validace, lepší předvídatelnost, plnění ISO norem na jakost softwaru pro bezpečnost atp.).

[25] Zdroj: Vlastní

9 Standardy využitelné při tvorbě softwaru a v simulaci

Standardy mají v současné době významný vliv. Vzhledem k široké dostupnosti znalostí je relativně snadné implementovat softwarová řešení. Vlastní způsob implementace však může být pro různé autory diametrálně odlišný a to přestože používají stejný programovací jazyk. Taková situace vede k tomu, že kód je neudržovatelný a nepřenositelný. Zapojení dalších/jiných zaměstnanců do takové implementace je téměř vyloučeno. Situace je lepší za podmínky, kdy programátoři při implementaci dodržují dobré zásady (přičemž široce používaný pojem dobré zásady je diskutabilní), nebo zásady instituce. Odstranění (nebo alespoň zlepšení) problému přináší až použití standardů, které umožní zajistit unifikaci procesů, názvů a postupů.

Právě kontext, týkající se vlastní práce (resp. vlastní metodiky a fakt, že je nutné respektovat již zavedené standardy, jsou motivem pro rešeršní zkoumání tohoto tématu.

9.1 Standardy využívané v softwarovém inženýrství

Standardů pro obor softwarového inženýrství je velké množství a často se liší podle zaměření. Standardy se tak věnují procesu návrhu softwaru, implementaci i testování.

Pro dokumentaci softwaru existuje například norma IEEE830, která se zabývá specifikací požadavků na systém. Požadavky jsou tak definovány nezávisle na dalších procesech pro vývoj a uplatnitelné jsou standardní metodiky jako UP a RUP, nebo agilní metodiky, jako SCRUM. Při návrhu je třeba dodržovat požadavky na kvalitu, kromě všeobecně známého předpisu ISO9001 se doporučuje využívat také IEEE730 Software quality assurance.

Dalšími zohledněnými návrhovými standardy jsou IEEE1016 Software design description a IEEE1058 Software project management, právě poslední zmíněný standard pomohl v metodice unifikovat styk s objednatelem a určit smluvní role.

Pro testování softwaru existovala dlouhodobě řada standardů. Především jde o standard IEEE1012 Software verification and Validation, který pojednává o praktikách pro ověřování softwaru a v jazyce anglickém specifikuje slova verifikace, validace a rozdíl mezi nimi – což je často diskutované téma. Součástí sledovaných norem je mimo jiné ISO 829 Software test documentation, která se věnuje způsobu testování softwaru a dokumentaci těchto testů – což je klíčové především pro dodržování systémů kvality. V roce 2014 byla celá problematika ucelena ve standardu ISO 29119 (standard a jeho části jsou stále ve vývoji). Celá norma byla zohledněna především v nosné části dizertační práce – to znamená v původní metodice *Metodis*.

Součástí řešení musí být také uživatelská příručka dána předpisem IEEE 1063. Tento dokument popisuje klíčové kapitoly uživatelské příručky.

9.2 Standardy využívané v simulaci

V oboru počítačové simulace je možné najít řadu metodik pro vývoj. Metodiky se nejčastěji zaměřují na logickou a fyzickou úroveň řešení. Příkladem mohou být prezentované standardy IEEE1516:2010 a IEEE1278. Obě technologie nebudou dále diskutovány, neboť jim již byla věnována pozornost v kapitole 6.

Za chronologicky poslední jde označit standard: INACSL Standards of Best Practice: SimulationSM Simulation Design, vydaný v roce 2016, který se zaměřuje na problematiku návrhu simulátorů z logické úrovně – omezení je patrné zejména pro medicínskou techniku, i když většina doporučení je široce uplatnitelná. Rozšíření standardu přišlo v říjnu 2017 pod názvem INACSL Standards of Best Practice: Simulations (kterézto k definici využívají 8 samostatných monografií), které se vztahuje k dalším tématům v simulaci.

10 Simulátory a jejich nasazení

Pro tvorbu metodiky je klíčová znalost již existujících simulátorů, jejich aplikačních domén a dalších faktů (použitých technologií, architektur aj.). V této části práce jsou proto prezentovány základní informace o skutečně používaných simulátorech. Kapitola vychází z rozhovorů s odborníky v ČR i zahraničí (v rámci studia bylo navštíveno deset dalších univerzit či univerzitních pracovišť, z toho čtyři v ČR, dvě ve Velké Británii, dvě na Slovensku a po jedné v Polsku, Itálii a Francii). Informace byly dále získány na konkrétních prezentacích a ukázkách (např. během mezinárodních konferencí, nebo přímo na pracovištích např. Taktické simulační centrum v Pardubicích). Současně došlo k rešerši v periodicích a sbornících dostupných prestižních konferencí. Poslední významný studijní materiál pochází z USA – kde díky spolupráci ozbrojených složek (United States Army Simulation and Training Center) a univerzit (především University of Iowa) vznikl přehledový dokument monitorující využívání jednotlivých technologií pro simulaci. Konkrétní zdroje (vyjma výše uvedených osobních rozhovorů a prezentací), ze kterých kapitola čerpá jsou: (NACSL, 2014), (IDANS, 2011), (Gustavsson a kol., 2009), (Skrzypecki a kol., 2018), (Office of the Secretary of Defense, 2005), (Pierzchala, 2016), (Ryan a kol. 1998), (Jadrić a kol., 2014), (Doshi, 2006), (Bruzzone a kol., 2010), (Elbadi a kol., 2016), (Onggo, 2010).

10.1 Geografické členění

V současném globalizovaném světě existují jen mírné geografické rozdíly v aplikaci simulátorů a simulačních technologií, na drobné nuance je přesto žádoucí se zaměřit. Světovým lídrem ve výzkumu a vývoji jsou Spojené státy americké (mimo jiné původci většiny výše zmíněných technologií). Zastoupení technologií v USA je ve všech segmentech, vyjma zastaralých technik direct programmingu. Specifikem USA je fakt, že kromě HLA je zde stále velmi aktivně využívána DIS. Obrázek reflektující klíčové statistiky je uveden v příloze.

Vývoj simulací v Evropě je soustředěn zejména kolem armádního výzkumu a vesmírného výzkumu. Z toho také plyne, že státy NATO a státy s aktivním výzkumem vesmírných technologií jsou mnohem aktivnějšími uživateli pokročilých simulátorů než státy ostatní. Evropskými lídry v aplikaci simulace jsou Itálie, Francie a Velká Británie, přičemž další významná centra jsou v Německu, Španělsku, Švédsku a Švýcarsku.

Velmi důležitou součástí evropského výzkumu a vývoje v oboru simulací je aplikace simulačních technologií ve zdravotnictví. Právě v tomto oboru je Evropa světovým lídrem. Na výzkumu se podílí několik evropských vědeckých center. Dramatický rozvoj pro využití simulačních technologií je možné sledovat v Číně, která soustředí značné prostředky na vývoj vlastních technologií nebo adaptaci obecných technologií pro vlastní podmínky (Čína například vytvořila vlastní RTI pro HLA).

Mezi další důležitá mimoevropská centra patří kromě Číny zejména Japonsko, Jižní Korea, Izrael, Jihoafrická republika, Rusko, Indie a z amerického kontinentu kromě USA také Kanada, Brazílie a Kolumbie.

V České republice existují dvě hlavní třídy institucí, které se na simulaci soustředí a těmi jsou Armáda ČR a vysoké školy České republiky. Armáda má dvě hlavní centra, jimiž jsou Centrum leteckého výcviku Pardubice (které patří ke světovým špičkám, využívá simulátory navržené dle norem STANAG, hardware je pak řešený na zakázku a skládá se z běžně dostupných zařízení), druhým centrem je Centrum simulačních a trenažérových technologií Armády ČR (přičemž odborníci, kteří zde působí, jsou paralelně aktivní na Univerzitě Obrany v Brně). Primárně jsou nasazeny systémy armádních řešení od společností Lockheed Martin, dále je kladen důraz na vlastní vývoj v oblasti DIS a aplikace vlastních simulátorů v HLA a jejich propojení s ostatními systémy NATO.

Univerzity v ČR obecně experimentují s mnoha různými technologiemi i paradigmaty, avšak bez vysoké míry specializace. Vývoj probíhá na ČVUT, VUT, ZČÚ, Univerzitě Obrany v Brně, VŠCHT a samozřejmě na Univerzitě Pardubice. Dílčí výsledky jsou v několika dalších institucích, například CERVO institutu, Českém hydrometeorologickém ústavu a Českém vodohospodářském ústavu.

10.2 Členění dle aplikační domény

Simulační technologie jsou v současné době využívány napříč obory. Některé obory jsou však tradičně silnější. Mezi nejsilnější obory patří samozřejmě ozbrojené složky, a to především armáda, která se zaměřuje na simulátory odpovídající normě STANAG (v současné době tedy používají zejména HLA, ukončována je podpora DIS). Vojenské simulační laboratoře se zabývají simulacemi strategickými a taktickými, trenažéry a také tzv. živou simulací. Způsob dekompozice nebo návrhové architektury bohužel nebylo možné zjistit. Z dílčích publikací expertů (US Department of Defense, 2014), kteří se podílí na výzkumu a vývoji, lze s vysokou pravděpodobností předpokládat, že je užíván princip agentově orientované simulace.

Dalším oborem, v němž jsou velmi často využívány simulační technologie, je obor chemie, biochemie a zdravotnictví. Mnoho vyvíjených simulátorů si je velmi podobných (vyjma trenažérů, které jsou silně pěstovanou doménou zdravotnictví). Jde o silně výpočetně orientované simulátory, které jsou velmi často úzce specializované a na rozdíl od ostatních oborů je relativní problém s komerční dostupností simulátorů této třídy (opět s výjimkou zdravotnických simulátorů, které jsou finančně velmi nákladné, avšak tržně dostupné). Využívané simulátory jsou velmi často monolitické, avšak dekomponované metodou dekompozice na logické mikroprocesy, nebo (a to především v minulosti) systémem dekompozice na paralelní procesy. Co se týká vývoje, probíhá velmi často pomocí direct-

programmingu. Jednodušší systémy jsou samozřejmě dostupné v předprogramované verzi a jednotlivá pracoviště pak tvoří pouze simulující systémy v hotových nástrojích.

Důležitým oborem je vesmírný výzkum, který je možné kategoricky dělit na dvě části. První jsou simulátory vesmírných dějů a událostí, které jsou svými principy a aplikovanými algoritmy velmi příbuzné s meteorologickými simulátory. Druhá část simulátorů se zabývá konstrukčním a materiálovým inženýrstvím. Algoritmy, postupy a paradigmata, která jsou vyvíjena v tomto oboru, se následně velmi často používají v dalších průmyslových odvětvích (letecký, drážní a automobilový průmysl). Samotné použité technologie významným způsobem souvisí s místem, kde k vývoji dochází. Primární vývoj je veden ve Francii, Británii a Itálii, kde se využívá HLA architektura v kombinaci s komerčními produkty. Tyto státy (a jejich instituce) jsou navíc průkopníky ve využívání cloudů, clusterů a serverových farem pro simulační výpočty.

Dvě další oblasti, které využívají simulátory, jsou doprava a průmysl. Velmi často jde o systémy pro podporu strategického a taktického plánování, méně často je možné setkat se také se systémy pro podporu operativního plánování. S rostoucími výkony počítačů stoupá ekonomická výhodnost těchto řešení.

10.3 Komerční simulátory

V komerční sféře můžeme sledovat dva zásadní proudy. První představují nadnárodní korporace, většinou vlastněné zbrojními giganty, které se soustředí na vývoj prostředí pro tvorbu simulačních modelů. Tato vysoce specializovaná pracoviště jsou špičkou ve svém odvětví, avšak pro širokou veřejnost i akademiky jsou absolutně nedostupná. Výjimkou jsou kurzy pro jimi vyvíjené softwary, které jsou cenově nasazeny mimo možnosti akademické sféry.

Druhý proud v komerční sféře představuje prosté zpracovávání analýz a simulačních studií komerčními subjekty. V tomto případě již nejde o špičková vývojová pracoviště, nýbrž o společnosti, které pouze uživatelsky využívají vyvinuté softwary.

10.4 Akademické simulátory

Akademická sféra je velmi specifická, neboť se v ní mísí všechny vlivy a technologie zmíněné ve výše uvedených kapitolách, vyjma detailnějšího náhledu do vojenských technologií. Tento jev je způsoben tím, že mnoho pracovníků specializovaných firem či laboratoří se velmi často paralelně vyskytuje v akademickém prostředí.

Jediná věc, se kterou se není možné setkat na opravdu špičkové úrovni (avšak profesionální řešení existují), je tvorba vlastních softwarů pro tvorbu simulačních modelů. Vývoj vlastních simulátorů či simulačních jader je spíše výjimečný. Mnoho z akademických simulátorů se však postupně, prostřednictvím komerčních společností, stává úspěšnými řešeními.

10.5 Shrnutí

Současné trendy jsou velmi nakloněny řešení problémů prostřednictvím simulace, a to od malých problémů (řešení evakuace budovy, propustnost infrastruktury, nebo návrhu křižovatky) až po velmi komplexní řešení (trenažéry, modely pro předpověď počasí, ekonomické modely, modely šíření chorob, chemických reakcí aj.). Tyto trendy jsou podporovány růstem výpočetních výkonů počítačů a moderních počítačových architektur (cluster, cloud, aj.), které jsou schopné těchto výkonů maximálně využít.

Na trhu je v současné době velké množství produktů pro podporu budování simulačních modelů, nebo vlastních simulátorů.

METODIKA A POSTUP JEJÍHO NÁVRHU

11 Metodika jako výstup VaV činnosti

Vlastní výklad, nebo definice pojmu metodika je relativně volný a literatura v něm není zcela jednotná. Navíc je pojem metodika často mírně odlišný pro různé obory. V českém jazyce pak běžně dochází ke splynutí pojmu Metodika a Metodologie, přestože jde o různé pojmy. Tato práce chápe pojem Metodika tak, jak je chápán Metodikou pro hodnocení výzkumných organizací platné v době zpracování práce. (Definice výsledků, 2016)

11.1 Pojem metodika

Metodika je popis postupů (metod), který obsahuje základní prvky metodiky, tedy kdo, co, komu, proč a jak. (Navrátilová, 2013)

Metodika je často používaná forma komunikace. Nezáleží na jejím názvu (Pravidla, Příručka, Návod, Pokyny, Postupy, Směrnice, Postupy, …), záleží na smyslu, obsahu a struktuře. V zásadě jde o popis postupu nebo souboru postupů, kterými lze dosáhnout stanoveného výstupu či cíle. Jde o nástroj, který napomáhá k dosažení cíle. Vede k dosažení cíle, pokud jsou postupy dobré a správně aplikované (Navrátilová, 2013).

Základní charakteristiky metodiky jsou dle Navrátilové (2013):

- Aplikovatelnost: Metodiku je možné použít (tj. povede k dosažení cíle) ve variabilním prostředí právě při splnění vstupních podmínek a omezení metodiky.
- Interaktivnost: Metodika je interaktivní nástroj, který je možné a nutné měnit dle vývoje v praxi.
- Přenositelnost: Metodika je využitelná v organizacích s obdobnou klientelou a záměry s přihlédnutím k odlišnému zázemí (jiný rozpočet, struktura, materiální a personální kapacita). Tedy, při změně jednoho či více postupů dosáhneme stejného, nebo podobného cíle.

Vymezení nejčastějších chyb (tedy identifikace toho, co již není metodika), je možné určit tak, že metodikou není dokument s nejméně jedním níže uvedeným atributem:

- Chybí mu některý ze základních prvků, nebo vlastností metodiky,
- dosahuje příliš vysoké míry obecnosti,
- dosahuje přílišné míry složitosti (postupy jsou neproveditelné),
- sdělení jsou nesrozumitelná, nepřehledná, nebo v nich chybí návaznosti,
- jde o formát projektové či technické dokumentace (jednorázový záměr),

- obsahuje velké množství dat, dlouhodobých procesů, různých odborných a hodnotících závěrů.

Jiná definice uvádí, že:

„Metodika je obecně pracovní postup (metoda) nebo nauka o metodě. Metodika v oblasti metodologie vědy je metoda vědecké práce. Metodika v oblasti pedagogiky je nauka o metodě vyučování v určitém oboru (teorie vyučování).

Ve vývoji software metodika představuje souhrn doporučených praktik a postupů, pokrývajících celý životní cyklus vytvářené aplikace. V tomto oboru velmi často dochází – kvůli nesprávnému překladu z angličtiny – k záměně pojmu metodika za metodologie. Pro řešení dílčích problémů mohou být v rámci nasazení metodiky uplatněny specifické postupy – metody.“ (Martinů a Čemák, 2018). S touto definicí se shodují i většiny dalších výkladů, které lze najít napříč literaturou, nebo legislativními dokumenty.

Zvláštní pozornost je třeba věnovat definici metodiky tak, jak ji chápe současná legislativa: „*Výsledek „Metodika“ je souhrnem doporučených praktik a postupů schválených, certifikovaných nebo akreditovaných) kompetenčně příslušným orgánem veřejné správy nebo, pokud kompetenčně příslušný orgán neexistuje, autorizovaným certifikačním (akreditačním) subjektem, provádějícím certifikaci (akreditaci) na základě mezinárodních smluv, norem či obdobných dokumentů s jednoznačně vymezenými a zveřejněnými kompetencemi pro konkrétní oblasti, obory či odvětví a s jednoznačně vymezenými uživateli tak, aby tito uživatelé měli jistotu, že při jejím dodržení budou získané výsledky průkazné, opakovatelné a že se jich lze dovolat. Výsledek „Metodika“ realizoval původní výsledky výzkumu a vývoje, které byly uskutečněny autorem nebo týmem, jehož byl autor členem.*“ (Definice výsledků, 2017). Toto úzké vymezení pojmu metodika slouží výhradně pro potřeby hodnocení a financování VaVaI a jde o zásadní legislativní změnu, ke které došlo během řešení vlastní dizertační práce.

Pro další práci je tak klíčové zmínit, že v rámci hodnocení výsledků výzkumu a vývoje je možné pracovat s vágnější definicí slova Metodika. Pokud však metodika nesplňuje podmínky certifikované metodiky N_{cert}, je nutné ji, v rámci kategorizace výzkumu a vývoje, zpravidla, charakterizovat značkou O – ostatní výstupy. (Definice výsledků, 2017)

11.2 Účel metodiky

Účely metodiky se překrývají s jejími silnými stránkami (benefity). Tyto benefity je možné nalézt v různých zdrojích (Navrátilová, 2013), (Martinů, 2018), přičemž dále je uvedena syntéza těchto tezí. Metodiky tak přináší níže uvedené přínosy (tj. plní níže uvedený účel):

- **Zaznamenání postupu** umožňuje vytvořit posloupnost činností nutných k naplnění požadovaných cílů. Specifikovaná sekvence je definována pro opakující se činnosti,

které jsou prováděny v souladu s metodikou a mohou být prováděny různými spolupracovníky. Pokud se na jedné činnosti může střídat více pracovníků, je možné, prostřednictvím unifikace procesu (tj. použití metodiky) zajistit jejich návaznost v rámci tohoto procesu.
Př: Pokud má výměna kola u automobilu sedm pracovních úkonů (od povolení šroubů, zvednutí vozidla, přes sundání kola, vyvážení kola, atd.), může libovolný pracovník nahradit jiného pracovníka během procesu výměny jednoho konkrétního kola.

- **Uchování postupu** umožňuje zajistit stabilitu postupu (resp. metodiky) v prostředí, kdy se mohou měnit zaměstnanci. Praktickým důsledkem je potřeba konkrétního zaškolení zaměstnance přesně v intencích konkrétní metodiky.
  Př: *Pokud přijde do autoservisu nový pomocník, proto, aby byl schopen výměny kola, stačí ho seznámit právě s touto jednou metodikou. Po proškolení v této jedné konkrétní oblasti se stává kvalifikovaným pro tuto jednu konkrétní činnost.*
- **Zajištění úrovně kvality** je realizováno zejména vzhledem k pracovníkům/týmům pracujícím na různém místě či v různém čase. Zajištění kvality je třeba odlišit od jakosti díla – tím se práce, ani samotná obecná metodika nezabývá. Kontrola kvality je důležitá zejména v případě hledání chyby v díle. Metodika musí být vždy navržena tak, aby dílo/činnost, vzniklé na základě této metodiky bylo kvalitní (avšak bez nároku na úroveň jakosti). Dodržení metodického postupu vede ke kvalitnímu výsledku.
  Př: Pokud je dodržen postup výměny kola v servisu, je tato výměna vždy kvalitní. Pokud postup metodiky není dodržen (např. nedojde k vyvážení kol a automobil má tendenci při jízdě vyšší rychlostí vibrovat), může dojít k nekvalitnímu provedení. Druhým příkladem může být zajištění úrovně kvality ve vztahu k zadavateli činnosti. Pokud jsou po výměně disku kola, místo pěti šroubů fixovány pouze dva, zřejmě došlo k nekvalitnímu odvedení práce.
- **Zajištění očekávání** umožňuje vytvořit vazbu mezi zadavatelem činnosti a jejím realizátorem. V situaci, kdy je známa metodika, její vstupy i výstupy, vzniká předpoklad očekávání na straně zadavatele a možnost ověřitelnosti tohoto očekávání.
  Př: Očekávání při výměně kompletů kol jsou taková, že je předán automobil s kompletem A a navrácen s kompletem B, přičemž ostatní stav vozidla se nezmění.
- **Hodnocení postupů a jejich úspěšnosti** je specifický přínos, který umožňuje hodnotit konkrétní míru přínosu jednotlivců nebo týmů v kontextu celé metodiky, nebo jednotlivých metodických postupů. Díky zavedení metodiky, kdy je postup unifikován, je možné vyhodnotit jednotlivce či skupiny relativně vůči jiným jednotlivcům či skupinám.
  Př: Při výměně kola je možné srovnat nejen dva různé pneuservisy mezi sebou, ale také jednotlivé zaměstnance z jednoho servisu. Dokonce je možné toto srovnání provádět nikoli pro celý proces výměny kola, ale pouze pro konkrétní činnosti.

- **Snížení nákladů** umožňuje metodika tím, že postupy zavádí, unifikuje a jednoznačně určuje jejich návaznosti. Dochází tak k unifikaci z pohledu managementu a odstranění chybovosti (tím, že každý má jasně danou roli či odpovědnost). Jde o přirozený důsledek přínosu zachování postupu a zaznamenání postupu.
- **Optimalizovatelnost** znamená, že postupy je možné hodnotit a agilně upravovat pro potřeby metodiky například na základě zkušeností z praxe. Tato vlastnost plyne z pravidla interaktivnosti. Důsledkem tohoto přínosu je možnost neustále metodiku, resp. její jednotlivé metodické postupy zlepšovat.

