Projekt: Dizertační práce, Univerzita Pardubice, Studentská 95, 532 10 Pardubice
Interní číslo projektu: UPA-FEI-BR-2017-CM
Řešitel: Univerzita Pardubice, studentské dílo dle zákona
Výsledek projektu: Certifikovaná metodika

Metodika budování distribuovaných simulátorů odrážejících provoz systémů s decentralizovaným řízením

Autor: Josef Brožek
Datum: 31. července 2017

Obsah

Knihovní údaje

Název

Metodika budování distribuovaných simulátorů odrážejících provoz systémů s decentralizovaným řízením

Autor

Josef Brožek

Anotace

Metodika definuje metodický postup, který je třeba realizovat pro tvorbu distribuovaných simulátorů. Simulátory, které budou vybudovány dle metodiky a softwarově implementovány dle HLA, nebo DIS se stanou kompatibilní s těmito standardy. Metodika je zaměřena na tvorbu simulátorů pro takové systémy, které jsou charakteristické decentralizovaným řízením. Během budování a následného testování metodiky však bylo zjištěno, že metodika vyhovuje také pro ostatní systémy. Názvosloví metodiky často vychází z HLA, kterou měla metodika původně rozšířit, avšak po zobecnění tvoří vzhledem k HLA komplementární část.

Klíčová slova

HLA, DIS, simulace, simulátor, budování, analýza, testování, software, emulace, online simulace, decentralizované řízení

Klasifikace dle CZ-NACE

28.99 Výroba ostatních strojů pro speciální účely j. n.
71.20 Technické zkoušky a analýzy
72.19 Ostatní výzkum a vývoj v oblasti přírodních, technických věd

Klasifikace oboru řešení

Hlavní obor řešení:	IN	Informatika
Vedlejší obor řešení:	BC	Teorie a systémy řízení

Cíl metodiky

Metodika pro budování simulátorů odrážejících provoz systémů s decentralizovaným řízením (zkráceně také Metodis) usnadňuje vývoj simulátorů typu specifikovaných samotným názvem metodiky. Jejím cílem je vhodným výběrem existujících softwarových paradigmat a jejich aplikací do oblasti vývoje simulátorů unifikovat, strukturovat a celkově usnadnit proces vývoje distribuovaného simulátoru. Prostřednictvím specifických technik, které jsou využitelné pro budování simulátorů, zpřístupní vývoj simulátorů i odborným pracovníkům z oblasti informačních technologií, kteří nejsou úzce specializováni v oblastech počítačové simulace.

Ohraničení metodiky

Metodika umožní především vývoj vlastních simulátorů – tedy počítačových programů určených k tvorbě modelů dynamických systémů určených ke zkoumání příslušného simulovaného systému systémem simulujícím (prostřednictvím experimentální metody – vlastní simulace). Určení obecného vhodného vztahu mezi simulujícím a simulovaným systémem a způsob abstrakce jsou součástí této metodiky, avšak implicitně předpokládají požadované znalosti z oblasti analýzy. Určení konkrétních vztahů mezi simulovaným a simulujícím systémem je vždy věcí konkrétní implementace, a proto tyto vztahy v rámci metodiky není možné obecně popsat.

Simulované systémy, na které je cílena tato metodika, jsou charakterizovány absencí centralizovaného řízení. Metodika je však využitelná i pro systémy s centralizovaným řízením, neboť systémy s centralizovaným řízením jsou zvláštním případem systémů s decentralizovaným řízením. Lze říci, že systémy s decentralizovaným řízením (resp. bez požadavku centralizovaného řízení) jsou obecnější než systémy, kde existují požadavky na centralizovaný řídící prvek – provést restrikci obecnějších zásad je tedy možné a metodika je tam uplatnitelná i na simulované systémy s centralizovaným řízením – i když pro ně není primárně určena.

Metodika se omezuje na tvorbu distribuovaných simulátorů. Vzhledem k charakteru distribuovaných řešení však metodika připouští vytvořit simulátor realizovaný prostřednictvím jediného výpočetního uzlu. Distribuovanost není metodikou chápána jako stav, pro který je nutnou podmínkou využití více výpočetních uzlů. Vlastnost distribuovanosti je možné splnit i za předpokladu běhu více nezávislých procesů (zpravidla s nezávislou časovou bází) na jednom výpočetním uzlu, přičemž v rámci dílčích procesů dochází k synchronizaci. Stav, kdy je využíván právě jeden výpočetní uzel je restrikcí standardního distribuovaného modelu. Vymezení počtu výpočetních uzlů je charakteristické pro každý simulující systém, a přestože metodika připouští využití pouze jednoho výpočetního uzlu, požadavky vlastní metodiky často mohou vést k potřebě využití více výpočetních uzlů – především v případě interaktivních simulátorů, kde jsou požadovány přímé uživatelské vstupy.

Metodika připouští práci se simulací jako předem prototypovaným modelem s jasně určeným stavovým prostorem, deterministickými, nebo stochastickými procesy (resp. událostmi). Metodika však zároveň připouští možnost protypování během simulačního výpočtu, resp. metodika je aplikovatelná v případě, že dochází k uživatelským změnám stavového prostoru simulace během simulačního výpočtu. Metodiku jde tedy nasadit také tam, kde jsou předpokládány interaktivní simulátory, včetně interaktivních simulátorů pro online systémy.