11.3 Kategorizace metodik

V textu je vycházeno z definic určených v rámci materiálu Metodika hodnocení výsledků výzkumných organizací a hodnocení výsledků ukončených programů (Metodika hodnocení, 2013). Tento materiál byl schválen usnesením vlády č. 475 ze dne 19. 6. 2013. Zohlednění tohoto dokumentu je validní i v roce 2017, přestože byl pro potřeby hodnocení výzkumných organizací a hodnocení programů účelové podpory výzkumu, vývoje a inovací usnesením vlády č. 107 z 8. února 2017 schválen nový dokument nazvaný Metodika hodnocení výzkumných organizací a hodnocení programů účelové podpory výzkumu, vývoje a inovací (Metodika hodnocení, 2017). Tento nový dokument říká, že:

„*V souvislosti s M17+ budou aktualizovány definice druhů výsledků, kritéria jejich ověřitelnosti a způsob zadávání údajů do IS VaVaI pro potřeby evidence VaVaI na národní úrovni. Aktualizace definic druhů výsledků a jejich zavedení musí být provedeno s ohledem na kontinuitu se stávajícími definicemi a s ohledem na zamezení retroaktivních dopadů. Aby byla umožněna aktualizace bez potřeby otevírání výchozího dokumentu, bude tato problematika zpracována formou samostatného materiálu schvalovaného vládou. Do schválení tohoto materiálu zůstávají v platnosti definice dle Metodiky 2013–2016.*“ (Metodika hodnocení, 2017)

Metodiky však vznikají i v jiném rámci (s jinou motivací, jiným záměrem), než je pouze rámec institucionálních podpor a čerpání dotací. Pro úplnost je níže uvedena další kategorizace metodik:

1. Elementární a prostou formou metodiky je **návod**, který může mít rozdílné formy (textový popis, obrázkový popis), nebo manuál (nebo jeho část), pokud obsahuje metodické postupy.
2. Běžně užívanou formou metodiky je **volná metodika**. Jde o široce známou metodiku, resp. komplet metodických postupů, které jsou v rámci vybrané komunity obecně považované za správné. Tento typ metodiky je charakteristický tím, že pro různé aplikační obory se, pro stejnou kategorii činnosti, značně liší. Do této skupiny metodiky je možné zařadit například Metodiku vědecké práce.

3. Metodické postupy, které byly ověřeny v praxi, tvoří další kategorii, která je nazývána **Ověřená metodika**. Přestože dle klasifikace Metodiky hodnocení (2013) i Definice výsledků (2017) jde stále o výsledek nižší kvality (zařazený do výstupů typu O – Ostatní), pro aplikační sféru, resp. příslušnou aplikační agenturu (například TAČR) je tento výsledek přijatelný, neboť již nejde o čistě teoretický výstup, nýbrž výstup, který má blízkou vazbu na praxi. Vzhledem k tomu, že kategorie není nikde pevně legislativně ukotvena, není postup Ověření jednoznačný. Obecně je využíván přípis aplikačního garanta/partnera z praxe. Vzhledem k legislativnímu neukotvení se může odlišovat i vlastní název výsledku, například CTTZ Univerzity Pardubice (2015) používá název *Metodika s ověřenými vlastnostmi* (CTTZ Univerzita Pardubice, 2015).
4. **Certifikovaná metodika** označovaná zkratkou N_{cert} (Metodika hodnocení, 2013), kterou legislativa uznává pro projekty započaté do 1.1.2018. Tento typ výsledku je předmětem řešení této dizertační práce. Právě proto, že je předmětem vlastní práce, je jí věnována zvláštní pozornost, proto je více detailů uvedeno v samostatné kapitole 11.4.
5. **Metodika schválená příslušným orgánem státní správy**, do jehož kompetence problematika spadá je jedním z nových výstupů VaVaI dle Definice výsledků (2017). Označuje se N_{metS} a platí pro ni specifická podmínka daná předpisem: „*Podmínkou je udělení mezinárodně uznávané certifikace (akreditace) u příslušného odborného certifikačního (akreditačního) orgánu nebo osvědčení příslušného orgánu veřejné správy, který je věcně odpovědný za oblast, ve které jsou metodika nebo postup uplatňovány. V případě, kdy schvaluje, resp. certifikaci (akreditaci) uděluje věcně příslušný orgán veřejné správy, tj. i poskytovatel, musí být takové schválení/certifikace/akreditace uděleno na základě vypracování dvou nezávislých oponentních posudků. Schvalovací/certifikační/akreditační postup může být upraven samostatným předpisem příslušného schvalujícího, resp. certifikačního (akreditačního) orgánu.*“ (Definice výsledků, 2017).
6. **Metodika certifikovaná oprávněným orgánem** je druhý z nových výstupů VaVaI dle Definice výsledků (2017). Označuje se N_{metC}. Platí pro něj stejná podmínka, jako u N_{metS}. Od předchozí se liší především v tom, že certifikace probíhá v instituci jiné, nežli je organizační složka státu, nebo jiná státní instituce. Certifikaci mohou provádět i další osoby splňující podmínky (obecně odborné certifikační orgány).
7. Metodika a postupy akreditované oprávněným orgánem patří mezi nejměkčí formu nového způsobu posuzování metodik dle Definice výsledků (2017). Faktickým obsahem a požadavky se blíží, dříve nedefinované, Ověřené metodice. Označuje se N_{metA} a obsahuje stejnou podmínku udělení, jako předchozí dva případy. V tomto případě je však snížen požadavek na rozsah – kromě metodiky je způsobilý též postup. Zároveň je snížen požadavek na rozsah akceptace kompetenčně příslušným orgánem a místo certifikaci umožňuje realizaci akreditace (prakticky jde o prohlášení o ověření).

11.4 Certifikovaná metodika

Certifikovaná metodika v této kapitole reflektuje stav legislativy v době řešení práce, data vychází z legislativních dokumentů, které nejsou v okamžiku publikace práce platné, nebo na základě Definice výsledků (2017) došlo k 1. 1. 2018 ke změně. Protože je však dizertační práce řešena již od roku 2013, nebyly změny předpisů reflektovány. Fakt, že nedošlo k reflektování těchto změn, má svůj důvod především v tom, že v odborné komunitě s novou strukturou neexistují zkušenosti, precedenty ani prováděcí předpisy. Toto rozhodnutí autora je v souladu s legislativou, neboť výsledky jsou hodnoceny podle metodiky, která je platná v okamžiku začátku platnosti řešení – což v tomto konkrétním případě je metodika 2013 – 2016.

Z výše uvedeného tak plyne, že i v současné době jsou platné definice výsledků, tak jak je chápe Metodika hodnocení (2013). Tato definice potom uvádí, že uznávaným výstupem v rámci VaV je Certifikovaná metodika, značena jako N_{met}, která je definována následovně:

„Výsledek „Certifikovaná metodika“ realizoval původní výsledky výzkumu a vývoje, které byly uskutečněny autorem nebo týmem, jehož byl autor členem. Jedná se o výsledek, kdy autor výsledku vypracuje metodiku (nutnou podmínkou je novost postupů), která byla příslušným orgánem státní správy nebo příslušným odborným certifikačním (akreditačním) orgánem schválena a doporučena pro využití v praxi.“ (Metodika hodnocení, 2013)

Přičemž kromě vlastní definice je dále třeba zohlednit upozornění uvedené v tomtéž dokumentu:

„*Podmínkou je udělení mezinárodně uznávané certifikace (akreditace) u příslušného odborného certifikačního (akreditačního) orgánu nebo osvědčení příslušného odborného orgánu státní správy, který je věcně odpovědný za oblast, ve které jsou metodika nebo postup uplatňovány. V případě kdy certifikaci uděluje věcně příslušný odborný orgán státní správy, tj. i poskytovatel, musí být taková certifikace udělena na základě vypracování dvou nezávislých oponentních posudků. Certifikační postup bude upraven samostatným předpisem.*“ (Metodika hodnocení, 2013)

Dokument, na který je odkazováno, je v agendě jednotlivých ministerstev a specifikuje způsoby a postupy, jakými dosáhnout výstupu N_{met}.

Pro definici certifikované metodiky je důležité také doporučení Rady pro výzkum, vývoj a inovace k certifikaci metodik z 23. ledna 2014:

„Pro výsledek certifikovaná metodika platí:

- *Každý, kdo užije certifikovanou metodiku, by měl mít jistotu, že při jejím dodržení budou získané výsledky průkazné a opakovatelné. Z této základní charakteristiky certifikované metodiky mj. vyplývá, že řadu postupů nelze certifikovat (certifikovat*

tedy lze jen takové postupy, které jsou státem a jeho orgány popř. mezinárodními úmluvami regulovány nebo doporučovány).

- *Gestor za oblast bezpečnostního výzkumu, kterým je Ministerstvo vnitra, je povinen předávat výsledky výzkumu a vývoje do Rejstříku informací o výsledcích (RIV) Informačního systému výzkumu, experimentálního vývoje a inovací. Tato povinnost je uložena zákonem č. 130/2002 Sb., o podpoře výzkumu, experimentálního vývoje a inovací z veřejných prostředků a o změně některých souvisejících zákonů (zákon o podpoře výzkumu, experimentálního vývoje a inovací), a nařízením vlády 2 č. 397/2009 Sb., o informačním systému výzkumu, experimentálního vývoje a inovací.*
- *Metodiky, které nelze předložit k certifikaci příslušnému odbornému certifikačnímu orgánu, neboť takový orgán pro danou oblast, v níž má být metodika uplatňována, neexistuje, budou předloženy k osvědčení příslušnému orgánu státní správy, který je věcně odpovědný za oblast, ve které je metodika uplatňována.“* (Doporučení RVVI, 2014)

11.5 Náležitosti metodiky a její struktura

Požadavky na metodiky jsou v současné době unifikované výše zmíněnou legislativou a jejími prováděcími vyhláškami. Úvodní list musí obsahovat údaje o projektu, interní číslo projektu, jméno řešitele, typ výsledku, pojmenování metodiky, jména autorů a datum vydání aktuální verze metodiky.

Vlastní obsah metodiky musí obsahovat především:

- knihovní údaje,
- popis metodiky, její vymezení, ohraničení a určení předpokladů pro aplikaci metodiky,
- vymezení odborné terminologie,
- popis vlastní práce s metodikou,
- popis vlastních činností realizovaných metodikou,
- vymezení novosti postupů v rámci metodiky,
- odkazy na externí literaturu.

11.6 Proces certifikace metodiky

Certifikace metodiky je proces, při kterém je metodika uznána vhodným odborným orgánem/institucí v oboru, pro které byla metodika certifikována. Certifikovaná metodika splňuje požadavky na strukturu, obsah, rámec, novost a unikátnost. Je použitelná v rámci aplikovaného vývoje či experimentálního výzkumu, jednotlivé kroky jsou ověřitelné,

a uživatel certifikované metodiky tak může dodržovat příslušná pravidla systémů pro sledování jakosti.

Certifikace metodiky provádí vždy úřad (resp. jeho odbor) odpovědný za daný úsek (lidské) činnosti. Součástí procesu certifikace je začátek řízení, příprava několika (zpravidla nejméně dvou) nezávislých posudků (zajišťuje příslušný odbor) a revize formálních a obsahových kritérií. Při splnění všech požadavků a kladných oponentských posudcích může být metodika certifikována.

12 Postup tvorby původní metodiky

Vlastní konkrétní postup pro tvorbu metodiky, to jest jeho nosné části, a vlastního vědeckého úkolu, který byl realizován v souladu se zadáním dizertační práce, je uveden v následujícím textu. Před vlastní tvorbou metodiky byly vymezeny základní předpoklady, ze kterých bylo vycházeno již na počátku vlastní tvůrčí činnosti (při tvorbě metodiky).

Proto, aby bylo možné metodiku realizovat bylo nutné specifikovat několik konkrétních kritérií. Další podkapitoly řeší detailně jednotlivá z kritérií, kterými jsou:

1. Forma výstupu výzkumně-vývojové činnosti
2. Základní technické předpoklady
3. Stanovení strukturální stavby metodiky
4. Návrh konkrétní struktury metodiky
5. Vypracování obsahu metodiky

12.1 Forma výstupu výzkumně-vývojové činnosti

Jasná specifikace formy výstupu je prvním krokem pro jeho faktickou realizaci. Přestože za standardní situace nemusí být určení vhodného typu výstupu výzkumně-vývojové činnosti snadné, pro potřeby práce, jejímž cílem je tvorba Metodiky, je stanovení cíle velmi přirozené. Výstupem musí být metodika.

Při detailnějším seznámení s kategorizací metodik (kapitola 11) bylo důležité rozhodnutí, kterou formu metodiky zvolit. Níže jsou uvedeny jednotlivé formy, včetně krátkého pojednání o konkrétní vazbě na typ výsledku a tuto dizertační práci.

- **Nestrukturovaná metodika** (např. návod, technický postup) by byl jako výstup dizertační práce nevhodný. Tím, že forma výstupu není pevně strukturálně, ani legislativně ukotvena, nedovoluje snadné ověření, zda byl výstup splněn. Tento typ výstupu je obecně používán spíše pro komunikaci, která nemá charakter vysoce odborné výměny informací, resp. nebývá prostředkem pro prezentaci originálních postupů spojených s VaVaI. Často jde o prostředek, který se využívá pro komunikaci například mezi výrobcem a spotřebitelem.
- **Obecná metodika** má jasně specifikovanou formu a je tak snáze uchopitelným výstupem, nežli výstup předchozí. Využití tohoto výstupu je spíše v oblasti implementační (např. interní metodiky budované pro konkrétní organizace v rámci business modelování), nebo vysoce teoretické (např. souhrnné a obecné metodiky). Tento typ výstupu již lze považovat za výstup, který je použitelný v oblasti VaV. Vzhledem k tomu, že dizertační práce vzniká na poli technickém, ke konkrétnímu účelu, bylo by vhodnější zvolit silnější formu výstupu.
- **Ověřená metodika** je obecná metodika, která je fakticky ověřitelná a která byla ověřena. Možnost ověřitelnosti je možné naplnit tím, že metodika obsahuje konkrétní

postupy jako prostředky pro dosažení cílů, přičemž dosažení libovolného cíle je z krátkodobého hlediska ověřitelné. Pokud je takováto ověřitelná metodika ověřena na konkrétním projektu, ve kterém je dosaženo cíle (pro který byla metodika navržena) za použití metodiky, je možné metodiku klasifikovat jako ověřenou.

- **Certifikovaná metodika** je nejkvalitnějším výstupem z kategorie metodik. Vyšší předpokládané kvality je dosaženo tím, že pro klasifikaci je nutné zajistit nejméně dva oponentské posudky (které mají jasně stanovená pravidla) a certifikaci (resp. atestaci) od příslušného orgánu či organizace. Při posuzování jsou kladeny vysoké nároky nejen na odbornou a obsahovou stránku, ale také na splnění požadovaných kritérií, formy, formátu, vyplnění veškerých údajů atp. Samotný proces certifikace je finančně i časově náročný. K certifikaci se předkládá metodika *Metodis* v podobě, v jaké je formalizována v příloze této dizertační práce, to znamená již označená jako Certifikovaná metodika.

Výstupem dizertační práce tak je metodika, která je ve formě, která umožňuje okamžité podání k certifikační autoritě[26]. Jde tedy o výstup ekvivalentní s certifikovanou metodikou. Bez příslušné certifikace je však, v čase a místě obhajoby práce, vlastní výstup na stupni „pouze“ ověřené metodiky.

12.2 Základní technické předpoklady

Po stanovení základní formy výstupu (obecně stanovení toho jakou formou prezentovat výstup) bylo nezbytné odpovědět na otázku, jaké konkrétní nové metody a postupy jsou obsahovou náplní metodiky (je tedy třeba říci co je nosnou myšlenkou metodiky). Základní technické předpoklady pro tvorbu metodiky vychází z nejlepší dostupné praxe. Některé postupy byly přebrány z již existujících metodik. Tyto přebrané postupy byly adaptovány pro konkrétní potřeby *Metodis*, nebo zobecněny, pro možnost širšího využití, než původní záměr předpokládal. Základními technickými předpoklady, se kterými bylo nutné v rámci *Metodis* seznámit čtenáře byly následující:

12.2.1 Řešení na úrovni federace

Metodika se v první fázi věnuje **řešení na úrovni federace**, tedy koncepčně z pohledu celého distribuovaného simulačního modelu. Nejdříve metodika musí oborově zvládnout příslušný typ problematiky, popis způsobu analýzy a dekompozice. Analýza a její metoda je klíčová a musí zahrnovat takové postupy, které umožní využívat širokou škálu moderních metod pro tvorbu softwaru, avšak zároveň musí být natolik robustní, aby splňovala příslušné ISO a případné IEEE předpisy pro komerční software. Konkrétně tedy musí být zvoleny

[26] Autor dizertační práce působil v letech 2014 – 2019 jako externí oponent pro TAČR. Součástí zpracování oponentských posudků byly také oponentské zprávy pro Certifikované metodiky pro další orgány veřejné správy. Údaje jsou tak podpořeny vlastními zkušenostmi autora z praxe.

takové analytické metody, aby umožnily nasazení standardních i agilních metod vývoje při zachování optimálních postupů z hlediska vybrané aplikační domény.

Obdobou tvorby architektonického modelu pro modelované systémy je pro distribuované simulátory volba vhodné **dekompoziční metody** a její aplikace. Proto, aby byla metodika dostatečně robustní, by se měla vyjádřit k širší škále metod, jimiž je možné dekomponovat simulátor. Může se totiž stát hlavní přidanou hodnotou práce; současné metody i metodiky se prakticky výhradně věnují pouze jedné třídě dekompozičních metod. Originální je přístup, kdy SW architekt není závislý na konkrétní metodě provedené dekompozice. Celý navržený princip je relativně složitý a jeho úspěšné zavedení je jedním z potenciálních oborových přínosů této práce.

Po dekompozici je třeba podívat se na celou problematiku z hlediska použité technologie, přičemž se předpokládá aplikace HLA. Z toho důvodu je nezbytné zaměřit se na **komunikační protokoly**, resp. jejich předpis v podobě OMT. Ideální složení je však postupovat od SOM, které je nezbytné definovat velice obšírně a kompletně tak, aby byla celá evoluce simulačního výpočtu sledovatelná přímo z kontrolního rozhraní RTI. Druhá složka OMT (tedy FOM), by měla být definována až s přihlédnutím k nefunkčním požadavkům, které vzešly z analýzy. Problematice návrhu FOM musí být věnována zvláštní pozornost, jak ukazují zkušenosti z dalších pracovišť[27].

Logicky následuje **kategorizace jednotlivých federátů** v rámci federace a jejich fungování, resp. dopadů jejich fungování na federaci. Zároveň metodika navrhuje různé **synchronizační metody** pro různé typy federátů (resp. pro federace složené z různých tříd federátů).

12.2.2 Řešení na úrovni federátů

Řešení na úrovni federátů, resp. jednotlivých sekvenčních simulátorů je pro metodiku klíčové především proto, že nejde jednoduše definovat implicitní přístupy. Metodika se tak musí k celé problematice stavět důsledně, a to jednak vzhledem k architektuře jednotlivých tříd federátů a jednak vzhledem k dekompoziční metodě, jejímž výsledkem federát je. Kombinací dvou výše uvedených faktorů je určen **charakter federátu**.

Jednotlivé federáty mohou být například logické procesy, či pouhé fragmenty logických procesů. V extrémním případě může být federátem i pouhý fragment o velikosti jedné funkce. Metodika musí být vystavěna tak, aby na jejím základě bylo možné stavět federáty s libovolně dekomponovanými logickými procesy.

[27] Například armádní pracoviště v Pardubicích propojené s armádním pracovištěm v Norsku, pro tvorbu FOM při nácviku bojových situací prostřednictvím interaktivních simulací, využívá pouze tři různé stavové proměnné (z důvodu rizik spojených s latencí) a to přestože má armáda vyhrazenu vlastní přenosovou síť, kde jsou rizika vysoké latence minimální.

Kromě charakteru federátu je zohledněna druhá klíčová vlastnost, tj. **architektura federátu**. Ta je přirozeně závislá na metodice, kterou byl federát analyticky navržen, avšak vždy existuje určitá volnost. Zároveň platí, že pro většinu dekompozičních metod je uplatnitelná širší škála architektonických metod pro návrh federátů.

Metodika se také zaměřuje na standardní synchronizační metody obsahující kalendáře událostí, avšak aby metodika mohla být navržena udržitelně do budoucna, je třeba zároveň navrhnout taková řešení, která nejsou postavena na simulačním jádře v jeho standardním pojetí. Demonstrátor pro ověření metodiky z tohoto pohledu aplikuje systém s **vlastní interní synchronizací**.

12.2.3 Rozhraní federát – federace

Po definici prostředí federace i prostředí federátu se metodika musí věnovat rozhraní mezi těmito dvěma softwarovými vrstvami/rozhraními. Korektní napojení logiky, komunikačních schémat a distribuované architektury (tedy složek federace) je třeba implementovat v jednotlivých instancích federátů. Tato část metodiky zahrnuje celou problematiku z nejnižší úrovně a věnuje se především komparaci jednotlivých alternativních přístupů a řešení.

V poslední části metodiky, se věnuje pozornost řešením zaměřeným na provoz simulace. Tato část metodiky se zaměřuje na logické a technické náležitosti nutné k řádnému propojení předchozích dvou vrstev.

12.3 Stanovení strukturální stavby metodiky

Proto, aby bylo možné uplatnit výše uvedené principy, je nutné především důsledně dbát na zajištění kompatibility metodiky s ostatními normami a standardy. Z výše uvedeného plyne, že součástí strukturální stavby metodiky musí být jednoznačné vyjádření o vztahu metodiky k ostatním standardům.

Při vlastní realizaci metodiky je třeba především pochopit účel a smysl metodiky jako takové. Metodiku je třeba budovat tak, aby plnila svůj primární účel – dokumentovatelným způsobem ujednotit vývojové procesy tak, aby bylo možné zajistit maximální odolnost proti pochybení, zajistit úspory při realizaci a využít dalších výhod plynoucích z unifikovaného vývoje.

Protože vlastní metodika se snaží zachytit celý proces vývoje simulátoru, bylo nutné celý, takto rozsáhlý proces, vhodně etapizovat. Pro strukturální stavbu obsahové části metodiky (tj. té části, která se věnuje vlastnímu metodickému postupu) byla vybrána forma čtyř hierarchických úrovní etapizace. Jednotlivé úrovně lze popsat následujícím způsobem:

- **Metodický postup** zahrne veškeré činnosti, které jsou nutné k implementaci distribuovaného simulátoru odrážejícího provoz vybraného systému

s decentralizovaným řízením. Vstupem pro tento metodický postup je znalost simulovaného systému, řešitelský tým s poučeným (v oblasti simulace) vedoucím týmu a vlastní metodika. Výstupem celého metodického postupu je hotový, ověřený a předaný simulátor.

- **Fáze** jsou takové dílčí celky metodického postupu, které jsou charakteristické tím, že je možné určit požadované vstupy, očekávané výstupy a procesy, jejichž prostřednictvím dojde k transformaci vstupů na výstupy. Zároveň je fáze charakteristická tím, že logicky sdružuje obdobné činnosti. Fáze by měly být nejmenší samonosnou etapou práce.
- **Bloky** jsou takové části fáze, pro které platí, že obsahově naplňují fáze. Všechny fáze jsou členěny do stejných typů bloků v totožném pořadí. Bloky mají usnadnit orientaci ve fázích. Z výše uvedeného lze logicky dovodit, že přirozenými kandidáty na vlastní blok jsou vstupy, očekávané výstupy, procesy, atp.
- **Kroky** jsou dále granulované bloky. Krokování je provedeno pro bloky, které jsou samy svým rozsahem netriviální. Účelem dalšího rozpadu bloků na kroky je zvýšení přehlednosti konkrétního bloku.

Výše uvedená struktura je variací jedné z osvědčených forem (Smith a Apple, 2014), jak metodiku tvořit. Při tvorbě *Metodis* se ukázala, jako vhodná. Konkrétní struktura je uvedena v příloze (vlastní metodice).

12.4 Návrh konkrétní struktury metodiky

Po přípravě obecného rámce, jak strukturovat metodiku, bylo možné přistoupit k praktické realizaci vlastní metodiky. V rámci textu je kladen důraz především na vlastní autorskou tvorbu obsahu metodiky. *Metodis* obsahuje i povinné části dané legislativou, které však nemají oborový přínos, a proto nejsou dále diskutovány.

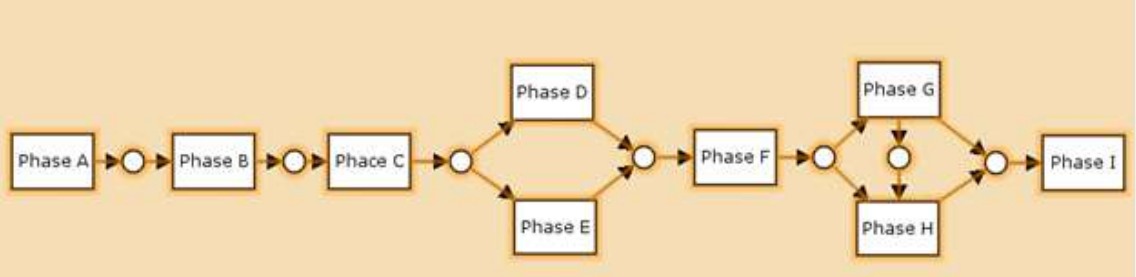
12.4.1 Návrh konkrétních fází metodiky

Metodický postup, který metodika zachycuje, byl rozdělen na níže uvedené fáze:

- Fáze A – Přípravná fáze
- Fáze B – Analýza vstupních dat
- Fáze C – Architektura
- Fáze D – Implementace lokálních částí
- Fáze E – Implementace globální části
- Fáze F – Propojení lokální a globální části
- Fáze G – Verifikace
- Fáze H – Validace
- Fáze I – Předání

Na každou fázi je možné nahlížet jako na samostatnou etapu zpracování. Některé fáze (fáze B a fáze C) je však možné implementovat jen v případě znalosti celé metodiky, neboť klíčové postupy k jednotlivým sub-variantám implementace jsou uvedeny až v blocích, které se takovému tématu věnují. Lze tedy říci, že úspěšný architektonický návrh globálního komunikačního rozhraní je nutné realizovat ve fázi C, avšak je nezbytně nutné přihlédnout k požadavkům fáze E a fází ověřovacích G a H. Fáze I se věnuje ukončení projektu – předání, nebo nasazení.

Struktura metodiky není čistě lineární, ale dochází v ní k možnosti paralelizace fází. Tato paralelizace je uvedena na následujícím schématu, přičemž ve schématu je využito anglické označení fází v souladu s TOGAF (Haren, 2011).



Obrázek 20: Jednotlivé aplikovatelné fáze metodiky[28]

Na obrázku 20 je zobrazena struktura jednotlivých fází metodiky ve formě síťového grafu. Vlastní fáze jsou reprezentovány vrcholy grafu (graficky reprezentované obdélníky), jejichž návaznosti jsou značeny orientovanými hranami. Graf obsahuje také pomocné milníky, které mohou sloužit také jako body synchronizace (tedy vrcholy graficky znázorněné kruhovými objekty). Jednotlivé fáze jsou označeny klíčovým slovem **Phase** a identifikátorem příslušné fáze. Pomocné milníky (označeny kružnicí) jsou klíčové především v případě větvení (paralelizace) fází. V případě změn je nutné vždy pracovat od milníku bezprostředně následujícího fázi změn. Pokud by například na základě problémové implementace (fáze D) muselo dojít k architektonickým změnám (fáze C), je znovu nutné uvažovat realizace všech fází následujících po milníku, který následuje po fázi C. Znovu realizována tak musí být fáze D i fáze E. Toto pravidlo se využije rekurzivně.

Každá fáze je popsána strukturovanou formou (pomocí bloků). Popis začíná blokem **definice cílů** a stručným popisem postupu, následuje blok **popisu vstupů** (tj. předpokladů pro zahájení dané části implementace) a blok **popisu výstupů**. Blok **popis věcné náplně** fáze odpovídá způsobu popisu v TOGAF a při vysoké složitosti je dle potřeby dále členěn na kroky.

[28] Zdroj: (Brožek, 2016).

Je třeba zdůraznit, že tělo metodiky obsahuje nejen metodický postup, ale také odkazy na příklady dobré/běžné praxe pro ty problémy, které nejsou standardně řešeny v souvisejících normách/doporučeních/standardech.

12.4.2 Struktura fází

Každá fáze (neboli dílčí část metodiky) má unifikovanou podobu. V některých fázích mohou některé standardní bloky chybět – to je způsobeno tím, že pro danou fázi neexistuje příslušný výstup. Struktura každé fáze sestává z těchto bloků:

- **Cíle** specifikují vymezení cílů fáze. To jest určení čeho je třeba v rámci dané fáze dosáhnout. Vymezení cílů zároveň nahrazuje jistou anotaci cíle, neboť poučený čtenář je schopen odhadnout, zda cíle fáze odpovídají realizačním požadavkům.
- **Způsob dosažení cíle** obsahuje postupy, které je nutné realizovat pro dosažení příslušného cíle. Jde zpravidla o konkrétní vymezení činností, metod, nebo procesů. Tento blok slouží k vymezení cest pro dosažení cíle. Při vysoké složitosti je blok dále rozdělen do kroků.
- **Vstupy** uvádějí výčet požadovaných informací, dat, nebo dalších dokumentů či materiálů, které budou v rámci fáze zpracovány. Vstupy je možné nahradit jejich substituty.
- **Postup** je robustní částí fáze, jejímž obsahem je vysvětlení konkrétní postupu. Tedy snahou jasně specifikovat činnosti, které je třeba vykonat, v jakém pořadí, za využití jakých vstupů, případně jaké jsou očekávané výstupy. Tento blok je zpravidla dále členěn na kroky.
- **Výstupy** slouží jako základní akceptační kritéria pro vyhodnocení úspěšné realizace fáze. Pokud jsou po realizaci fáze dostupné všechny předepsané výstupy, existuje předpoklad, že metodika byla použita správným způsobem.

Další nepovinné bloky mohou popisovat složitější problémy, které není možné jinam zařadit. Často jde o údaje získávané prostřednictvím rešerší, nebo techniky realizované dobrou praxí (tj. řešení, které jsou používána). Doplňkové bloky nejsou nutnou podmínkou metodiky, avšak zvyšují její kvalitu tím, že doplňují konkrétní text o zkušenosti z aplikační domény/praxe. Tam, kde to bylo možné, byly tyto další bloky zařazeny do příloh metodiky a v textu byly zavedeny odkazy.

12.5 Vypracování obsahu metodiky

Při tvorbě metodiky bylo třeba důsledně respektovat povinné náležitosti. Některé povinné náležitosti nebyly přirozeně použitelné v okamžiku, kdy metodika vznikala jako výstup dizertační práce, nikoli jako výstup samostatného projektu.

Realizace knihovních údajů podléhala údajům platným k době poslední revize metodiky, to jest k létu roku 2017. Klasifikace CZ-NACE i klasifikace oboru řešení byla realizována po konzultaci se zaměstnanci Národní technické knihovny.

Vymezení cílů metodiky, její ohraničení, klasifikace simulátorů, pro které byla metodika určena a předpoklady pro aplikaci metodiky vyplývaly ze zadání dizertační práce a vlastní rešeršní činnosti. Důraz byl kladen na maximální obecnost metodiky.

Vymezení kompatibility metodiky bylo definováno vůči majoritním technologiím, kterými bylo možné metodiku implementovat. Zároveň byla příslušná část zaměřená na zohledněné normy. Součástí metodiky je také slovník odborných pojmů, které metodika využívá.

Prakticky zaměřené úvodní části metodiky, v souladu s požadavky na formální podobu metodiky, uvádí, jak s metodikou pracovat a vytváří doporučení vzhledem k systému řízení řešitelského týmu. Tyto části jsou podmínkou pro to, aby metodika byla uplatnitelná v systémech s požadavky na jakost. Součástí úvodních částí je vymezení zájmů a zainteresovaných osob a především definice jednotlivých zainteresovaných osob.

Po těle metodiky je uvedena povinná část *Novost postupů*, která specifikuje v čem je metodika jedinečná, resp. v čem překonává současná doporučení. Vzhledem k tomu, že na dané téma nebyla, v rámci rešeršního šetření, žádná metodika nalezena, bylo obhájení jedinečnosti a novosti řešení relativně snadné. V závěru metodiky je uvedena literatura a samotná metodika obsahuje jednu přílohu úzce související s ISO/IEC/IEEE 29119.

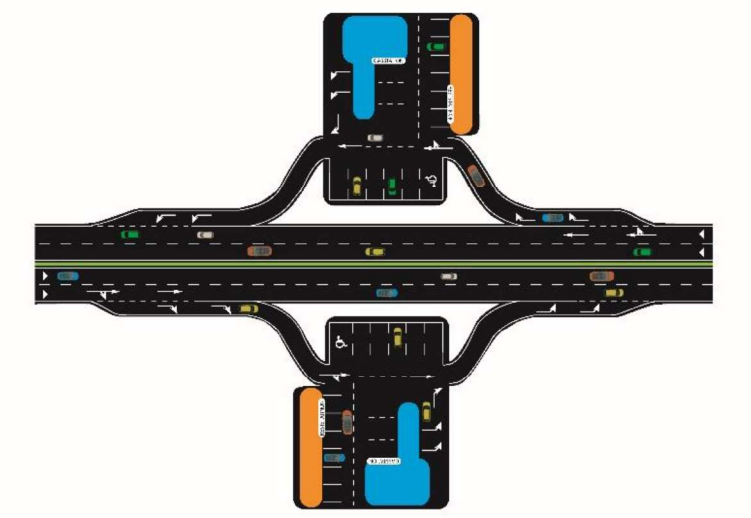
Kompatibilita vytvářené metodiky je klíčová pro udržení standardů při tvorbě procesů realizujících jednotlivé kroky metodiky, nebo procesů s realizací cílů metodiky souvisejících. Tedy, bylo nutné dbát na to, aby současné normy, legislativní opatření a doporučení byly v souladu s nově utvářenou metodikou. Pokud není metodika naprosto převratná, nebo pokud se nevěnuje naprosto inovativnímu tématu, není žádoucí, aby porušovala zásady běžné pro daný obor. Pro zajištění kompatibility metodiky bylo nutné udělat velmi robustní šetření, které umožnilo vytvoření souvislého přehledu o souvisejících normách. V kontextu této práce bylo nutné sledovat především normy týkající se oboru simulace a oboru softwarového inženýrství (návrhu, implementace, testování a dokumentace). Právě proto byla realizována poměrně rozsáhlá rešeršní činnost v této oblasti.

ŘEŠENÍ OVĚŘUJÍCÍ METODIKU

Následující část práce se zaměřuje na představení jednotlivých praktických výsledků, které vznikly během různých fází zpracování dizertační práce. Všechny výsledky úzce souvisí s tématem práce – totiž počítačovou simulací. Představené výsledky jsou různého rozsahu i účelu. Motivace k jejich vzniku mohla být prostá snaha pochopit konkrétní technologii, nebo ověřit fungování technologie, nebo paradigmatu v praxi. Případové studie představené v kapitolách 13 až 19 sloužily jako podpora při tvorbě metodiky. Umožnily čerpat zkušenosti s implementací simulátoru, nebo použití technologií, případně umožnily ověřit konkrétní fáze vznikající metodiky. Zvláštní postavení má případová studie představená v kapitole 20, která vznikla za účelem ověření uplatnitelnosti *Metodis* jako celku.

13 Simulátor jednoduchého dopravního systému – dálnice

Prvním softwarem, který vznikl na základě zaměření výzkumu školícího pracoviště se stal jednoduchý simulátor dopravního provozu na úseku dálnice. Simulovaným systémem je úsek dálnice se zjednodušeným dopravním provozem ilustrovaný na obrázku 21. Úsek dálnice dále obsahuje odbočovací pruhy (směřující k odpočívadlu, čerpací stanici a obchodům) a připojovací pruhy (sloužící k návratu vozidel zpět na dálnici). Entitami jsou jednotlivá vozidla, která na úsek dálnice vstupují. Na základě parametrizace vozidla buď systémem prochází přímo, nebo s využitím sjezdu (a potenciálně s využitím odpočívadla, čerpací stanice, nebo obchodu).



Obrázek 21: Jednoduchý dopravní systém[29]

[29] Zdroj: Vlastní

13.1 Motivace k řešení

Zadání problematiky vzešlo od školitele a vycházelo ze zadání, které řešili v různé technologické variaci další doktorandi školícího pracoviště. Na systému tak byly různými pracovníky zkoumány možnosti nasazení webové simulace, simulace využívající vzdálené volání metod, atp. Konkrétní publikace zabývající se danou problematikou jsou následující: (Voráček a kol., 2014), (Pénzeš a Kavička, 2013), (Hřídel a Karták, 2013), (Karták a Kavička, 2014) a (Karták, 2015). Zajímavý je především model Hřídele a Kartáka (2013), který byl pro konkrétní nastavení vstupních parametrů verifikován a částečně validován numerickými metodami a je využitelný jako etalon pro další simulátory.

Výhodou uvedeného jednoduššího simulátoru bylo především:

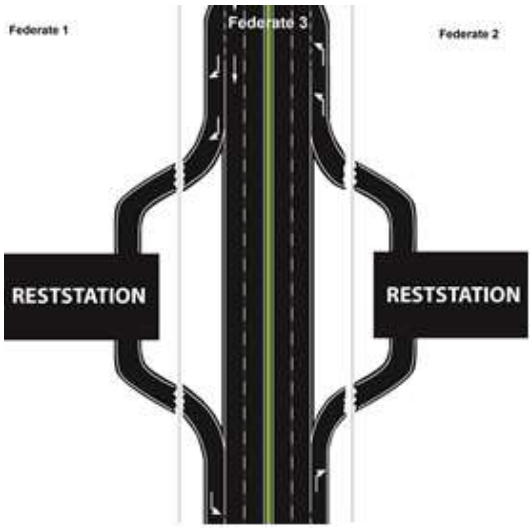
- Model tohoto rozsahu byl vhodný pro prvotní seznámení se s problematikou distribuované simulace.
- Intuitivní popis systému a chování entity v systému podporoval vizuální kontrolu chování simulujícího systému.
- Simulovaný systém bylo možné dekomponovat na dvě heterogenní třídy simulátorů: dálnici a odpočívadlo. Je možné provést různé sestavení distribuovaného simulátoru.
- Systém je svým charakterem systémem s decentralizovaným řízením a tím zapadal do rámce řešení dizertační práce.

13.2 Vlastní řešení

Na základě výše specifikovaného simulovaného systému vznikly celkem tři odlišné samostatné simulující systémy, které se lišily svým účelem.

První simulující systém sloužil k seznámení doktoranda se samotnou architekturou HLA. V době realizace nebylo v ČR nalezeno pracoviště, které se HLA zabývalo. Z toho důvodu vznikl první simulující systém během zahraničního působení doktoranda na Lancaster University Management School pod vedením Stephana Bhakti Ongga. Simulátor splnil svůj účel, neboť došlo k úspěšné realizaci řešení za využití RTI od švédské společnosti Pitch technologies. Simulátor byl implementován v programovacím jazyce Java bez dalších podpůrných knihoven nebo frameworků. Nevýhodou prvního dálničního modelu byla navržená architektura. Simulátor byl homogenní (sestával z jednoho federátu) – jádro simulátoru obsluhovalo veškeré procesy, které v simulátoru probíhaly. Přestože bylo možné sledovat změny stavového prostoru v nástroji Pitch Control Center, nešlo o distribuovaný simulační výpočet v pravém slova smyslu. Řešení nemělo GUI, avšak prokázalo funkčnost vlastního simulačního jádra, stejně jako ověřilo další klíčové funkcionality programovacího jazyka k běhu simulátoru (např. fungování generátorů pseudonáhodných čísel). Pro účely výzkumného směřování měl etapový výsledek smysl a šlo o první simulační model vytvořený pod plnou kontrolou autora (tj. s přístupem ke všem částem zdrojového kódu).

Druhý simulující systém byl navržen jako distribuovaný simulátor se dvěma heterogenními typy simulátorů: Dálnicí a odpočívadlem. Vzhledem k tomu, že přestože je řešení heterogenní, zároveň obsahuje dva homogenní simulátory (odpočívadla), bylo možné ověřit fungování dvou instancí téže třídy simulátoru v rámci jedné distribuované simulace. Způsob, jakým byl simulační model dekomponován, je ilustrován na obrázku 22. Přínosem implementace tohoto simulačního modelu bylo seznámení se s distribuovanou simulací provozovanou prostřednictvím architektury HLA.



Obrázek 22: Dekompozice řešení[30]

Třetí dálniční model, který byl v rámci využití tohoto typu simulovaného systému implementován, vychází co do nastavení procesů i dekompozice simulujícího systému z řešení druhého. Odlišnost obsahuje simulátor jednu, avšak klíčovou. Na rozdíl od druhého simulačního modelu je aplikována „inteligence entit" – to znamená, že entity se stávají, do jisté míry odpovědné za své chování a stávají se svým charakterem aktivními mikroprocesy (vizte kapitolu 5.6). Jednotlivá rozhodnutí o návštěvě odpočívadla, rychlosti jízdy (pokud je to možné) a požadavky na návštěvu odpočívadla jsou řešeny přímo v logice jednotlivých entit. Toto řešení má řadu výhod – především při úpravách nebo rozšiřování modelu. To platí především v okamžiku, kdy jsou do simulátoru přidány nové třídy entit (například tehdy, kdy k současným osobním a nákladním automobilům přibydou například motocykly). Vlastní řešení s inteligentními entitami má výrazný dopad na simulátor a jeho údržbu, a to, přestože jde o vizuálně nezjistitelnou změnu. V kontextu řešené problematiky však byl, pro všechna další šetření, brán jako výchozí dálniční model třetího typu.

[30] Zdroj: (Brožek a Jakeš, 2015)

13.3 Souhrn

Jednotlivé typy simulujících systémů umožnily vlastní experimenty s distribuovanou simulací a charakterem problémů, které se týkají vlastního vývoje distribuovaných simulátorů. Zároveň byla vytvořena příležitost pro zkoumání různých architektonických návrhů simulátorů a odlišných implementačních technologií (vč. HLA).

14 Framework pro tvorbu simulátorů v souladu s HLA

Po prvních experimentech s HLA a pochopení jejích hlavních principů bylo vytvořeno softwarové řešení, které mělo usnadnit implementaci simulátorů této třídy. Tento krok měl za cíl umožnit vývoj HLA simulátorů i dalším pracovníkům na pracovišti a to bez nutnosti dlouhého experimentování s novou technologií.

14.1 Motivace k řešení

Vývoj simulátorů využívajících komerční RTI není triviální. Přestože centrální RTI komponenta poskytuje řadu služeb, je implementace zcela v kompetenci programátora. Součástí vyvíjeného řešení musí být příprava OMT, jednotlivých rozhraní, interní synchronizace federátu, reflexe stavových změn do RTI a implementace řady dalších rozsáhlých obsluh. Proto, aby došlo k usnadnění vývoje simulátorů, byl implementován framework *Hlava* (High Level Architecture Virtual Assistant).

Koncepce měla za cíl zjednodušit a zpřístupnit použití HLA s těmito hlavními cíli:

- Implementovat generické simulační jádro pro federát tak, aby se uživatel frameworku sám nemusel starat o implementaci kauzální synchronizace.
- Umožnit základní operace s federátem (připojení k federaci, registrace objektů, výběr synchronizační metody HLA) jen prostřednictvím volání metod, bez hlubší znalosti fungování HLA.
- Generalizovat problematiku zveřejňování, úpravu a předávání proměnných (těch, které jsou pro stavový prostor definovány v OMT) a jejich hodnot.

Softwarové řešení bylo v době publikace (Brožek a kol., 2014b) originální, což lze doložit i oceněním výsledku na odborné konferenci zařazením do TOP5 příspěvků a nabídka na rozšíření a upřesnění problematiky a její publikaci v odborném časopise. Framework byl používán na školícím pracovišti pro podporu budování distribuovaných simulátorů v rámci bakalářských a diplomových prací do roku 2014. V roce 2014 došlo k narazení řešení prostřednictvím komerčních produků Pitch Visual OMT a Pitch Developer Studio.

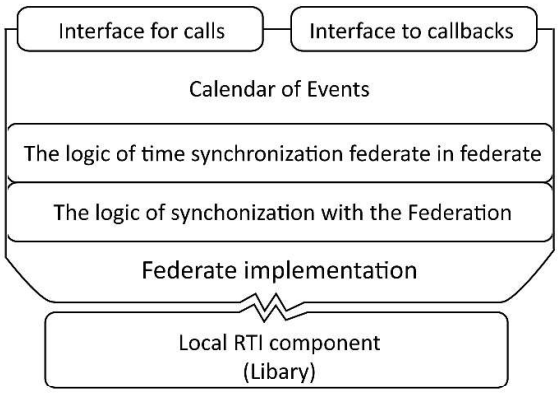
14.2 Vlastní řešení

Implementace vlastního frameworku je provedena v programovacím jazyce Java (resp. JavaFX pro možnost využití ve webové simulaci). Framework se skládá z knihovny vlastního výkonného jádra a předpisu dvou rozhraní. Pro libovolný program, který má do své struktury přilinkovanou knihovnu a implementuje obě rozhraní, platí, že je takový program možné připojit k RTI. Pokud jsou respektována pořadí pro volání metod (od inicializace, přes připojení k federaci, až po synchronizaci vnitřního času a spuštění

simulačního výpočtu), je takový program možné prohlásit za federát fungující v souladu s IEEE1516:2010.

Struktura knihovny, resp. struktura jádra, které knihovna obsahuje, je znázorněna na obrázku 23. Na tomto obrázku je možné vidět strukturu jednotlivých vrstev knihovny, přičemž v horní části jsou vidět dvě, již výše uvedená rozhraní (jde o rozhraní pro volání a o rozhraní představující události přicházející do federátu z HLA). Kromě těchto rozhraní se jádro dále skládá z:

- Local RTI component, tedy lokální RTI komponenta je jedinou externí součástí frameworku. Jde o knihovnu, která je součástí dodávky každého RTI řešení. Rozhraní je pevně dáno standardem, avšak implementace se liší. Při provozu simulátoru na řešení například od společnosti Pitch je nutné využít implementací této komponenty právě od společnosti Pitch. Pokud by mělo dojít ke změně a provoz by byl realizován prostřednictvím komponenty od jiné společnosti (například MÄK), bylo by, pro korektní fungování frameworku, nutné vyměnit knihovnu této komponenty za komponentu, kterou dodává jiná společnost.
- Federate implementation, což je vrstva, která odstiňuje programátora od standardní metody RTI Ambassadora a Federate Ambassadora. Volání zpracovává, a buď přímo mění vnitřní stavový prostor programu, nebo předává data dalším vrstvám.
- Dvou vrstev zajišťujících časovou synchronizaci. Jednak jde o časovou synchronizaci federátu s federací. Parametricky lze při spuštění nastavit, zda bude použita standardní konzervativní, bariérová konzervativní, nebo optimistická metoda synchronizace. Zároveň je možné nastavit, zda je federát pro čas v rámci federace řídícím, řízeným, nebo kombinovaným členem. Synchronizace místního federátu pak zajišťuje kauzalitu uvnitř federátu. Parametricky je možné nastavit například to, zda má federát běžet v režimu real-time nebo maximální rychlostí.
- Calendar of Events, který představuje pro uživatele intuitivní kalendář událostí. Jde o prioritní frontu, do které může přidávat generické události uživatelská aplikace, nebo nižší vrstvy frameworku na základě informací z federace. Bez jakékoli uživatelské akce dojde k synchronizaci události v rámci federátu i federace. O posun lokálního času i vytvoření příslušné události se stará framework – uživatel pouze prostřednictvím rozhraní dostane informaci o tom, že je třeba obsloužit událost (pokud k takové potřebě dojde).



Obrázek 23: Jádro frameworku *Hlava* [31]

14.3 Souhrn

Vlastní framework sloužil především k usnadnění vývoje HLA aplikačních řešení a je využíván v dalších simulátorech, které byly autorem, nebo pod autorovým vedením, realizovány. Hlavní přínos frameworku je v jeho jednoduchosti. Umožňuje používat, relativně složitou architekturu HLA k tvorbě vlastních simulačních modelů jen za využití dvou rozhraní (rozhraní pro volání a rozhraní pro zpětná volání), jejichž implementace není příliš složitá. Navíc řešení umožňuje různé režimy práce s časovou osou (režim reálného času, režim asfastaspossible, striktně řízený režim, serverově řízený režim atp.). Tedy, samotný framework značně usnadňuje vývoj aplikací.

[31] Zdroj: (Brožek a Jakeš, 2014)

15 Simulátor jednoduchého dopravního systému – železnice

Prvním samostatným simulátorem, který byl postaven na frameworku *Hlava* byl interaktivní simulátor zjednodušeného železničního provozu. Simulátor byl naprogramován v programovacím jazyce JavaFX a využíval událostně orientované simulace. Animační jádro není pouhou nadstavbou diskrétního simulátoru, ale aktivní výpočetní součástí celého řešení, neboť grafický stavový prostor je přímou integrální součástí stavového prostoru simulátoru, nikoli pouze jeho vizualizací. Vlastní simulátor byl vybudován jako interaktivní – to znamená, že je možné provádět jeho parametrizaci během samotného simulačního výpočtu.

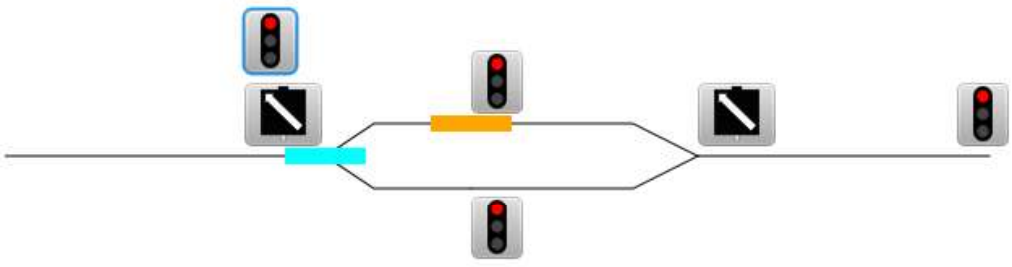
15.1 Motivace k řešení

Vyvinutý simulátor měl za cíl ověřit fungování:

1. Frameworku *Hlava* a dále posloužit k odhadu potřebné doby vývoje, s jakou lze simulátor, za jeho využití, implementovat.
2. Principů interaktivní simulace na běžícím distribuovaném simulátoru.
3. Přístupů sloužících ke snímání uživatelských vstupů/interakcí.

15.2 Vlastní řešení

Pro simulovaný systém byla použita část železniční tratě Přelouč – Prachovice. Tato trať byla rozdělena do několika úseků tak, jak byly řízeny z konkrétních stanic. Každý úsek je pod plnou kontrolou dispečera, který může měnit stav návěstidel. Jednotlivé soupravy se pohybují právě v souladu se stavem návěstidel (stav, kdy dojde k nebezpečné situaci je přípustný – to znamená, že ručně lze návěstidla nastavit i tak, že soupravy začnou v jeden okamžik obsazovat totožný kolejový úsek z opačných směrů). Činnosti vlastních návěstidel neodpovídají zcela drážním předpisům, samotné bezpečnostní prvky jsou zjednodušeny v souladu s cílem řešení simulátoru.



Obrázek 24: Zjednodušená kolejová infrastruktura stanice Choltice[32]

[32] Zdroj: Vlastní

Při vývoji simulátoru nebyl kladen důraz na vysoce hodnověrné grafické zpracování řešení. Ukázka simulátoru (konkrétně úseku se stanicí Choltice) je vizualizována na obrázku 24.

Proto, aby došlo k ověření frameworku *Hlava*, bylo nutné simulátor provozovat v rámci distribuovaného výpočtu. Dekompozice na jednotlivé dílčí federáty je možné vidět v tabulce 2. Celkem byla trať, prostřednictvím metody rozkladu zmíněné v předchozích kapitolách, dekomponována na sedm úseků. Při spuštění simulátoru provozu (nejméně) v rámci dvou sousedních federátů bylo prostřednictvím vizualizace viditelné, že souprava skutečně mezi úseky/federáty cestuje. Tedy, po opuštění infrastruktury tratě spravované jedním federátem vstupuje vlak na sousední úsek tratě spravovaný/řízený druhým federátem.

Tabulka 2: Dekompozice simulace na konkrétní stanice a úseky tratí[33]

Název	Typ	Poznámka
Prachovice	Stanice	Restrikce – mimo nákladní tratě cementárny
Kostelec – sklady paliva	Stanice	
Kostelec – trať	Úsek tratě	
Heřmanův Městec	Stanice	
Choltice	Stanice	Vizualizováno na obrázku 24
Valy	Úsek tratě	Restrikce jen na trať Veselí – Přelouč
Přelouč	Stanice	Restrikce jen na nástupiště tratě Veselí – Přelouč

15.3 Shrnutí

Simulátor měl za cíl ověření synchronizace uživatelského času a různých přístupů snímaní uživatelských interakcí prostřednictvím frameworku *Hlava* a zjištění náročnosti vývoje za využití frameworku. Tohoto cíle bylo dosaženo.

Konkrétní stanice a úseky byly implementovány pod správou vlastních federátů. Vnější logiku řešila RTI (konkrétně Pitch pRTI). Vnitřní logika byla řešena prostřednictvím frameworku *Hlava* v kombinaci s událostně orientovaným programováním animačního engine JavaFX. Celá simulace byla koncipována jako interaktivní – uživatel mohl chování simulátoru ovlivňovat manipulací s návěstidly.

Simulační studie splnila vytyčené cíle a umožnila tvorbu dalších simulátorů založených na frameworku *Hlava*, které však již byly implementovány třetími osobami.

[33] Zdroj: Vlastní

16 Interaktivní simulátor střeleckého souboje

Další nezávislé ověření možnosti aplikovat framework *Hlava* bylo realizováno v rámci interaktivního simulátoru střeleckého souboje, který byl náplní diplomové práce (Jakeš, 2014).

16.1 Motivace

Pro ověření frameworku *Hlava*, jako uvažované klíčové části řešení dizertační práce, bylo přistoupeno k tomu, že původní framework bude použit nezávislou osobou pod vedením autora frameworku. Zadání bylo specifikováno jako simulátor střeleckého souboje více osob realizované prostřednictvím HLA. Takto specifikované zadání umožnilo nejen aplikovat framework, ale zároveň ověřit jeho nasazení na trenažéru, pro nějž platí, že rychlost běhu simulačního času simulujícího systému je totožná s všeobecně vnímanou rychlostí běhu času v realitě.

Vyvinutý simulátor měl ověřit jednoduchost a intuitivnost využití frameworku *Hlava* a zároveň ověřit postupy pro připravovanou metodiku tak, aby bylo prokázáno, že na základě výše uvedeného je možné realizovat interaktivní simulátor provozovaný v reálném čase.

16.2 Vlastní řešení

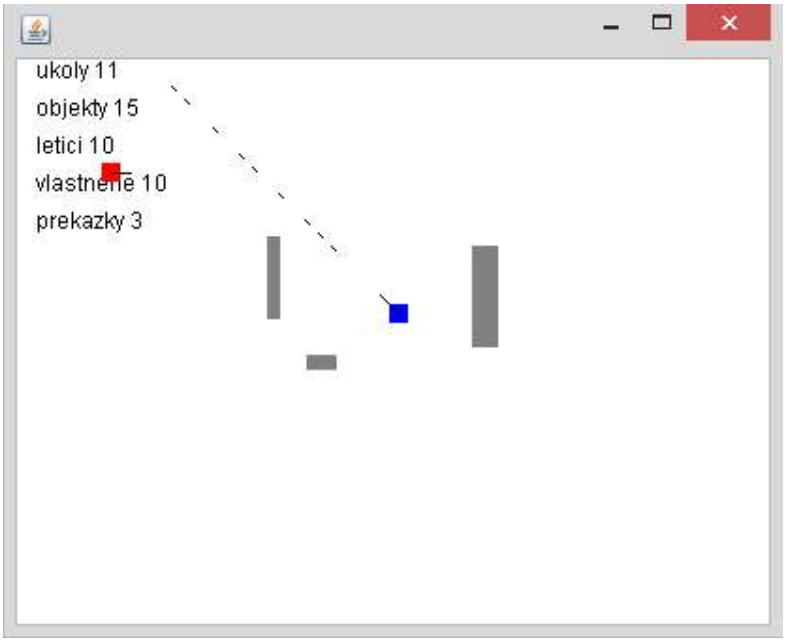
Pro realizaci řešení byla vytvořena scéna (herní plán), který byl volnou plochou obsahující překážky. Každý uživatel (jeden uživatel na jeden federát) byl reprezentován nejvýše jedním avatarem. Uživatel měl možnost ovládat avatar – pohybovat se po ploše a střílet. Úkolem bylo zasáhnout protivníka a nebýt zasažen.

Vlastní simulátor se skládá z (teoreticky) libovolného množství homogenních uživatelských sezení (SEAT). Každé uživatelské sezení přidává do simulace další avatar (uživatelský avatar konkrétního uživatele, obecný avatar pro ostatní). Základní komponenty simulátoru jsou uvedeny v tabulce 3. Systém střelby je řešen na bázi koncepce: „vystřel a zapomeň“ (v simulaci též označován jako princip: „Já vystřelím a ty vyhodnoť, zda jsi byl zasažen.“).

Tabulka 3: Komponenty simulátoru[34]

Název	Typ	Poznámka
Avatar	Entita	Atributy: Vektor natočení, vektor pohybu. Události: Pohyb, střelba, vyhodnocení zasažení.
Střela	Entita	Atributy: Typ střely, vektor pohybu
Překážka	Prostředí	
Herní plán	Prostředí	

[34] Zdroj: Vlastní



Obrázek 25: Debugovací režim interaktivního simulátoru[35]

Vlastní implementace byla provedena v programovacím jazyce Java s využitím jednoduchého grafické zobrazení. Uživatelské okno jednoho uživatele je možné vidět na obrázku 25. Vizualizována je základní situace se dvěma avatary a velmi jednoduchým herním plánem. V levé horní části jsou vidět vybrané stavové informace, které byly přijímány z RTI. Konkrétní detaily je možné nalézt v práci Jakeše (2016).

16.3 Shrnutí

Úspěšná implementace simulátoru, rozvoj jehož stavového prostoru bylo možné intuitivně sledovat přímo ve vizualizačním okně simulátoru nebo v administračních softwarech RTI, pomohla zvýšit věrohodnost předpokladu, že Framework *Hlava* je použitelný pro úlohy tohoto typu (tj. implementace interaktivních simulátorů).

[35] Zdroj: (Jakeš, 2014)

17 Online komponenty pro trenažéry

Dizertační práce se zaměřuje (mimo jiné) na simulátory typu trenažér. Jelikož tyto typy simulátorů mají velmi často vstupně výstupní rozhraní emulující simulovaný systém (např. trenažér stíhacího letounu má vstupní komponenty vzhledem i rozměrem odpovídající simulovanému stíhacímu letounu), bylo vlastní praktické bádání směrováno i do této oblasti. Výsledky byly použity také jako komponenty nejvýznamnější případové studie, která je uvedena v kapitole 20.

17.1 Motivace

Proto, aby bylo o dizertační práci, respektive o *Metodis* možné prohlásit, že umožňuje vývoj také hardwarových komponent simulátorů, bylo třeba provést fyzickou realizaci a softwarovou implementaci několika demonstračních řešení. Protože autor, ani pracoviště nedisponovalo zkušenostmi se zapojením online systémů do (distribuované) simulace, byly první komponenty vyvíjeny zcela od základů. Součástí řešení tak muselo být fyzické zapojování vodičů a implementace ovladačů pro tento nově vzniklý hardware. Až poté bylo možné provést připojení zařízení jako periferií k počítači.

Samostatným výstupem dizertační práce byla implementace online systémů pro připojení rozličných zařízení do simulace. Typově šlo o zapojení proprietárních vstupních zařízení (volant, joystick) a alternativních výstupní zařízení (sedačky, přístrojové desky).

Vyvinuté online komponenty měly velkou hodnotu pro tvorbu autorova nadhledu nad implementací jednotlivých řešení. Čistě softwarové prostředky nemusí stačit pro tvorbu kvalitního rozhraní pro simulátory – proto je tvorba online zařízení (umožňující věrnější emulaci simulovaného systému) výhodná. Na základě výzkumného zaměření vzniklo několik publikací, které přispěly k poznatkům v příslušné oblasti aplikovaného vývoje (Brožek a Fikejz, 2017g), (Fikejz a Brožek, 2017) a (Benedikovič, Brožek a kol., 2016).

17.2 Vlastní řešení

Na základě výše uvedené motivace bylo vyvinuto celkem osm online komponentů. Nejvýznamnější jsou uvedeny dále.

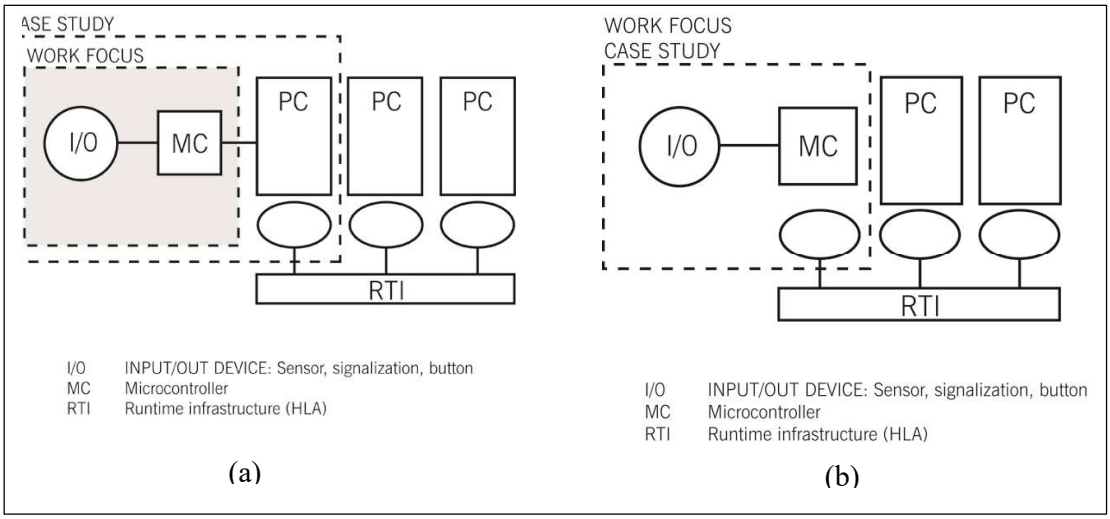
17.2.1 Připojení mikropočítače do simulace

Nejdříve byla zkoumána možnost aplikace principů pro připojení mikropočítače do simulace. Předmětem výzkumu i implementačních prací byla snaha oddělit vhodnými prostředky mikropočítač od simulace a vyhodnotit, která z prověřovaných řešení jsou vhodnější.

Obrázek 26 zobrazuje dvě klíčové architektonické koncepce, které byly v rámci práce ověřovány. Na obrázku je vidět, že obě řešení jsou implementována v souladu s HLA

architekturou. Jako běhové prostředí bylo použito Pitch pRTI. Standardní provoz federátů byl realizován na klasických výpočetních uzlech (PC, notebooky) s operačním systémem Windows, což platí pro obě řešení. Odlišné jsou způsoby připojení online mechanismů (vstupních i výstupních).

V prvním řešení došlo k připojení prostřednictvím standardního výpočetního uzlu jako mezičlánku, na kterém běžel middleware (obrázek 26a). Druhé řešení využilo připojení mikrokontroleru jako vlastního federátu přímo do federace (obrázek 26b). Původním cílem vlastní vývojové činnosti bylo stanovení jednoznačně výhodnějšího implementačního přístupu. Během testování vlastních implementací se projevilo několik klíčových vlastností jednoho i druhého řešení. Přestože se druhé řešení (obrázek 26b) jeví principiálně jednodušší, v rámci testování se ukázala jedna z nevýhod používání simulátorů v souladu s HLA, kterou je vysoká aplikační a komunikační režie. Bylo zjištěno, že fakt, že simulátory spolu komunikují prostřednictvím RTI (tj. dochází k výměně zpráv ve formátu XML), zamezuje použití standardních mikrokontrolerů (např. typu ATMega). Právě tato zařízení nedisponují dostatečným výpočetním výkonem pro tyto potřeby. Využitelnou alternativou jsou moderní zařízení, jakými je například mikropočítač Raspberry PI (v rámci vlastního řešení byla využita druhá generace tohoto mikropočítače). Použití tohoto mikropočítače však principiálně stírá rozdíly mezi oběma implementacemi. Z vlastního testování vyplynulo, že ani jedno z navrhovaných řešení není jednoznačně výhodnější pro obecné použití. Vlastní implementace i testování umožnilo vyslovit praktická doporučení, která uvádí, pro jaké třídy aplikací je vhodnější použití toho kterého implementačního přístupu.



Obrázek 26: Řešení připojení mikropočítače k simulátoru[36]

[36] Zdroj: (Brožek a Jakeš, 2015b)

17.2.2 Online vstupní a výstupní zařízení

Mezi další online zařízení (kromě standardní klávesnice, myši a monitoru) určená pro emulovaná prostředí, patří různé typy periferií, které v různé míře napodobují vzhled skutečných simulovaných zařízení.

V rámci tvůrčí činnosti pracoviště bylo vytvořeno několik vstupně výstupních zařízení snažících se emulovat prostředí externě. Pro tvorbu vstupních zařízení byly použity komponenty pro počítačové hry (volant, pedály, joystick), pro tvorbu výstupních zařízení (otáčkoměry, tachometry) díly ze starších automobilů. Emulace místa řidiče bojového vozidla s joystikem na ovládání palebné věže je vidět na obrázku 27.

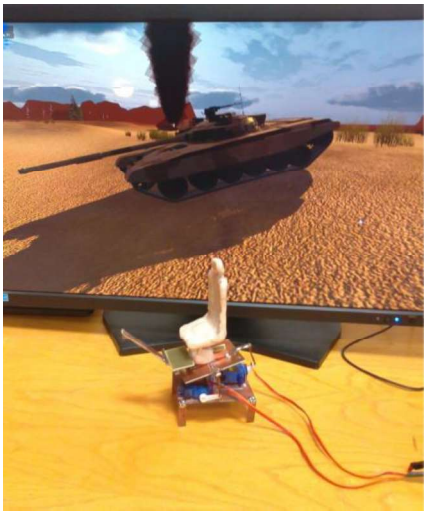
Obrázek 27: Řídící zařízení pro simulátor [37]

Další, čistě výstupní zařízení, které vzniklo, je model sedačky, které bylo zamýšleno pro rozšíření uživatelského vjemu na trenažéru. Model sedačky (obrázek 28) je připojen prostřednictvím mikrokontroleru Arduino, který funguje jako ekvivalent softwarového ovladače wrapper. Model židle se naklání v souladu s náklonem vozidla v simulátoru.

17.2.3 Využití mobilních zařízení a virtuální dopravní sál

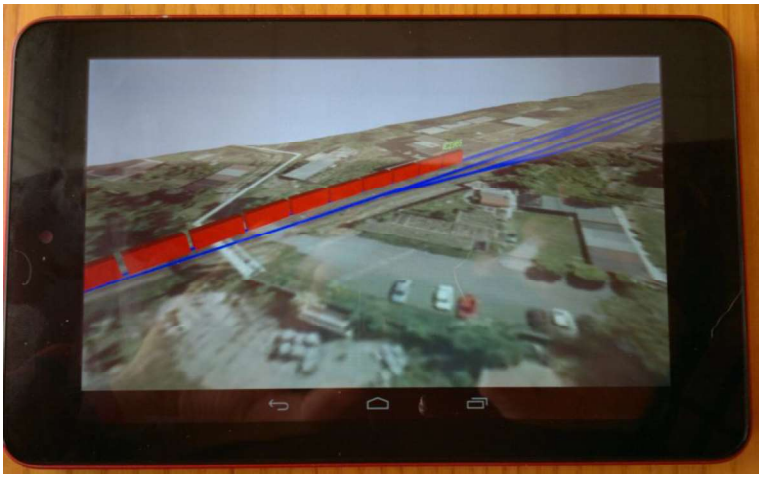
Dále byla zkoumána aplikace mobilních zařízení v simulaci. Pro zavedení mobilních zařízení byl vybrán starší model jednoduchého dopravního systému (simulující systém popsaný v kapitole 15), který byl postupně přepracováván do modelu zjednodušeného virtuálního dopravního sálu, jehož klienty a/nebo vizualizační prvky bylo možné provozovat i na mobilních zařízeních (obrázek 29).

[37] Zdroj: Vlastní



Obrázek 28: Model židle s dvěma servomotory[38]

Celý systém (postavený na technologii HLA) měl hodnotu především proto, že kombinoval řadu rozdílných technologií a přístupů – standardní federáty, mobilní technologie atp. Simulátor byl vyvíjen v souladu s metodikou vytvořenou v rámci této dizertační práce, čímž došlo k částečnému ověření uplatnitelnosti této metodiky.

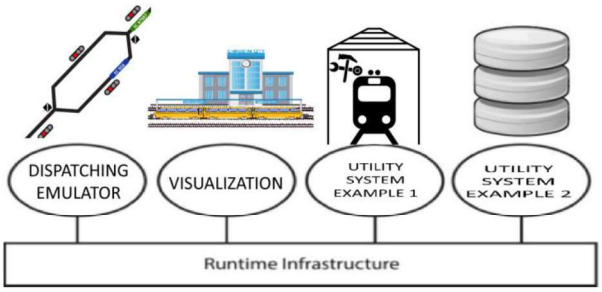


Obrázek 29: OpenGL Vizualizace simulátoru na mobilním zařízení[39]

Na obrázku 30 je možné vidět strukturu simulujícího systému. Původní koncepce (která obsahovala pouze federáty homogenních tříd) byla rozšířena o další heterogenní federáty. Účelem této změny bylo (mimo jiné) otestovat simulaci, kde budou federáty různých typů.

[38] Zdroj: Vlastní
[39] Zdroj: (Hemnani a kol., 2018)

Vizualizační federát je **federát striktně (externě) řízený** (tj. neobsahuje řídící parametr vzhledem k řízení časové osy na RTI, v souladu s HLA). Dále systém obsahuje několik **federátů s vnořeným vrstveným simulátorem**, například Utility system (obslužný systém). Konkrétním příkladem může být depo pro kolejová vozidla (údržba/čištění) – kde vnitřní procesy údržby a čištění jsou vnořeným simulátorem (v souladu s architekturou kompozitního modelu, tak jak byl vysvětlen v předchozích částech práce). Dalším typem může být obslužný systém federátu databázového zapisovače pro AAR systémy (**striktně řízený federát**). Dalším systémem bylo pracoviště dispečera, které mělo vlastnosti **řídícího i řízeného federátu** – to především vzhledem k projevu prováděných změn výhradně v jiném federátu (dispatching emulator na obrázku 30).



Obrázek 30: Simulátor dopravního sálu[40]

17.3 Souhrn

V kapitole byly představeny tři nejvýznamnější zástupci realizovaných online systémů (resp. komponentů systémů) distribuovaných simulátorů. Vývoj každé konkrétní komponenty pomohl ke zvládnutí problematiky online systémů a trenažérů běžících v reálném čase. Vzhledem k tomu, že na řadě etapových výsledků se podíleli další řešitelé (a to samostatně, nebo kolektivně) a systém řízení, nebo posloupnost kroků řešení, byl nastaven v souladu s *Metodis*, došlo opět k určitému ověření dílčích fází metodiky. Především ověření fungování týmu, nebo nastaveného systému řízení je v kontextu práce velmi cenné.

Vybrané online systémy byly (jako dílčí moduly) použity také v dalších simulátorech. Konkrétní příklad aplikace některých online systémů je uveden v kapitole 20.

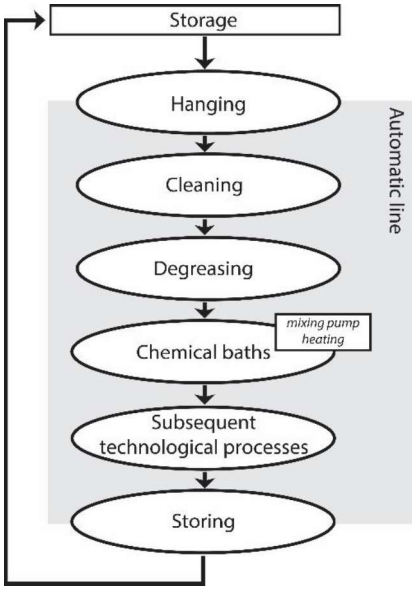
[40] Zdroj: (Brožek, 2015b)

18 Predikční simulační systém

Ve spolupráci s průmyslovým podnikem z praxe vznikls softwarová aplikace pracovně nazvaná predikční simulační systém.

18.1 Motivace

Na základě konzultací s praxí vzniklo zadání pro simulační řešení nasaditelné pro predikci výskytů závad v provozu s vysokou technickou zátěží. Technicky šlo o online simulátor, který je schopen provádět predikce chování a výskytů závad technologického procesu výroby. Základní chemický proces na automatizované lince je zobrazen na obrázku 31. Celý proces vždy začíná a končí na skladě. Jednotlivé kroky (vždy spojené s konkrétním výrobkem) zkoumaného výrobního procesu jsou: Zavěšení (uchycení), očištění (odstranění mechanických např. pískováním), odmaštění (odstranění mastnoty, nebo jiných chemických defektů povrchu), máčení v chemické lázni (různých typů, dle cíle výroby, přičemž máčení může být samostatné, nebo doprovázené elektrolytickým procesem), sušení (nebo zapékání) a zpětné uskladnění. V případě selhání kteréhokoli kroku je výrobek okamžitě likvidován jako zmetek. Opětovné zavedení výrobku do procesu by bylo nákladnější, nežli jeho vyřazení.



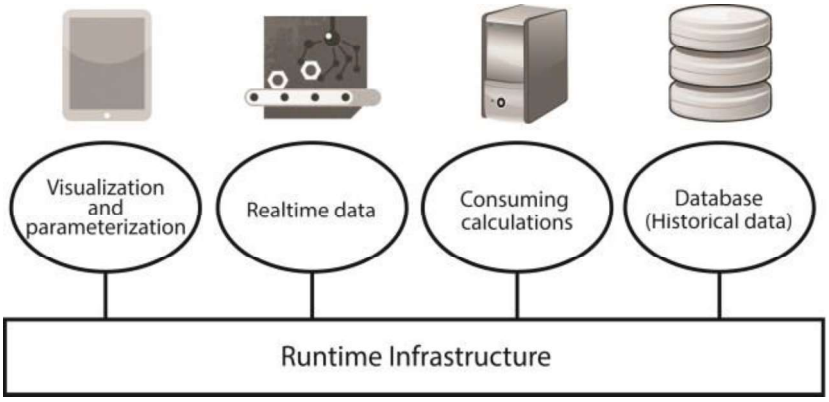
Obrázek 31: Proces chemické úpravy povrchů[41]

[41] Zdroj: (Brožek a Jakeš, 2017)

18.2 Vlastní řešení

Vlastní simulační model se zaměřoval na simulaci provozu linky při získávání online údajů z linky, získávání historických dat z databáze a odhadování potenciálních rizik. Princip simulátoru byl relativně jednoduchý. Simulátor neustále načítal online data z provozu. V případě výskytu vybraných typů poruch (po nichž následují částečné odstávky, nebo omezení provozu) dojde k vypočtení určitého množství předdefinovaných variant dalšího potenciálního provozu linky (např. další provoz s omezenou propustností linky, zastavení provozu a oprava závady, převedení výrobního provozu na jinou linku). Odpovědný zaměstnanec na základě výsledků výpočtů expertně rozhodne, jak bude skutečně probíhat další provoz linky.

Prostřednictvím vnořených simulací bylo možné predikovat rizika a provádět simulační výhledy do budoucnosti, které pomohou (relativně) ad-hoc najít vhodné řešení při poruše (např. odpovědět na otázku, zda se vyplatí dokončit výrobu série se zvýšenou tvorbou výrobků nevyhovující jakosti, nebo je lepší výrobu přerušit, část výrobků odepsat a provést opravy linky). Vlastní počet simulovaných variant i jejich parametrizace byla přenechána na uživatelské volbě. Standardně docházelo k simulaci právě jedné varianty pro jedno konkrétní nastavení dat. Tato varianta byla vždy porovnávána s výsledky stanovenými pro okamžité přerušení výroby (včetně všech důsledků okamžitého zastavení výroby).



Obrázek 32: Architektura simulátoru[42]

Simulační model byl vybudován v HLA na platformě openHLA a využíval federáty čtyř různých typů tak, jak je uvedeno na obrázku 32. Klasickými řízenými federáty byly instance využívané k těžbě historických dat ze systému (na obrázku označené jako Database). Federáty byly zaměřeny na získávání historických dat pro výrobky, výroby obdobných typů a kalkulace příslušných středních dob výroby, kazovosti a dalších technických parametrů pro případné simulace. Další federát získával realtime data z výrobní linky (standardní online data, na obrázku označen jako Realtime data). Tento federát byl provozován jako

[42] Zdroj: (Brožek a Jakeš., 2017)

sběrnice přímo na centrální jednotce řídící automatizaci celé továrny. K výpočtům a vyhodnocením byl využíván zvláštní federát běžící na výkonném výpočetním uzlu (na obrázku označen jako Consuming calculations). Právě na tomto federátu bylo možné realizovat vnořené simulace, které umožňovaly prověřování odlišných provozních variant. Poslední federát (na obrázku označen jako Visualization and parameterization) byl provozován na přenosném zařízení (tabletu), který má k dispozici obsluha/servisní technik. Právě prostřednictvím tohoto federátu je možné provést parametrizaci simulačních experimentů a vyhodnotit nejvhodnější koncepci pro další servis/provoz výrobní linky/celé továrny.

18.3 Shrnutí

Představený simulátor byl pro vlastní řešení cenný, protože byl budován od počátku v souladu s kroky vznikající *Metodis*. Zároveň docházelo k úpravě *Metodis* právě na základě zkušeností s konstrukcí tohoto simulátoru.

Rozsahově došlo k realizaci všech fází *Metodis* až na komplexní validaci, která byla nahrazena testovacím nasazením do provozu.

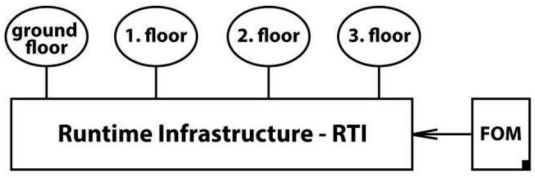
Velmi důležitým poznatkem z řešení simulátoru byl také fakt, že OpenHLA, tedy volně šiřitelná implementace RTI, není dokončena a není vhodná pro nasazení v jiné situaci, než je výuka. Množství neimplementovaných metod, které jsou standardně vyžadovány normou, bylo příliš vysoké.

19 Simulátory provozu polikliniky založené na různých architekturách

Tato kapitola představuje simulační studii, které významným způsobem přispěla k ověření části vyvíjené metodiky. Simulační studie je založená na dvou simulačních modelech téhož simulovaného systému – polikliniky. Každý ze simulačních modelů vznikl v rámci bakalářské práce jiného ze studentů. Společně studenti prováděli analýzu, sběr (měření) a zpracování (statistické vyhodnocení) dat a interpretaci výsledků simulační studie. Nezávisle prováděli zpracování simulačních modelů (jeden vznikl v nástroji Arena, druhý v programovacím jazyce Java za využití Pitch RTI). Výsledek byl pro dizertační práci významný především proto, že experimentálně ověřil, zda jsou studenti (kteří jsou spíše poučenými laiky, nežli odborníky), za využití *Metodis*, schopni vybudovat distribuovaný simulátor a interpretovat jeho výsledky.

19.1 Motivace

Na konkrétní komerční poptávku byla, za využití posloupnosti kroků navržených v *Metodis*, vytvořena simulační studie zaměřená na provoz polikliniky ve Dvoře Králové nad Labem. Simulační studie měla odpovědět na otázky týkající se blížící rekonstrukce polikliniky – především v kontextu omezení provozu při rekostrukci a možnosti/vhodnosti přesunu ordinací/pracovišť služeb na jiná fyzická místa v budově. Kromě faktického zadání nebyla dostupná žádná další data. Detaily o těchto řešeních je možné nalézt v pracích Bašeho (2016) a Samotána (2014). Právě práce Samotána (2014) obsahovala cenné výsledky, které pomohly potvrdit uplatnitelnost částí *Metodis.*



Obrázek 33: Příklad architektury simulačního modelu provozu polikliniky[43]

19.2 Vlastní řešení

V rámci řešení proběhla konstrukce dvou rozdílných simulujících systémů pro jeden systém simulovaný. Všechny informace o vstupních proudech, vnitřních procesech i dobách jejich trvání bylo třeba naměřit. Dále bylo třeba tato naměřená data statisticky zpracovat, navrhnout hypotézy o typech rozdělení pravděpodobnosti náhodných veličin a tyto hypotézy

[43] Zdroj: (Samotán, 2014)

ověřit (testem dobré shody). V rámci realizace byl pro oba systémy proveden jen jeden sběr, a jedna analýza dat.

Faktické odlišnosti v postupu obou studentů je možné nalézt až při tvorbě simulujících systémů. Baše (2016) vytvořil simulační model v nástroji Rockwell Arena a Samotán (2014) použil programovací jazyk Java a framework *Hlava*. Konkrétní návrh dekompozice navržený Samotánem (2014) je vidět na obrázku 33. Na čtyřpatrovou budovu byla použita metoda dekompozice rozkladem a výsledný distribuovaný simulační model se skládá ze čtyř federátů. Protože rozsah případové studie byl poměrně malý, mělo nasazení distribuované architektury spíše demonstrativní charakter.

19.3 Shrnutí

Simulátory prezentované v této kapitole jsou specifické tím, že byly vybudovány proto, aby jejich prostřednictvím mohla vzniknout skutečná simulační studie konkrétního problému specifikovaného soukromým subjektem. Tato studie byla navíc realizována v souladu s postupy *Metodis* a autor metodiky sloužil pouze v roli konzultanta, nikoli aktivního řešitele případové studie. Právě toto opatření sloužilo ku prospěchu dopadu případové studie – totiž nezávislému ověření posloupnosti procesů tak, jak jsou uvedeny v *Metodis*. Pro realizaci byl k dispozici simulovaný systém a cíle studie. To umožnilo využít téměř všechny fáze *Metodis*. Navíc došlo k využití vybraných fází ve společném řešení dvou rozdílných simulujících systémů a některé fáze potom individuálně pro řešení konkrétního simulátoru.

Protože řešitelé provedli úspěšné vymezení systému, určení procesů, sběr dat, návrh simulujících systémů, jejich implementaci i nasazení, lze prohlásit, že realizace tohoto simulátoru částečně přispěla k ověření *Metodis*.

20 Případová studie většího rozsahu

Případové studie, které byly až doposud uvedeny v rámci dizertační práce a představeny mezinárodní komunitě, mají rozsah simulátorů, které je možné implementovat v přijatelném čase pouze jedním autorem. V situaci, kdy je autor takového simulátoru znalý problematiky vývoje simulačního softwaru/aplikace simulačních paradigmat, není možné průkazně vyhodnotit přínos metodiky jako takové.

Klíčovým prvkem pro ověření metodiky je možnost jejího praktického využití. Toto využití je, bez dalšího, možné sledovat na simulátoru z libovolné aplikační domény. Omezujícím kritériem je to, aby simulovaný systém splňoval podmínku decentralizace řízení. Přestože je vlastní metodika obecnější, podmínka byla zavedena vzhledem k tématu dizertační práce, která vyžaduje ověření na modelu systému s decentralizovaným řízením.

Na základě této skutečnosti vznikla rozsáhlejší simulační studie, které byla realizována širším autorským týmem. Záměrem řešení případové studie bylo ověření praktické použitelnosti metodiky. Požadavkem bylo sestavení řešitelského týmu, který bude mít dostatečné znalosti v oblasti implementace softwaru, ale žádné, nebo minimální, zkušenosti z oblasti tvorby simulátorů. Výchozí premisu bylo možné v případě vybudování takovéhoto týmu vyslovit v tomto znění: Pokud by byl takovýto řešitelský tým schopen, za použití a přispění metodiky, vytvořit simulátor netriviálního rozsahu, bylo by možné metodiku prohlásit za ověřenou. Motivací pro tvorbu rozsáhlejší případové studie, která je představena v této kapitole, tak primárně nebylo (na rozdíl od předchozích případových studií) řešení simulačního problému, nýbrž ověření metodiky.

Při přípravě tématu případové studie se vyskytlo několik příležitostí, které vytvářely potenciální simulované systémy, jež by bylo možné využít. Každé téma bylo důsledně analyzováno a vyhledány jeho silné i slabé stránky. Na základě této analýzy bylo, ze čtyř potenciálních témat, vybráno jedno cílové téma. Potenciálními adepty na řešení byly:

1. Virtuální dopravní sál železniční dopravy
2. Simulační model provozu vybraného úseku dálnice
3. Trenažér vozidla autoškoly
4. Taktický simulátor bitvy

Protože jde o rozsáhlejší kapitolu, bylo zvoleno její členění tak, aby maximálně usnadnilo orientaci čtenáře. Struktura příslušných podkapitol je následující:

- **Vymezení problému** seznamuje čtenáře s předmětným simulačním řešením a jeho obsahem.
- **Specifikace systému** uvádí vybrané technické detaily simulátoru.
- **Komponenty řešení** uvádí, jaká další řešení se stala součástí simulátoru.

- **Zpracování řešení** shrnuje, jakým způsobem bylo řešení zpracováno.
- **Řešitelský tým** se vyjadřuje ke složení řešitelského týmu.
- **Vyhodnocení nasazení metodiky** při realizaci případové studie uvádí klíčové sledované charakteristiky, které během řešení sloužily jako ukazatele pro ověření metodiky.

20.1 Vymezení problému

Pro prvotní popis systému je výhodné použít poněkud vyšší míru abstrakce hraničící až s vědecko-populárním pohledem na celý problém. Tento úhel pohledu je však pro zkoumání výhodný, neboť intuitivní popis problematiky plní účel této kapitoly.

Obrázek 34: Ilustrační případ[44]

Na obrázku 34 je ilustrace klasické arkádové počítačové hry (občas nazývané též simulační hrou), ve které jednotliví uživatelé využívají jako svůj avatar tanky (každý uživatel vždy právě jeden tank). Uživatelé jsou rozděleni do dvou týmů. Cílem týmů je vždy eliminace protivníka. Na rozdíl od simulátoru zde nejsou respektovány skutečné technické specifikace a parametry pancéřování nebo kanónů jsou podřizovány hratelnosti.

Zamýšlený simulátor se řídí obdobnými pravidly, od hry se však odlišuje v řadě klíčových prvků, kterými jsou především: Tank či bojové vozidlo nevydrží řadu průstřelů, není možné se během boje zázračně léčit či opravovat, jedno vozidlo může být ovládáno více uživateli (řidič x střelec), dynamika jízdy vozidel i chování projektilů je blíže realitě, výhled z vozidla odpovídá skutečnému výhledu, nikoli tak zvanému pohledu třetí osoby, nejsou přípustné vady způsobené určitou mírou tolerance (laggy, ztráty spojení, vysoká latence) aj.

Navržený simulátor vytváří intuitivní virtuální prostor pro střet dvou soupeřících stran. S jistou mírou abstrakce si ho lze představit jako vymezený fyzický prostor o určité rozloze, ve kterém soupeří dvě strany. Základními prostředky, kterými mohou toto soupeření realizovat, jsou bojová vozidla. Bojová vozidla mohou eliminovat vozidla soupeře, nebo být eliminována, dále mohou střílet, pohybovat se, skrývat se atp. Celý souboj probíhá s omezeným množstvím zdrojů na omezeném prostoru. Cílem simulátoru je vytvořit

[44] Zdroj: Vlastní

uživatelské posty, na kterých bude možné získávat zkušenosti z boje ve virtuální, skutečně nebojové situaci a tyto zkušenosti přenášet na simulovaný systém.

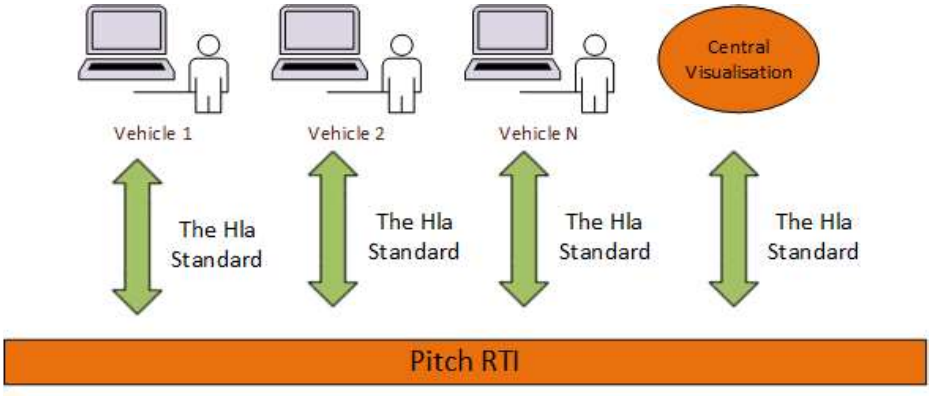
Taktický simulátor bitvy je simulační řešení, kde uživatelé, prostřednictvím svých avatarů, svádí virtuální souboje. Rozdíl mezi počítačovou hrou a simulátorem nemusí mít jasně ohraničený přechod. Oba typy softwaru mají řadu společných prvků. Dokonce existuje herní segment nazvaný simulátory. Kardinální rozdíl mezi oběma typy softwarů je však patrný již od okamžiku tvorby prvních požadavků na software. Cíle hry a simulátoru jsou již v okamžiku definice požadavků odlišné. Cílem hry je primárně zábava. Cílem simulátoru je studie dané problematiky, nebo trénink; v obecnějším pojetí je simulátor prostředkem, jehož výsledky je možné aplikovat na simulovaný systém. V simulátorech typu trenažér je pro označení osoby odborně kvalifikované jako man-in-the-loop přijatelné označení hráč.

Cílem hráčů, rozdělených do týmů, je eliminace týmu protivníka, proniknutí do oblasti, nebo udržení oblasti. Uživatelskými avatary jsou v tomto konkrétním případě bojová vozidla. Podle nastavení a typu vozidla je jeden avatar obsluhován jedním až třemi hráči. Avatary jsou dvou typů. Typ Humvee může být obsluhován až dvěma hráči (řidič, střelec) a typ tank až třemi hráči (řidič, velitel, střelec).

20.2 Specifikace systému

Simulátor vznikl na platformě Unity3D, která zajišťuje poměrně dobré možnosti vizualizace a podporuje kvalitní fyzikální jádro, které umožňuje provádění sofistikovaných grafických interakcí. Práce na simulátoru byly realizovány v souladu s metodikou tak, aby bylo možné ověřit i to, zda je metodika použitelná pro rozsáhlé projekty.

Celé řešení je navrženo jako multi-uživatelská interaktivní simulace s vyšší úrovní abstrakce použitelná jako taktická simulace (neklade si za cíl být přímou emulací a věrně odrážet uživatelská seat). Architektura řešení je na obrázku 35.

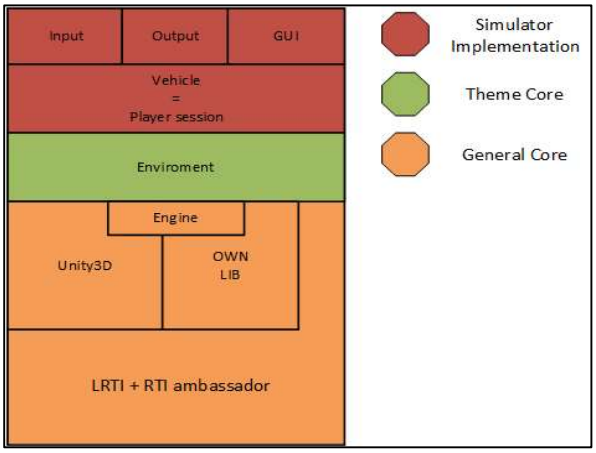


Obrázek 35: Architektura řešení[45]

[45] Zdroj: Vlastní

Na obrázku je zobrazeno členění simulátoru s granularitou jednotlivých vozidel (resp. federátů reprezentujících bojová vozidla). Jedno vozidlo je vždy realizováno na jednom výpočetním uzlu bez ohledu na počet uživatelských seat. Tento uzel je prostřednictvím RTI propojen se zbytkem distribuovaného simulátoru. Jednotlivá vozidla mají vlastní logiku a synchronizaci, kterou lze provádět prostřednictvím RTI. Nezáleží tak na pořadí připojení federátů, dokonce je možné federáty připojovat za běhu tréninku (simulačního výpočtu). Uživatelská seat se dají aktivovat nezávisle, nebo lze využít emulace uživatelských seat prostřednictvím middleware. Tento middleware pak může vytvořit větší množství uživatelských emulací jednotlivých seat (vstupních i výstupních, například polohovací sedadlo pro zvýšení uživatelského prožitku). Důsledkem tohoto opatření je možnost ovládat jedno vozidlo více uživateli (přičemž každý uživatel musí být v rámci tohoto ovládání v exkluzivní roli). Simulátor klade z výkonových důvodů omezení, kterým je vytvoření nejvýše jedné instance seat s virtuální realitou na jeden avatar. Toto omezení je stanoveno technickými možnostmi výpočetních zařízení, na kterých docházelo k vývoji celého řešení.

Komunikace probíhá prostřednictvím pitch RTI dle strandardu HLA (použitou verzí je IEEE1516:2010). Veškerá komunikace probíhá prostřednictvím RTI a je ve formátu XML. Časová synchronizace na úrovni federace (time management) i přiřazení odpovědností za objekty (ownership management) je ponecháno plně v režii RTI serverové komponenty. Dohled nad řešením je možný prostřednictvím softwaru od společnosti Pitch (Pitch Control Center).



Obrázek 36: Struktura dílčích federátů[46]

Na programové úrovni byl původně nasazen původní framework *Hlava* (kapitola 14), který byl později nahrazen robustnějším komerčním řešením skládající se z komponent Pitch RTI Controller a Pitch Commander. Při konstrukci byl použit Pitch pRTI – v každém federátu je přímo implementován RTI ambassador. Na úrovni federátu došlo k využití časových

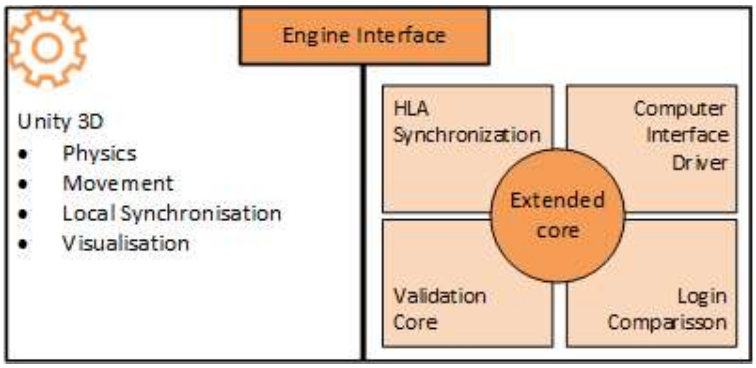
[46] Zdroj: (Brožek a kol., 2017c)

synchronizací herních engine tak, aby odpadla nutnost explicitní synchronizace na úrovni jádra. Tedy, časová synchronizace federátu je řešena prostřednicvím hybridního (spojitě-diskrétního) fyzikálního jádra herního engine se třemi úrovněmi zpracování synchronizační vazby (využity byly metody FixedUpdate, Update a LateUpdate), čímž došlo k vytvoření velmi těsné vazby mezi stavovým prostorem simulátoru a vizualizační scénou. Vzhledem ke zvolené architektuře je vlastní vizualizace objektu pevnou součástí jeho stavového prostoru. Jde o odlišný způsob, nežli je využíván standardně při tvorbě tradičních simulátorů (online animace postavená nad stavovým prostorem). V tomto případě je animace součástí stavového prostoru a případné odchylky v animaci se projeví přímo ve stavovém prostoru. Výhodou je, že celé řešení je snáze udržovatelné a počet maximálně zapojitelných federátů vzrostl.

Architektura jádra softwarového řešení jednotlivého federátu je prezentována na obrázku 36. Obrázek ilustruje tři vrstvy softwaru.

1. Na první úrovni (na obrázku oranžovou barvou) je místní **RTI komponenta**, v tomto konkrétním případě jde o řešení Pitch pRTI. Součástí řešení je dále herní engine Unity3D, který poskytuje podporu pro vizualizaci, fyzikální engine a rutiny pro lokální synchronizaci. Součástí řešení jsou vlastní knihovny implementované v programovacím jazyce C#. Jde o vlastní rozšíření engine o konkrétní fyzikální jevy (např. balistiku), systémy pro tvorbu objektů, přípravu prefab atp. Vlastní knihovny a rutiny herního engine tvoří společně (vlastní) rozhraní, které je dále použito pro konkrétní simulátory. Součástí vlastních knihoven je také validační jádro a systém pro správu logů vč. podpory AAR (více níže). Detaily této vrstvy jsou ilustrovány na obrázku 36, který je detailem části obrázku 37 v rozsahu Engine, Unity3D a OWN LIB (tj. prakticky celé vrstvy bez prostředků RTI).
2. Zelená vrstva je společná pro federáty obdobných typů (například jen pro federáty uživatelských avatarů). **Vrstva prostředí** (na obrázku označeno jako Enviroment) zajišťuje veškeré parametrizace v rámci odlišných simulačních scénářů, které mají vliv na prostředí. Zasazení prostředí do hlubší úrovně simulace bylo nezbytné pro korektní zachování veškerých funkcionalit a přitom otevřeného rozhraní pro tvorbu uživatelských session.
   Do úrovně prostředí je možné (dynamicky) umisťovat jednotlivé uživatelské avatary. Tyto avatary jsou identifikovány v rámci simulace jako samostatné entity. Simulace není schopna dále určit, zda je avatar řízen jedním, nebo více uživateli.
3. Červená vrstva je pak implementována samostatně (mírně odlišně) pro každé jednotlivé typové řešení (uživatelské rozhraní, vizualizátor, pro každý typ vozidla atp.). **Uživatelské rozhraní, vstupní rozhraní i výstupní rozhraní** je součástí této vrstvy. Toto rozhraní je vzájemně zástupné, resp. potenciálně multi-duplicitní (použit návrhový vzor Observer), což znamená, že do scény může být n-násobně duplikováno libovolné množství vozidel libovolného typu. Každé vozidlo tak může mít vlastní GUI a řadu vstupních či výstupních tříd.

Pokud dojde ke shrnutí výše uvedených skutečností, lze říci, že první vrstva je společná pro všechny simulátory postavené na této platformě. Prostřední vrstva je společná pro každou třídu simulátorů a obsahuje odlišnosti stavového prostředí (např. letadla, či vozidla). Tím, že je i prostřední vrstva spojena s RTI, může používat OMT, což umožnilo vybudovat vrstvu flexibilnější. Nejvyšší vrstva je společná pouze pro instance stejného objektu v rámci simulátoru (např. všechny tanky v rámci konkrétního simulátoru mají tuto vrstvu strukturálně totožnou, přestože její konkrétní stavový prostor může být jiný).



Obrázek 37: Architektura části jádra federátu[47]

Detailní pohled na architekturu řešení umožňuje obrázek 37. Jak bylo uvedeno výše, obsahuje detail vlastního jádra. Řešení je postaveno na Unity3D a využívá jeho zpracování fyzikálních výpočtů pro řadu činností – výpočty chování prostředí (vítr, světlo), pohyb (tření, hmotnost, dodávaná síla), střelba (vektor směru, rychlost, balistická křivka) a vizualizace (3D scéna). Do jádra herního engine je jako atribut přidán údaj o simulačním čase a v případě potřeby i o rychlosti plynutí simulačního času, který se může přípustně odchylovat až o 5 % od času reálného. Zavedením této odchylky je dosaženo synchronizace v případě problémů v síti (nebo jiných technických problémů). Vlastní jádro obsahuje napojení na synchronizaci, kterou poskytuje RTI, dále obsahuje zapisovač (Logger) a zpřístupňuje všechny ovladače počítače (například volant, pokud by byl připojen například přes USB). Přínosem zvoleného řešení je kompomentární přístup k simulátoru, který je nyní udržovatelný po částech.

Všechny tyto techniky umožnily zavedení dalších pokročilých paradigmat. Jedním z nich je například zavedení systému AAR (after action review), tedy možnost zasazení simulátoru do vybraného času v minulosti, který kausálně vrací stavový prostor zpět.

Část vizualizační scény je vidět na obrázku 38. Obrázky ilustrují standardní grafickou kvalitu realizovaného simulátoru v roce 2015. Na obrázku je vidět vozidlo Humvee, terén

[47] Zdroj: (Brožek a kol., 2017c)

zakončený strmým ohraničením a skála. Drobné textury (např. tráva) se generují náhodně a slouží pouze k vizuálnímu komfortu řešení.

Obrázek 38: Bojové vozidlo v rámci Taktickéh simulátoru bitvy[48]

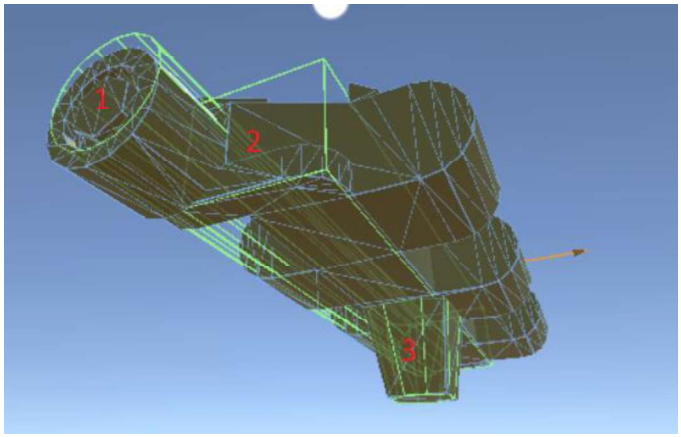
20.3 Komponenty řešení

Vlastní simulátor se skládá z komponent, které byly realizovány jednotlivci (v rámci bakalářských či diplomových prací) nebo týmy (v rámci předmětu Projekt vývoje softwaru). Příklad několika nejvýznamnějších komponent je uveden v této podkapitole.

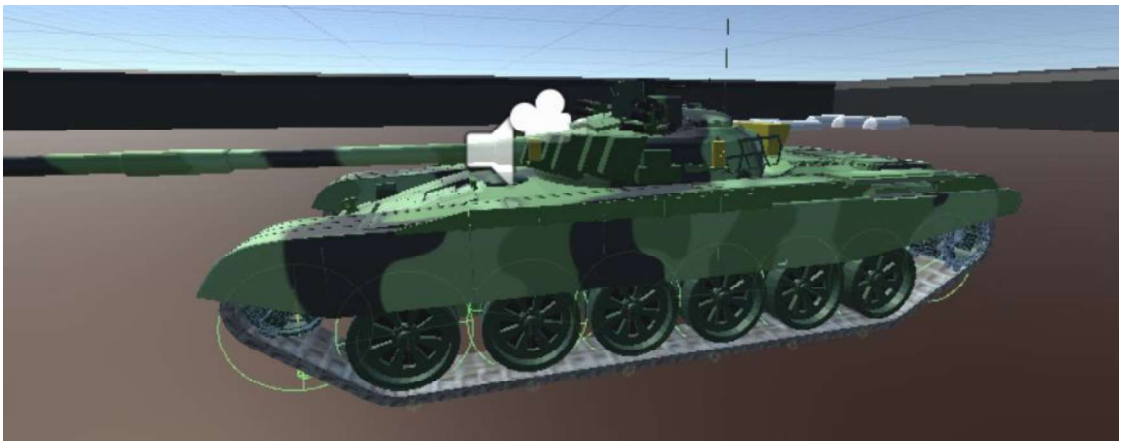
20.3.1 Simulátor tanku T72M4CZ

Proto, aby simulátor mohl využít vysoce kvalitních modelů vozidel, byla vypracována vlastní sada assetů. Například tank T72M45CZ, resp. jeho 3D model, chování (logika dynamiky jízdy, možnost otáčení věží, sklápění hlavně) byly realizovány v rámci bakalářské práce (Kasal, 2016). O tom, že vlastní model je vypracován vysoce podrobně svědčí například obrázek 39, kde je vizualizován model jednoho obecného dílku pásu. Vizualizován je také výsledný model tanku, přičemž je zobrazen s nižší úrovní LOD (Level of Details). Zdůraznit je však potřeba fakt, že součástí modelu jsou, kromě scriptů pro pohyb, také veškeré audio soubory.

[48] Zdroj: (Kasal, 2016)



Obrázek 39: Fragment pásu[49]

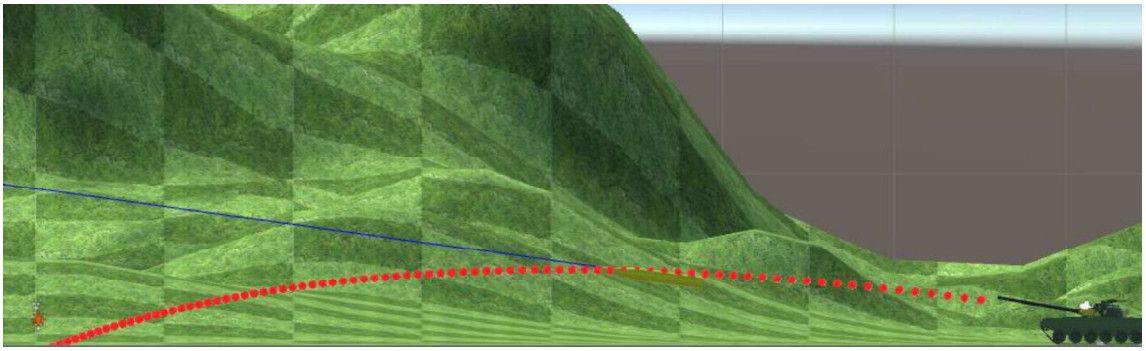


Obrázek 40: Model tanku T72M4CZ[50]

20.3.2 Simulátor střelby po balistické křivce

V rámci další práce (Beránek, 2016) vznikl modul, který se zabývá střelbou po balistické křivce. Vlastní logiku výpočtu, zapojení rychlosti větru, hmotnosti projektilu, směru projektilu, nebo komplexnějších vlastností řešení (jako typu munice), popisuje právě výše uvedená práce. Součástí práce bylo nejen naprogramování modulu pro střelbu, ale také řešení logiky nabíjení, nebo vyhodnocení ohrožení projektilem (tj. probití). Proto, aby bylo řešení dostatečně věrné byla využita kombinace fyzického projektilu s predikčním vektorem. Princip směřování jednoho konkrétního predikčního vektoru je možné vidět na obrázku 41.

[49] Zdroj: (Kasal, 2016)
[50] Zdroj: (Kasal, 2016)



Obrázek 41: Trajektorie střelby[51]

20.3.3 Konektor na externí komponenty a jejich prototyp

Simulátor svým rozsahem počítal s připojením hardwaru, který není standardní součástí počítače. Proto vznikl softwarový konektor pro připojení externích vstupně výstupních komponentů. Pro ověření funkčnosti konektoru bylo vytvořeno několik prototypů, jako například řidičské sedadlo se dvěma servomotory (obrázek 42), které svým náklonem, nebo vibracemi, může zlepšit uživatelský prožitek, při používání simulátoru. Více lze nalézt například v bakalářské práci Zachovala (2017).

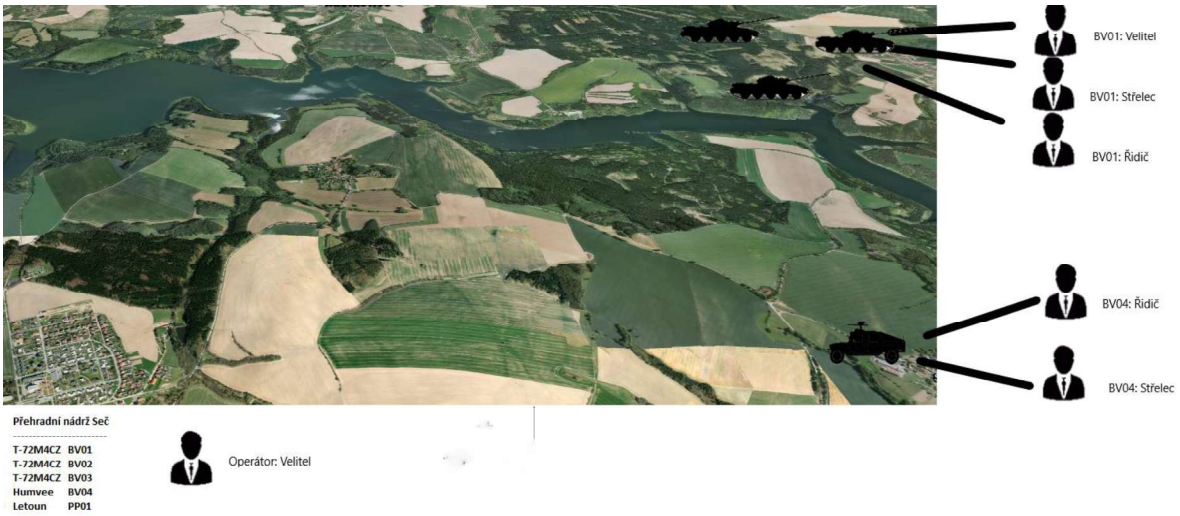
Obrázek 42: Sedačka[52]

[51] Zdroj: (Beránek, 2016)
[52] Zdroj: Vlastní

20.4 Zpracování řešení

Nyní bude věnována pozornost způsobu použití vlastní metodiky při řešení případové studie. Další členění na podkapitoly nižších úrovní odpovídá svými názvy, názvům vybraných kroků vlastní metodiky. Práce byly realizovány postupně v souladu s metodikou.

Na obrázku 43 je možné vidět ilustrační pohled z velitelského stanoviště na situaci kolem přehradní nádrže Seč. V rámci aktuálního scénáře simulace jsou nasazeny čtyři bojová vozidla, tři tanky T-72 M4CZ a jedno vozidlo Humvee. Do simulátoru je zapojeno celkem šest osob. Jedna osoba coby velitel operace, který má k dispozici přehledovou mapu tak, jak je vidět na obrázku 43. To, zda jsou viditelná jednotlivá vozidla je závislé na zařazení vozidla (vlastní/nepřátelské) a stavu detekce vozidla (u nepřátel, resp. nepřátelských vozidel). O množství a pozicích vozidel nepřítele se velitel může jen domnívat. Dále jsou zapojeny tři osoby jako obsluha tanku T-72M4CZ s označením BV01 a dvě osoby coby obsluha vozidla Humvee s označením BV04. Vzájemná komunikace jednotlivých osob a ovládání svěřeného vozidla resp. jeho části je vlastním příkladem činností, které naplňují jejich uživatelské role v kontextu tohoto simulátoru.



Obrázek 43: Schéma simulátoru - pohled shora[53]

20.4.1 Přípravná fáze

Přípravná fáze řešení vyžadovala činnosti jako vymezení problémů, stanovení cílů a ohraničení problému. Činnosti byly částečně uvedeny v *Metodis*. Součástí fáze bylo také stanovení rolí, které je ilustrováno v kapitole 4 metodiky. Faktické personální obsazení však nebylo v době přípravy známo. Vzhledem ke specifickému prostředí v akademické sféře bylo třeba připravit příslušné role anonymně pouze se znalostí profilu příslušného řešitele.

[53] Zdroj: Vlastní

20.4.2 Analýza

Vlastní analýza řešení byla postavena na znalosti příslušných požadavků a zároveň na řešeních konkurenčních (uvedeny například kapitole 7). Analýza byla provedena na logické úrovni s přihlédnutím ke standardním požadavkům jednotlivých uživatelských session i doplňkových systémů (např. společný vizualizátor AAR). Pro potřeby analýzy byla provedena dekompozice modelovaného systému na větší logické celky, které v této fázi mohou být značně nesourodé. Jde o:

- Bojové vozidlo (entita)
- Střelbu (aktivita)
- Prostředí (permanentní entity simulační scény)
- Další aktivity (doplňkové a režijní rutiny)
- Další rozhraní (rozhraní pro externí systémy a zařízení)

Základní vymezení pojmů, které bylo využitelné pro další kroky zpracování softwaru, je následující:

- **Bojové vozidlo** může být různých typů – pro diskutovaný simulátor minimálně jeden pásový a jeden kolový prostředek. Vozidlo bylo realizováno s možností spojité dynamiky jízdy, kde vliv na jízdní vlastnosti vozidla má nejen vozidlo samotné, ale také terén (např. pevnost terénu) a povětrnostní vlivy (např. námraza, silný boční vítr). Vozidla mají vypracovány 3D grafické modely a jsou texturovány. Každé vozidlo má stanoveny pozice pro simulované umístění obsluhy a stanovení zorného pole výhledu obsluhy v rámci uživatelských seat (např. řidič tanku vidí vpřed nehledě na natočení děla tanku). Vozidlo podporuje delegaci řízení na více osob. Mobilita vozidla je limitována nejen na pohyb celého vozidla, ale také na jeho jednotlivé části – například otočná věž s dělem. Vozidla mají další technické prostředky, jako například zařízení pro zadýmování. Vozidlo je schopno střelby. Střelba i zadýmování jsou přesněji probrány v jiné části analýzy. Z pohledu vozidla je třeba neopomenout implementaci přebíjení, kteréžto je důležitým doplňkem střelby. Klíčové tak byly například časy přebíjení a nabíjení.
- **Střelba** je realizována s logikou vystřel a zapomeň. Problematika logiky zásahu byla delegována na objekty, které je možné zasáhnout. Vlastní střela je buď bez projektilový typ (mina, zadýmení), nebo projektilový typ. Pro oba typy střelby se spustí příslušné částicové a zvukové efekty. Projektilová střelba respektuje balistickou křivku a ta respektuje další efekty prostředí (např. vítr). Různé typy projektilů mají vlastní specifika chování, letu i efektu při dopadu střely. Implementovány jsou minimálně střely průrazné, průpalné a tříštivotrhavé. Standardní chování umožnilo implementovat možnost penetrovat měkké cíle (např. zdi) s pokračováním letu náboje. Součásti vizualizace scény jsou trasírky pro jednotlivé typy střel, tak jak jsou běžně používány.

- **Prostředí** je prostor dostupné scény s rozlohou nejméně 1 km^2. Členitost terénu a obsah přírodních či umělých překážky je žádoucí a pro vytvořené scénáře doporučený. Použitelná je například voda, stromy, budovy. Členitost terénu musí mít dopad na možnost a vlastnosti pohybu vozidel a na viditelnost. Terén podléhá temporálním vlivům (různé denní doby, pohyb slunce, noc), klimatickým vlivům (sněžení, déšť, mlha) a obsahuje komplexní překážky, jako zemljanky, civilní budovy (i obsazené), ruiny, minová pole aj.
  Součástí prostředí byly prostředky aktivní detekce – kamerové systémy či prostředky leteckého průzkumu, které umožní zvýšit možnosti přehledu v rámci bojiště. Tyto prostředky jsou ovládány z velitelského stanoviště.
- **Další aktivity**, které jsou uvažovány implicitně, ale v rámci analytické přípavy řešení je třeba je zmínit jsou například (z)pozorování nebo vzájemná komunikace uživatelů. Tyto další aktivity spolu s aktivitou pohybu a střelby tvoří kompletní paletu aktivit simulátoru.
- **Další rozhraní**, kromě rozhraní vlastních vozidel, bylo k dispozici i rozhraní centrálního dohledu. Toto rozhraní umožňuje vytvoření určité míry přehledu o současném stavu simulátoru, a dokonce případně navrácení simulátoru do stavu v minulosti v rámci aplikace principu AAR. Zároveň další rozhraní umožnilo připojení externích systémů či zařízení, jde o prostředky vizualizace, nebo o vstupní zařízení.

Po analýze na logické úrovni byly provedeny příslušné kroky k rozkladu, dekompozici a tvorbě příslušných rozhraní. Klíčová byla především dekompozice na jednotlivé subsystémy, které bylo možné delegovat v rámci dílčích týmů:

- **Bojové vozidlo: Tank T72M4 CZ** je entita simulátoru spojená s 3D modelem vozidla složeným ze stovek částí, textur, assetů a scriptů. Mezi hlavní assety patří zvukový doprovod. Textury odpovídají standardnímu maskování. Scripty zajišťují výpočty dynamiky jízdy, dynamiky pohybu pohyblivých součástí, nabíjení, přebíjení a střelby. Součástí jsou také programové části umožňující připojení externích zařízení – prostředků pro virtuální realitu, zařízení emulující pohyb sedačky atp.
- **Bojové vozidlo: Humvee M1046** odpovídá svým rozsahem tanku T72M4 CZ, liší se vizuálními parametry (3D modelem), hodnotami atributů ovlivňujícími vlastnosti (rychlost, zrychlení) a faktem, že pro ovládání je připojitelné odlišné zařízení (volant).
- **Centrální přehledový systém** slouží jako velitelské stanoviště. Fakticky jde o přehledovou mapu, která umožňuje ve svém základním nastavení pouze sledování situace. Předpokladem je, že tato komponenta bude sloužit primárně pro výcvik personálu, který komunikuje s tanky. Kromě základního vizualizačního systému (2D vizualizace, textury) je z tohoto pultu možné aktivovat systém AAR. Součástí jsou záznamové subsystémy.

- **Společné komponenty** celého simulátoru jsou především OMT, kde je definice všech interaktivních (resp. interagujících) objektů, rozhraní pro externí zařízení, systém časové synchronizace aj.

20.4.3 Architektura, implementace a jednotkové testy

Architektonická část řešení a vlastní implementace byla vždy ponechána v kompetenci dílčího řešitele (nebo týmu dílčích řešitelů). Detaily o konkrétních návrzích jde najít vždy v bakalářské práci, diplomové práci, nebo závěrečné zprávě z předmětu Projekt vývoje softwaru, která se tématem zabývá. Návrh byl vždy konzultován s hlavním řešitelem.

Součástí povinností jednotlivých řešitelů bylo také vytvoření a uplatnění jednotkových testů v souladu s metodikou. Jednotkové testy měly prokázat, že navržené programové řešení je (ve svém rámci) korektní.

Zároveň jednotliví řešitelé dostali za úkol udělat maximum kroků k validaci svých řešení. Tato validace byla realizována částečně u některých dílčích řešení. Validace byla zpravidla prováděna provedením nezávislých výpočtů a srovnáním těchto numerických výpočtů s výstupy simulačního modelu. Takto bylo možné validovat pouze některé aktivity (střelba, pohyb), naopak nebylo možné provádět validaci komplexních sestav (např. dohled).

20.5 Řešitelský tým

Na realizaci případové studie se podílela řada spoluautorů právě proto, aby bylo možné posoudit vhodnost použití metodiky při nasazení vícečlenných řešitelských týmů. Předpokladem bylo, že je možné zapojit do tvorby komplexního simulátoru řadu řešitelů bez potřeby jejich bližšího seznamování s jinými komponenty řešení, než které jsou v jejich kompetenci. Řešitelům prostředí tak například nebylo třeba nutné blíže specifikovat rozhraní HLA. Naopak řešitelé HLA jádra nemuseli mít apriorní znalost toho, jaký konkrétní simulační model bude implementován, nebo v jakém prostředí bude tato implementace probíhat. Řešitelský tým byl složen z odborníků v oblasti informačních technologií, fyziky a počítačové grafiky – dohromady se řešení účastnil jeden profesor, pět inženýrů a sedm bakalářů (započítány jsou i osoby, jejichž řešení byla rovněž předmětem jejich kvalifikačních prací)

20.6 Vyhodnocení nasazení metodiky

Jak bylo zmíněno v úvodu kapitoly, hlavním účelem simulátoru nebylo primárně poskytnutí technického řešení, nýbrž nasazení vlastní metodiky při tvorbě simulátoru netriviálního rozsahu. Poté, co byl v kapitole nastíněn rozsah simulátoru, použité techniky, technologie a zdůrazněn postup v souladu s navrženou metodikou a představen řešitelský tým, je možné provést shrnutí celého řešení s jistou mírou nadhledu.

Hlavním účelem tvorby distribuovaného simulátoru bitvy bylo ověření metodiky, s cílem prokázat, že metodika má určitou přidanou hodnotu. Při posuzování metodiky byla zvažována dvě hlediska – hledisko technického přínosu a hledisko metodické. Hledisko

technického přínosu bylo posouzeno tak, aby bylo možné jednoznačně říci, zda metodika obsahuje nějaké nové, technicky zajímavé řešení, které jí činí originální. Metodické hledisko je pro posuzování velice přirozené a vychází z kapitoly 14 této práce, která se zabývá metodikou.

20.6.1 Hledisko technického přínosu

Identifikace, v čem je hlavní technický přínos myšlenek, na kterých je vlastní metodika postavena je možné najít již v samotné architektuře HLA, z níž myšlenkově vychází dokonce si z ní vypůjčuje názvosloví. Avšak, *Metodis* HLA značně zobecňuje a vypůjčená pravidla a paradigmata umožňuje využít i v jiných architekturách.

Významným technickým přínosem je stanovení koncepce vývoje, která zjednodušuje způsob, jakým je problematika řešena. *Metodis* stanoví, že (na logické úrovni softwaru) mají vzniknout tři vrstvy:

- vrstva pro federát,
- vrstva pro federaci a
- vrstva, které propojí obě předchozí vrstvy.

Toto členění umožňuje uživateli metodiky velmi rychle proniknout k jádru problematiky. I programátor bez kvalifikace v oboru simulace (tj. potenciální uživatel metodiky) je díky tomuto postupu a nastíněnému způsobu segregace schopen zpracovat Simulátor odrážející provoz vybraného systému s decentralizovaným řízením.

Druhým technickým přínosem je možnost vyvinout simulátor v souladu se standardy bez potřeby komplexních znalostí těchto standardů. Například, při použití *Metodis* a provozování celého řešení vyvinutého distribuovaného simulátoru na některé z komerčně dostupných variant RTI je celý simulátor automaticky v souladu se standardem IEEE 1516:2010. Toho lze dosáhnout přesto, že metodika má rozsah méně než čtvrtinový, oproti původnímu standardu.

Dalším technickým přínosem je skutečnost, že oba výše uvedené benefity jsou navíc díky *Metodis* uplatnitelné pro různé technologie a paradigmata (provozní i implementační), to jest metodika ruší původně uvažované omezení na HLA. Zobecnění vybraných pravidel mělo ten důsledek, že vlastní metodika není závislá na konkrétních technologiích, jejichž pořízení či provoz mohou být často finančně velice nákladné.

20.6.2 Metodické hledisko

Metodické hledisko vychází z premisy, že každá metodika, i kdyby neměla žádné originální technické přínosy, přináší přinejmenším výhody použití metodiky (uvedené mimo jiné v tabulce 4). Toto pozorování bylo provedeno na několika případových studiích menšího rozsahu a jedné případové studii většího rozsahu, což lze považovat, pro potřeby této práce, za postačující.

Tabulka 4: Konkrétní plnění obecných metodických přínosů ze strany Metodis[54]

Hledisko	Čím je naplněno
Zaznamenání postupu	Metodika zaznamenává postup budování distribuovaného simulátoru. Tento postup tvoří logicky provázanou posloupnost (sekvence s možností paralelismů) jednotlivých kroků řešení. Každý krok stanovuje vstupní požadavky i cíle a popisuje procesy jak ze vstupů vytvořit výstupy pro daný krok. Toto bylo prokázáno konkrétně tím, že metodika byla využita při více než jedné simulační studii, byla používána týmy i jednotlivci.
Uchování postupu	Metodika a její použití odpovídá podmínkám stability. Přestože se řešitelský tým obměňoval, resp. jednotliví řešitelé jednotlivých (sub)systémů byli rozdílní, byli schopni zpracovat svěřené úkoly. Metodika není závislá na operačním systému, ani použité technologii. Podmínku uchování postupu tímto splňuje.
Zajištění úrovně kvality	Metodika ve své struktuře jasně definuje, jaké jsou vstupy a požadované výstupy. Tam, kde je proces složitý, odkazuje na další normy, nebo standardy, nebo uvádí příklady. Neumožňuje odchylky, které by mohly snížit kvalitu řešení. Celá metodika je postavena tak, aby plnění jednotlivých kroků vedlo ke generální revizi celého řešení (verifikaci a validaci).
Zajištění očekávání	Metodika pro budování simulátorů odrážejících provoz vybraných systémů s decentralizovaným řízením popisuje účel metodiky, vymezuje vlastní metodiku i možnosti jejího nasazení. Již samotný název vzbuzuje očekávání, že výstupem vhodně použité metodiky bude simulátor s určitými vlastnostmi (s těmi, které jsou uvedeny v názvu vlastní metodiky). Úspěšně zrealizovanými případovými studiemi bylo prokázáno, že v případě použití metodiky je očekávání zajištěno a cílů lze dosáhnout.
Hodnocení postupů a jejich úspěšnosti	Metodika je strukturována na jednotlivé procesy. Pro každý proces je možné samostatně vyhodnocovat jeho úspěšnost a kvalitu. Proces lze dále optimalizovat. V případě konkrétní aplikace *Metodis,* na příkladu uvedeném v této kapitole, bylo možné konkrétní význam benefitu přímo sledovat. V situaci, kdy došlo k opakovanému vývoji podobné komponenty. Poté, co jeden řešitel provedl implementaci tanku, došlo k optimalizaci procesu pro vývoj a úprava požadavků. Další autor (který prvního autora fyzicky neznal) podobné komponenty mohl provést implementaci bojového vozidla za výhodnějších podmínek a s vyšším přínosem pro tým.
Možnost týmové spolupráce	Nasazení metodiky podporuje týmovou spolupráci tím, že všechny členy týmu seznamuje s celkovým procesem, vymezuje jim pracovní postupy a mantinely, a je tak možná vertikální (tj. na jednotlivé úkoly v tom rámci, jak je chápe metodika) i horizontální (tj. v rámci jednotlivých úkolů) dělba práce. Možnost spolupráce v širším týmu se prokázala právě v poslední představené případové studii.
Podpora samostatnosti	Použití unifikovaných postupů a procesů, definice rozhraní a jednoznačné určení hranic díla i jednotlivých činností umožňuje zvýšit samostatnost jednotlivých pracovníků. Přestože pracovníci vyvíjí samostatně, díky stanoveným rozhraním jsou přínosní pro celý tým. Vhodné použití metodiky je odstiňuje od problémů, které vychází z jiných částí simulátoru (nežli se oni sami věnují). Jednotliví pracovníci tak nejsou přetěžováni.

Tabulka 4 ilustruje fakt, že vlastní metodika naplňuje všechny charakteristické přínosy tvorby a aplikace metodik. Kromě toho, že vlastní metodika přináší vlastní a originální

[54] Zdroj: Vlastní

postupy či pohledy na budování simulátoru, je možné ji zařadit jako výstup VaVaI – proto je metodika navržena jako certifikovaná.

ZÁVĚR

21 Závěry dizertační práce

Během zpracování dizertační práce bylo dosaženo všech vytyčených cílů. Součástí dizertační práce je příloha, která obsahuje původní metodiku *Metodis* zaměřenou na budování distribuovaných simulátorů odrážejících provoz systémů s decentralizovaným řízením. Během zpracování dizertační práce byla metodika neustále podrobována ověřování. K ověřování sloužily samostatné demonstrátory, nebo simulátory menšího rozsahu – obecně jde tato řešení označit za případové studie. Většina z uvedených případových studií byla řádně prezentována v odborných publikacích. Odkazy na tyto publikace jsou součástí dizertační práce.

Nosným výstupem dizertační práce je původní metodika. Tato metodika je uvedena jako volná příloha, neboť její rozsah je více než 78 normostran textu, což by ve vlastním textu dizertační práče působil nesourodě. Dokument v příloze je označen jako Certifikovaná metodika, neboť v okamžiku obhajoby dizertační práce je odevzdán v podobě, v jaké by bylo možné podat metodiku k certifikaci – tj. včetně tohoto označení. Vlastní metodika je původním výstupem vlastní tvůrčí činnosti a v okamžiku odevzdání dizertační práce splňuje požadavky na klasifikaci jako Ověřená metodika. Případná legální změna z Ověřené metodiky na Certifikovanou metodiku[55] nemění podobu ani formu tvůrčího výstupu, ani oborový přínos autora metodiky.

Případové studie realizované pro ověření metodiky prezentované v rámci dizertační práce prokazují použitelnost metodiky. Řešitelem případových studií je autor metodiky, řešitelský tým, jehož členem byl také řešitel metodiky, nebo třetí osoby.

Provedením rozsáhlé rešerše, tvorbou původní metodiky a ověřením metodiky prostřednictvím několika softwarových děl bylo splněno zadání dizertační práce.

[55] Certifikovaná metodika vyžaduje udělení mezinárodně uznávané certifikace (akreditace) u příslušného odborného certifikačního (akreditačního) orgánu nebo osvědčení příslušného odborného orgánu státní správy, který je věcně odpovědný za oblast, ve které jsou metodika nebo postup uplatňovány.

22 Přehled použité literatury

BALMELLI, L., D. BROWN, M. CANTOR, and M. MOTT, *Model driven systems development*, in The IBM Systems Journal, Vol. 45, No. 3, July/September, 2006: http://www.research.ibm.com/journal/sj/453/balmelli.html

BANKS, J. Discrete-event system simulation. 5th ed. Upper Saddle River: Prentice Hall, c2010, xviii, 622 p. ISBN 0136062121.

BANKS, J.: Principles of Simulation. In BANKS, J., ed.: Handbook of simulation: Principles, Methodology, Advances, Applications, and Practice, 1998. John Wiley & Sons, Inc., New York. ISBN 0-471-13403-1.

BAŠE, Lukáš. *Vytvoření simulačního modelu jednoduchého logistického systému*. Pardubice, 2016. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

BERÁNEK, Jan. *Tvorba simulátoru bojového vozidla za využití hybridního simulačního jádra.* Pardubice, 2016. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

BRUZZONE, A.G., FADDA, P., FANCELLO, G., D'ERRICO, G., BOCCA, E. and MASSEI, M. (2010). *Virtual world and biometrics as strongholds for the development of innovative port interoperable simulators for supporting both training and R&D*, International Journal of Simulation and Process Modeling, Vol. 6, No. 1, pp.89–102.

BUCHALCEVOVÁ, Alena. *Agilní a rigorózní metodiky*. Praha, 2015. Dostupné také z: https://slideplayer.cz/slide/2331811/. Přednáška. Vysoká škola ekonomická.

Centrum transferu technologií a znalostí (CTTZ): Gama (proof-of-concept) [online]. 2015 [cit. 2016-11-26]. Dostupné z: https://vav.upce.cz/gama-proof-of-concept-

CRAIG, D. C. *Extensible Hierarchical Object-Oriented Logic Simulation with an Adaptable Graphical User Interface*. Department of Computer Science Memorial University of Newfoundland, 1996.

DEFINICE DRUHŮ VÝSLEDKŮ: Metodiky hodnocení výzkumných organizací a programů účelové podpory výzkumu, vývoje a inovací. In: *Usnesení vlády č. 837/2017*. 2017, Praha, 26s. (V textu jako Definice výsledků, 2017)

DEPARTMENT OF DEFENCE DOCUMENTATION OF VERFICATIIN, VALIDATION AND ACREDIDATITON FOR MODELS AND SIMULATIONS. Missile Defense Agency. 2008.

Doporučení Rady pro výzkum, vývoj a inovace k certifikovaným metodikám. In: *291. zasedání Rady pro výzkum, vývoj a inovace Úřadu vlády České republiky ze dne 28. 2. 2014*. 2014, Praha, 6s. (v textu jako Doporučení RVVI, 2014).

DUBITZKY, W., K. KUROWSKI a B. SCHOTT. *Large-scale computing*. Hoboken, N.J.: Wiley, 2012. Wiley series on parallel and distributed computing.

Emulátor. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/Emul%C3%A1tor

FIALA, P. JABLONSKÝ, F. MAŇAS, M.: *Vícekriteriální rozhodování*. Vysoká škola ekonomická v Praze, 1994, ISBN 80-7079-748-7.

FIGUEIRA, J. GRECO, S. EHRGOTT, M.: *Multiple Criteria Decision Analysis: State of the Art Surveys (International Series in Operations Research & Management Science)*. Springer, 2004, ISBN 978-0387230672.

FREE SOFTWARE FOUNDATION, Inc. *CERTI - Summary [Savannah]* [online]. 2002 [cit. 2014-04-22]. Dostupné z: http://savannah.nongnu.org/projects/certi

FRIEDMAN, J.H. Stochastic gradient boosting (2002) Computational Statistics and Data Analysis, 38 (4), pp. 367-378. doi: 10.1016/S0167-9473(01)00065-2.

FUJIMOTO, R. M. *Parallel and distributed simulation systems*. New York: J.Wiley, c2000, xvii, 300 s. ISBN 04-711-8383-0.

G. WITMER, Bob a Michael J. SINGER. Measuring Presence in Virtual Environments: A Presence Questionnaire. *U.S. Army Research Institute for the Behavioral and Social Sciences*. Massachusetts Institute of Technology, 1998, **7**(3), 225–240.

GEEKNET, Inc. *SourceForge.net: Open HLA - Project Web Hosting - Open Source Software* [online]. 2005 [cit. 2014-04-22]. Dostupné z: http://ohla.sourceforge.net/

Getting Process Heaps. Microsfot developer resources [online]. Microsoft, 2010 [cit. 2016-02-20]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ee175820%28v=vs.85%29.aspx.

H. KARIMANDA, Y. OMELCHENKO, et al., *A New Approach to Modeling Physical systems: Discrete Event Simulations of Grid-based Models*, Workshop on State-Of-The-Art in Scientific Computing (PARA), 2004.

HAO, F., K. WILSON, R. FUJIMOTO a E. ZEGURA. LOGICAL PROCESS SIZE IN PARALLEL SIMULATIONS. In: *Proceeding of the 1996 Winter Simulation Conference*. College of Computing: Georgia Institute of Technology, Atlanta, 1996, s. 645-652.

HAREN, Van. *TOGAF Version 9.1*. 10. Hertogenbosch, Netherlands.: Van Haren Publishing, 2011. ISBN 9789087536794.

HŘÍDEL, J. a S. KARTÁK, 2013. Web-based simulation in teaching. In: *Modelling and Simulation 2013 - European Simulation and Modelling Conference, ESM 2013*. s. 109-113. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84898481258&partnerID=40&md5=2f108965a144015555678d7f5e6a50ed

HUNTSINGER, 1988. *Simulation Environments and Symbol and Number Processing on Multiprocessors*. 1. San Diego, CA, USA: Society for Computer Simulation. ISBN 978-0911801392.

ISO/IEC 25021:2014. *Software engineering -- Software product Quality Requirements and Evaluation*. CH-1211 Geneva 20: ISO copyright office, 2014. Dostupné také z: https://www.iso.org/standard/64787.html.

ISO/IEC/IEEE 29148:2011. *Systems and software engineering -- Life cycle processes -- Requirements engineering*. CH-1211 Geneva 20: ISO copyright office, 2011. Dostupné také z: https://www.iso.org/standard/45171.html.

JAKEŠ, Martin. *Využití prostředků HLA a DIS k vytvoření počítačové hry pro více hráčů*. Pardubice, 2014. Diplomová práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

K. M. CHANDY and J. MISRA, *Asynchronous Distributed Simulation via a Sequence of Parallel Computations*, Communications of the ACM, vol. 24, 1981.

KARTAK, S. a A. KAVICKA, 2014. WebRTC technology as a solution for a Web-based distributed simulation. In: *26th European Modeling and Simulation Symposium, EMSS 2014*. s. 343-349. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84912139540&partnerID=40&md5=d47bc2188e69288fd48070d634fc770e

KARTAK, S., 2015. Web simulation as a platform for training software application. In: *27th European Modeling and Simulation Symposium, EMSS 2015*. s. 70-78. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84949504463&partnerID=40&md5=e0877ca57169a3218587973efd93a91f

KASAL, Jaroslav. *Tvorba simulátoru bojového vozidla za využití hybridního simulačního jádra.* Pardubice, 2017. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

KAVIČKA, A., V. KLIMA a N. ADAMKO. *Agentovo orientovaná simulácia dopravných uzlov*. 1. vyd. V Žiline: Žilinská univerzita, 2005, 206 s. ISBN 80-8070-477-5.

KINDLER, E. Simulační programovací jazyky. Praha: SNTL, 1980. 280 s.

KNIGHT, P., A. CORDER a R. LIEDEL. Evaluation of Run Time Infrastructure (RTI) Implementations. U.S. Army SMDS, 2001. Dostupné z: https://www.scs.org/confernc/hsc/hsc02/hsc/papers/hsc017.pdf

KORN, Granino A., c2011. *Interactive dynamic-system simulation*. 2nd ed. Boca Raton, FL: CRC Press. Numerical insights, v. 7. ISBN 978-1439836415.

KŘIVÝ, I. a E. KINDLER. *Simulace a modelování: Učební texty Ostravské Univerzity.* Ostrava: Přírodovědecká fakulta Ostravské univerzity 2001. Dostupné z: http://vendulka.zcu.cz/Download/Free/SkriptaKindlerMS.pdf

KUHL, F., DAHMANN, J., WEATHREY, R., Creating Computer Simulation Systems: An Introduction to the High Level Architecture, c2000, Upper Saddle River, NJ; Prentice Hall PTR. ISBN 01-302-2511-8.

KUHL, F., J. DAHMANN a R. WEATHERLY. Inc. *Creating computer simulation systems: an introduction to the high level architecture*. 18 August 2010. Upper Saddle River, NJ: Prentice Hall PTR, c2000, xvi, 212 p. ISBN 01-302-2511-8.

LETIZIA, N., ALESSANDRO, C. and FFRANCISCO, S. (2015) 'Advanced interoperable simulators for training in car terminals',International Journal of Simulation and Process Modelling, Fall, Vol. 10, No. 2, pp.132–143, DOI:10.1504/ijspm.2015.070482.

LETIZIA, N., ALESSANDRO, C., CHARLOS, A. and ALESSANDRO, D. (2014). *Hybrid approach for container terminals performances evaluation and analysis*, International Journal of Simulation and Process Modelling, Fall, Vol. 9, Nos. 1/2, pp.104–112, DOI: 10.1504/ijspm.2014.

LOGAN, B. a G. THEODOROPOULOS. *Dynamic Interest Management in the Distributed Simulation of Agent-Based Systems*. School of Computer Science, University of Birmingham. 2005.

MANLIG, F., 1999. Computer simulation of discrete events. Available from: http://www2.humusoft.cz/www/archived/pub/witness/9910/manlig.htm [accessed 15 July 2014].

MARTINŮ, Jiří a Petr ČERMÁK. *Metodiky vývoje software*. Olomouc: Moravská vysoká škola Olomouc, 2018.

Metodika hodnocení výsledků výzkumných organizací a hodnocení výsledků ukončených programů (platná pro léta 2013 až 2016). In: *Usnesení vlády č. 475/2013, č.250/2014 a č. 605/2015*. 2015, Praha, 60s. (V textu jako Metodika hodnocení, 2013).

Metodika hodnocení výsledků výzkumných organizací a hodnocení výsledků ukončených programů (platná pro léta 2013 až 2016). In: *Usnesení vlády č. 605/2015*. 2015, Praha, 60s. (V textu jako Metodika hodnocení, 2016).

Metodika hodnocení výzkumných organizací a hodnocení programů účelové podpory výzkumu, vývoje a inovací. In: *Usnesení vlády č. 107/2017*. (V textu jako Metodika hodnocení, 2017).

MUIRA, G. *Pushing the Boundaries of Traditional Heritage Policy: maintaining long-term access to multimedia content.* IFLA Journal 33 (2007): 323-326.

NAVRÁTILOVÁ, Gabriela. *Co je metodika a k čemu slouží*. 17. 6. 2013. Centrum podpory, 2013. Dostupné také z: www.nadacesirius.cz/co-je-to-metodika-a-k-cemu-slouzi-z-dvorakova-ops-sirius.pptx

NGUYEN, H. T. et al.: *Fuzzy and Neural Control*. Boca Raton, Chapman & Hall/CRC, 2003, ISBN 1-58488-244-1.

PARK A., R. M. FUJIMOTO, et al., *Conservative Synchronization of Large-scale Network Simulations*, PADS, 2004.

PELÁNEK, R.. Modelování a simulace komplexních systémů: Jak lépe porozumět světu. Vydání první. Brno: Masarykova univerzita, 2011. ISBN 978-80-210-5318-2.

PÉNZEŠ, J. a A. KAVIČKA, 2013. Graphical specification of distributed simulation models. In: *Modelling and Simulation 2013 - European Simulation and Modelling Conference, ESM 2013*. s. 128-133. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84898462324&partnerID=40&md5=8671399e55ee1c704ba9984bac40ed2c

PITCH TECHNOLOGIES AB. *Products - Overview* [online]. 1993 [cit. 2014-04-22]. Dostupné z: http://www.pitch.se/products/products-overview

POPOVICI, Katalin a Pieter J. MOSTERMAN, 2012. *Real-time simulation technologies: principles, methodologies, and applications*. Boca Raton, FL: CRC Press. ISBN 978-1439846650.

QUINN, J., D. MCDERMOTT, I. STIELL, and col. (2006). *Prospective Validation of the San Francisco Syncope Rule to Predict Patients With Serious Outcomes.* Annals of Emergency Medicine (Elsevier) 47 (5): 448–454.

RABELO, L., SALA-DIAKANDA, S., PASTRANA, J., MARIN, M., BHIDE, S., JOLEDO, O., BARDINA, J., 2013. Simulation Modeling of Space Missions Using the High Level Architecture. Available from: http://www.hindawi.com/journals/mse/ [accessed 15 July 2014].

REILLY, D. *Java RMI & CORBA – a comparison of competing technologies*, in Java Coffee Break (cit. 2015-01-20) [online] http://www.javacoffeebreak.com/

RHEINGOLD, H. *The virtual community: homesteading on the electronic frontier*. Rev. ed. Cambridge, Mass.: MIT Press, 2000. ISBN 0262681218.

ROUBTSOVA, Ella, 2016. *Interactive modeling and simulation in business system design*. New York, NY: Springer Berlin Heidelberg. ISBN 978-3-319-15101-4.

Run-time infrastructure (simulation). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, last modified 24 March 2014 [cit. 2014-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Run-time_infrastructure_%28simulation%29

S. FRANKLIN and A. GRAESSER. *Is it an agent, or just a program?: A taxonomy for autonomous agents.* In J. Muller, M. J. Wooldridge, and N. R. Jennings, editors, Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, volume 1193 ofLecture Notes in Artificial Intelligence, pages 21–35. Springer-Verlag, 1997.

SAMOTÁN, Vojtěch. *Vytvoření distribuovaného simulačního modelu jednoduchého logistického systému*. Pardubice, 2014. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

SILLITTI, Alberto, Tullio VERNAZZA a Giancarlo SUCCI. Service Oriented Programming: A New Paradigm of Software Reuse. *7th International Conference, ICSR-7 Austin, TX, USA, April 15-19, 2002 Proceedings*. Austin, Texas: Springer, 2002, 269-280.

SMITH, Donald R., 2000/01/01. Book reviews. *Journal of the American Society for Information Science*. John Wiley, **51**(7), 687-687. DOI: 10.1002/(SICI)1097-4571(2000)51:7687::AID-ASI123.0.CO;2-N. ISSN 0002-8231. Dostupné také z: https://doi.org/10.1002/(SICI)1097-4571(2000)51:7687::AID-ASI123.0.CO;2-N

T. W. MASTAGLIO and R. CALLAHAN. A large-scale complex virtual environment for team training. IEEE Computer, 28(7):49–56, July 1995.

TAMEGAYA Y., *Process and device composite simulation system and simulation method*. Nec Corporation 1997. US 5629877A.

TeamViewer - Vzdálená podpora, vzdálený přístup, servisní služba, online spolupráce a schůzky [online]. 2003 [cit. 2014-07-11]. Dostupné z: https://www.teamviewer.com

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc. *IEEE1012: IEEE Standard for System and Software Verification and Validation.* 25 May 2012, 223str, English.. New York, 2010. ISBN 978-0-7381-7268-2.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc. *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules*. 18 August 2010, 26str, English.. New York, 2010. ISBN 978-0-7381-6251-5.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc. *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specifications*. 18 August 2010, 100s, English. New York, 2010. ISBN 978-0-7381-6249-2.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc. *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Federate Interface Specifications*. 18 August 2010, 364s, English. New York, 2010. ISBN 978-0-7381-6247-8.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc. *IEEE1278: IEEE Standard for Modeling and Simulation (M&S) Distributed interactive simulation*. 18 August 2006, 106str, English.. New York, 2006. ISBN 978-0-7381-6251-5.

THE SIMULATION INTEROPERABILITY STANDARDS ORGANIZATION. Independent Throughput and Latency Benchmarking for the Evaluation of RTI Implementations. *The Simulation Interoperability Standards Organization*. Fall 2001. DOI: SISO-01F-SIW-033.

Thread Pools [online]. Oracle, 1995 [cit. 2016-02-20]. Dostupné z: https://docs.oracle.com/javase/tutorial/essential/concurrency/pools.html.

TING, Pei-yih. *Basic Object Design*. Taiwan, 2004. Dostupné také z: http://squall.cs.ntou.edu.tw/cpp/93spring/CPP-BasicObjectDesign.pdf. National Taiwan Ocean University.

TOGAF 9.1 CZECH GLOSSARY. *TOGAF® 9.1 Translation Glossary: English – Czech*. 9.1. The Open Group Standard, 2013, 43 s. Architecture: Software Engineering Services. Dostupné také z: https://publications.opengroup.org/c13c. ISBN: 1-937218-33-1.

ULRYCH, Z., P. RAŠKA a col., Základní metodika simulační studie při využí paralelní diskrétní simulace. ZČU v Plzni. In: Witness2007. Brno 2007.

van der HOEVEN, J., L. BRAM, and V. REMCO. *Emulation for Digital Preservation in Practice: The Results.* The International Journal of Digital Curation 2.2 (2007): 123-132.

VORACEK, J., J. PENZES a A. KAVICKA, 2014. Non-hla distributed simulation infrastructure. In: *26th European Modeling and Simulation Symposium, EMSS 2014*. s. 319-324. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84912106741&partnerID=40&md5=43e7c3428c8b9afd7d64e3f3cdf443e1

VT MÄK. *HLA RTI – Run Time Infrastructure – MÄK RTI* [online]. [cit. 2014-04-22]. Dostupné z: http://www.mak.com/products/link/mak-rti.html

ZACHOVAL, Tomáš. *Tvorba virtuálního řídícího panelu.* Pardubice, 2017. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Josef Brožek.

23 Přehled vlastních publikačních a tvůrčích výsledků

Přehled vlastních publikačních a tvůrčích výsledků je řazen vzestupně dle data publikace. Řazení abecedně, dle autora by vedlo ke snížení přehlednosti. V souladu se směrnicí bude pro publikační výstupy, které doposud nebyly publikovány (resp. prezentovány) předložen doklad o přijetí příslušného článku k publikaci.

23.1 Publikační výsledky

BROŽEK, J., B.S. ONGGO a A. KAVIČKA, 2014b. High level architecture virtual assistant framework. In: *26th European Modeling and Simulation Symposium, EMSS 2014*. s. 46-55. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84912097947&partnerID=40&md5=a505a7b1a633bb46c24d48a1683006fd

BROŽEK, J., M. JAKEŠ a L. GAGO, 2014. Using tablets in distributed simulation. In: *26th European Modeling and Simulation Symposium, EMSS 2014*. s. 451-456. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84912116953&partnerID=40&md5=00eb8933c42ba55cbec1aa4ccc16cf4e

BROZEK, Josef a Lumir GAGO, 2014c. Using tablets in treatmen. In: *European Industrial Doctoral School Summer Workshop 2014.* [26.-30. May 2014 Pardubice, Czech Republic]. In: s. 48-49. ISBN: 978-80-7395-779-7.

BROZEK, J. a M. JAKES, 2015. Hardware libraries for online control of interactive simulations. In: *27th European Modeling and Simulation Symposium, EMSS 2015*. s. 295-300. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84949507604&partnerID=40&md5=e6363900fa9d6ddcaae5373234cfe902

JAKEŠ, M. a J. BROŽEK, 2015b. Connection of microcontroller and microcomputer to distributed simulation. In: *27th European Modeling and Simulation Symposium, EMSS 2015*. s. 282-288. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84949522022&partnerID=40&md5=0fea1cfe7bdd080e1480ec37b4351e91.

BROZEK, J., V. FIALA, J. FIKEJZ a P. PICH, 2016a. Use of industrial control unit in intelligent homes. In: *ELEKTRO 2016 - 11th International Conference, Proceedings*. s. 489-494. DOI: 10.1109/ELEKTRO.2016.7512124. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84981294900&doi=10.1109%2fELEKTRO.2016.7512124&partnerID=40&md5=0de70f7c818ab15d29b37382b0b33b10

BROZEK, J., D. HAMERNIK, P. VESELY a V. SVOBODA, 2016b. Application of the Montessori method in tercial education of a computer 3D graphics. In: *ELEKTRO 2016 - 11th International Conference, Proceedings*. s. 655-659. DOI: 10.1109/ELEKTRO.2016.7512162. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-

84981314068&doi=10.1109%2fELEKTRO.2016.7512162&partnerID=40&md5=abfdbda59e4539bde6f9105ac41880e4

BROZEK, J., L. BASE, V. FIALA a V. SAMOTAN, 2016c. Simulation of customer flows in a polyclinic: Case study: Case study. In: *ELEKTRO 2016 - 11th International Conference, Proceedings*. s. 396-399. DOI: 10.1109/ELEKTRO.2016.7512106. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84981318354&doi=10.1109%2fELEKTRO.2016.7512106&partnerID=40&md5=618890b69bb277196392634ace0db1a2

SVOBODA, V., J. MAREK a J. BROZEK, 2016d. Software for determining the quality of an exam test. In: *ELEKTRO 2016 - 11th International Conference, Proceedings*. s. 674-679. DOI: 10.1109/ELEKTRO.2016.7512166. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84981333834&doi=10.1109%2fELEKTRO.2016.7512166&partnerID=40&md5=8964d80da53a9d2285a70a72d96ab4d4

BROŽEK, Josef, Jan FIKEJZ a Vojtěch SAMOTÁN, 2016e. Use of HLA During Customer Flow Simulation in a Policlinic. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Rijeka, Croatia: Croatian Society for Information and Communication Technology, Electronics and Microelectronics, s. 1618-1622. ISBN 978-953-233-088-5. Dostupné také z: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7513176

BROŽEK, Josef, Martin JAKEŠ a Václav SVOBODA, 2016f. Innovation of the Campbell Vision Stimulator with the Use of Tablets. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Rijeka, Croatia: Croatian Society for Information and Communication Technology, Electronics and Microelectronics, s. 321-326. ISBN 978-953-233-088-5. Dostupné také z: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7513176

BROŽEK, Josef, Jaroslav MAREK a Václav SVOBODA, 2016g. Software Solution Incorporating the Steganographic Principle for Hiding Pictures within Pictures. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Rijeka, Croatia: Croatian Society for Information and Communication Technology, Electronics and Microelectronics, s. 980-985. ISBN 978-953-233-088-5. Dostupné také z: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7513176

BROŽEK, Josef, Dan HAMERNÍK a Zbyněk KOPECKÝ, 2016h. Creating Assets as a Part of Tertiary Education of Technical Domains. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Rijeka, Croatia: Croatian Society for Information and Communication Technology, Electronics and Microelectronics, s. 973-979. ISBN 978-953-233-088-5. Dostupné také z: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7513176

BENEDIKOVIC, Miroslav, Václav SVOBODA, Josef BROŽEK a Karel ŠOTEK. Smart data flow control as possible future of mobile network. *International Journal of Conceptions on Computing and Information Technology*. 2016, **4**(5), 1-5. ISSN 2345 – 9808.

BROZEK, J. a M. JAKES, 2017. Application of mobile devices within distributed simulation-based decision making. *International Journal of Simulation and Process Modelling*. **12**(1), 16-28. DOI: 10.1504/IJSPM.2017.082782. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015317528&doi=10.1504%2fIJSPM.2017.082782&partnerID=40&md5=da13fcd8ecca71f4adc508ff265c9fc5

FIKEJZ, J. a J. BROŽEK, 2017b. Utilisation of computer simulation for dynamic calculation of train delays. In: *31st Annual European Simulation and Modelling Conference 2017, ESM 2017*. s. 252-257. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85050022400&partnerID=40&md5=6f55bdb8ecad2cd26bb3edc76b9fe8f1

BROZEK, J., M. JAKES, S. KARAMAZOV a D. HAMERNIK, 2017c. Combat vehicle simulator based on HLA prototype concept. In: *Proceedings of the 2016 17th International Conference on Mechatronics - Mechatronika, ME 2016*. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015271386&partnerID=40&md5=684a036482ef2ffb2832b2495a2e99f7

FIKEJZ, J. a J. BROZEK, 2017d. Localization of rolling stock within the railway network model utilizing of web services. In: *International Conference on Multimedia Computing and Systems -Proceedings*. s. 243-247. DOI: 10.1109/ICMCS.2016.7905599. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.085019090351&doi=10.1109%2fICMCS.2016.7905599&partnerID=40&md5=64ecfafbedab97585e89c977bf41a7b0

SVOBODA, Václav, Jaroslav MAREK a Josef BROŽEK, 2017e[in-print]. Steganographic Principle for Transfer Hidden Pictures within Pictures. In: *2nd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BROŽEK, Josef a Václav SVOBODA, 2017[in-print]. Using of the Tablets for Amplyopia Treatment. In: *2nd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BROŽEK, Josef a Zbyněk KOPECKÝ, 2017f[in-print]. Assets as Part of Tertiary Technical Education. In: *2nd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BROŽEK, Josef a Jan FIKEJZ, 2017g[in-print]. HLA Simulation of Customer Flow in a Polyclinic. In: *2nd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

HEMNANI, Karan, Dan HAMERNÍK, Josef BROZEK a Karel SOTEK, 2018[in-print]. Comparison of Different Self-developed Simulation Driver Seats with Two, Four and Six Servos. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BROZEK, Josef, 2018b[in-print]. Genetic Algorithms and their Ethical Degeneration in Simulated Environment. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BENEDIKOVIC, Miroslav, Dan HAMERNIK a Josef BROZEK, 2018c[in-print]. Arduino Wrapper for Game Engine-based Simulator Output. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

PROVAZNÍK, Josef, Zbyněk KOPECKÝ, Josef BROŽEK, Karel ŠOTEK, Monika BROŽKOVÁ, Simeon KARAMAZOV a Hana JANEČKOVÁ, 2018d[in-print]. Software Solution Incorporating Activation Congnitive Memory Portion in Early Stages of Alzhaimer's Disease. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

DHOLAKIYA, Nishaben S., Jan KUBÍK, Josef BROŽEK a Karel ŠOTEK, 2018e[in-print]. Unity3D Game Engine Applied to Chemical Safety Education. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

VAN DER HEIJDEN, Tim, Dan HAMERNÍK a Josef BROŽEK, 2018f[in-print]. Use of Game Engines and VR in Industry and Modern Education. In: *3rd International Conference on: Applied Physics, System Science and Computers*. Dubrovnik: Springer Verlag.

BEDNÁŘ, R. a J. BROZEK, 2019. *Neural interface: the potential of using cheap EEG devices for scientific purposes: the potential of using cheap EEG devices for scientific purposes*. **489**, s. -132, 127 s.. DOI: 10.1007/978-3-319-75605-9_18. Dostupné také z: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85049311763&doi=10.1007%2f978-3-319-75605-9_18&partnerID=40&md5=d4ede9147936844a762d59e9cd9a0be9.

DHOLAKIYA, Nishaben Desai, Milos FERJENČÍK, Josef BROŽEK, Zdenek JALOVY a Jan KUBÍK, 2019b. Don't Worry!! Laboratory Safety Is Fun Do, Let's Play a Mobile Game ☺. In: *AIChE Spring Meeting and Global Congress on Process Safety*. New Orleans, LA: American Institute of Chemical Engineers, s. 144-153. ISBN 978-0-8169-1109-7. Dostupné také z: https://www.aiche.org/conferences/aiche-spring-meeting-and-global-congress-on-process-safety/2019/proceeding/paper/144b-dont-worry-laboratory-safety-fun-do-lets-play-mobile-game

23.2 Ostatní tvůrčí výsledky

BROZEK, Josef. Demonstation system for aplication of HLA (SW). 2013.

BROZEK, Josef a Lumir Gago: System for rehabilitation of Amplyopia. 2013.

BROZEK, Josef, ONGGO, Stephan Bhakti, KAVICKA, Antonin. High level architecture virtual assistant framework (SW). 2014.

BROZEK, Josef, JAKES, Martin, GAGO, Lumir. Using tablets in distributed simulations (demonstrator - SW). 2014.

BROZEK, Josef. High Level Architecture Transport demonstrator, 2015.

BROZEK, Josef and Martin JAKES, Microcontroller operation connecter for HLA, 2015.

BROZEK, Josef and Martin JAKES. Systém pro demonstraci principů pro léčbu očních vad. 2016.

BROZEK, Josef. Šifrovací software pro časově omezené aktivace software. 2016.

BROZEK, Josef, SVOBODA, Vaclav, JAKES, Martin. Systém pro správu licenčních klíčů pro časově omezené licence určené pro mobilní zařízení. 2016.

BROZEK, Josef a Martin JAKES. HLA – Microcontroller connector, 2016.

BROZEK, Josef. HLA – serial connector, 2016.

BROZEK, Josef. HLA – online connector, 2016.

BROZEK, Josef. HLA – USB connector, 2016.

STEPANEK, Martin, BROZEK Josef a SVOBODA Vaclav. Online nástroj pro vyhodnocení statistických dat. 2016.

GAGO Lumir, JAKES Martin a Josef BROZEK. HLA simulator JAVA applet: Simulation of simple multiplayer figth. 2016.

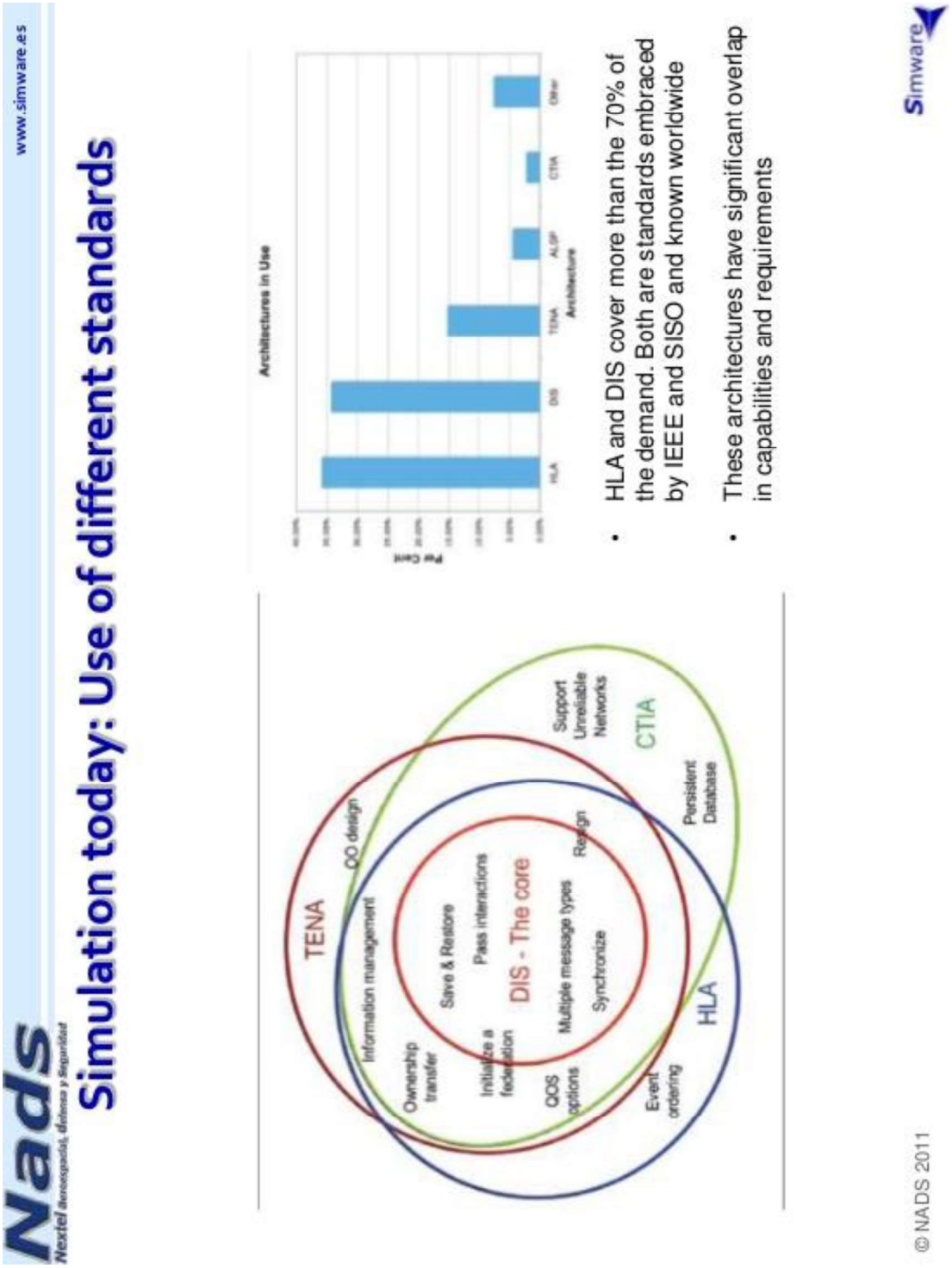
JAKES Martin, HAMERNIK Dan a BROZEK Josef. HLA. Combat whelicular simulator. 2017.

24 Přílohy práce

Práce obsahuje, kromě příloh zařazených v této kapitole také níže uvedené přílohy:

1. *Metodis* – výsledek vlastní vědecké a vývojové činnosti a nosný výstup dizertační práce. Příloha samostatně vázaná.
2. Souhrn práce v anglickém jazyce (tzv. Teze), jako povinná přílohu dle požadavků Studijního a zkušebního řádu Univerzity Pardubice ze dne 20. prosince 201, článku 15, odstavce 5 Studijního a zkušebního řádu Univerzity Pardubice ze dne 20. prosince 2016. Příloha samostatně vázaná.
3. Vložené médium s textem práce, tezí a dalších souvisejících dokumentů předkládaných studentem s dizertační práci pro zahájení obhajoby dizertační práce dle článku 16 odstavce 2 Studijního a zkušebního řádu Univerzity Pardubice ze dne 20. prosince 2016.

24.1 Technologie využívané k tvorbě simulátorů



24.2 **Schéma standardu ISO**