Metodika připouští jakoukoli práci s časem splňující podmínku kausality. Implicitně není předpokládán vztah reálného času, času simulujícího systému a času simulovaného systému (resp. vztah definitivního času a času simulovaného systému téměř vždy existuje, ale jeho zkoumání není předmětem této metodiky a ani obecně simulace jako vědecké metody). V některých případech – resp. pro některé konkrétní typy simulujících systémů – může metodika pravidlo vztahu definitivního času a času simulovaného systému omezit. K tomuto omezení dochází ve speciálním případu interaktivních simulací.
Metodika připouští libovolnou aplikaci vizualizační vrstvy – 2D vizualizaci, 3D vizualizaci i vizualizaci prostřednictvím externích zařízení – emulaci prostředí, nebo využití virtuální reality. Vizualizace je však v rámci metodiky vždy uvažována jako separátní proces, nebo v případě interaktivního simulátoru, událostně orientovaný proces.
Metodika připouští typově libovolné vstupní proudy s uplatněním stochastických i deterministických generátorů vstupních hodnot, stejně jako vstupní proudy parametrizované online, semioffline a offline systémy pro sběr externích dat, anebo parametrizované prostřednictvím uživatelských zásahů.

Klasifikace simulátorů, pro které je metodika určena

Přestože část Ohraničení metodiky vymezuje simulátory, na které je cíleno velmi přesně, může být příliš komplexní pro snadné pochopení. Proto je zde explicitně uvedeno, pro které třídy simulátorů je Metodika výhodná. Je nutné dodat, že metodika je použitelná i pro další třídy simulátorů, avšak není pro ně primárně určena.
Dle základní klasifikace simulátorů je metodika vhodná pro stochastické i deterministické simulátory, simulátory se spojitým, diskrétním, nebo kombinovaným časem, pro simulátory místní, distribuované, interaktivní (HIL i MIL) i neinteraktivní. Přestože metodika umožňuje tvorbu simulátorů třídy HIL, rozsah metodiky a náročnost implementace nemusí být výhradně pro HIL systémy vhodná.

Předpoklady pro aplikaci metodiky

Metodiku lze použít v případě:

- tvorby nového simulátoru,
- požadavku na rozšíření již existujícího simulátoru vybudovaného dle této metodiky,
- požadavku na rozšíření již existujícího simulátoru vybudovaného na některé z technologií, které jsou kompatibilní s touto metodikou, nebo
- adaptace již existujícího simulátoru na některou z technologií, které jsou kompatibilní s touto metodikou.

Faktickými předpoklady pro aplikaci metodiky jsou:

- dostupnost informací o simulovaném systému,
- možnost kvantifikace charakteristik simulovaného systému a schopnost matematického popisu procesů simulovaného systému (např. jak požaduje disciplína Teorie hromadné obsluhy), pro interaktivní simulátory navíc v kombinaci s definicí přípustných bodových procesů reprezentujících interakce,

- dostupnost informací, případně i dat, pro provedení validace,
- dostupnost vhodných nástrojů pro tvorbu simulátorů (nástroje pro rychlé prototypování simulačních modelů, nebo programovací jazyky),
- dostupnost vhodných běhových nástrojů pro provoz simulátorů (tj. dostupná ICT infrastruktura),
- možnost upravovat simulující systémy, případně jejich rozhraní, případně tvořit wrappery,
- dostupnost dalšího hardwaru, který má být použit při implementaci, případně tvorbě emulovaného prostředí.

Kompatibilita metodiky

Tvorba metodiky byla podřízena implementačním prostředím, které jsou standardně používány, především jde o HLA, DIS, TENA, ALSP a CTIA. Dostupnost civilních specifikací standardů HLA a DIS umožnilo specifikovat tyto technologie jako zcela kompatibilní s touto metodikou. Obecně je možné říci, že výslednou kompatibilitu metodiky lze určit podle níže uvedené tabulky. Tabulka uvádí jednotlivé dopředné i zpětné kompatibility metodiky. Tedy, v případě, že je analýza a programování provedeno v souladu s HLA a řešení je řádně dokumentováno, lze říci, že celé řešení je navrženo také v souladu s *Metodis* a je využitelné jako komplement pro další nasazení metodiky při systémové integraci. Totéž platí pro návrh dle architektury DIS. Naopak, zpětně platí, že pokud byl simulátor navržen dle této metodiky a programován v souladu s HLA, je s HLA plně kompatibilní. V případě návrhu simulátoru dle metodiky a jeho implementace dle agentově orientovaných paradigmat je následně celé řešení kompatibilní s předpisem ALSP.

Analýza a architektura	**Programování**	**Kompatibilita s**
HLA	HLA	*Metodis*
DIS	DIS	*Metodis*
Metodis	HLA	HLA
Metodis	Agenty	ALSP

Související normy, doporučení a standardy

Metodis vznikla v prostředí, ve kterém existující normy, standardy a doporučení. Tyto existující dokumenty byly při tvorbě zohledněny. Je tak využíváno existujícího názvosloví, principů, postupů všude tam, kde je to možné. Konkrétní výčet zohledněných standardů uveden v následující tabulce:

Název standardů	Označení standardu
Software verification and validation	IEEE 1012
Software quality assurance	IEEE 730
Software test documentation	IEEE 829
Software requirements specification	IEEE 830
Software design description	IEEE 1016
Software project management	IEEE 1058
Software user documentation	IEEE 1063

Terminologie a zkratky

Terminologie použitá v této metodice vychází z terminologie definované ve standardu TOGAF (2011) a oficiálního českého slovníku (TOGAF 9.1. Czech glossary, 2013), případně z odborné literatury uvedené přímo u jednotlivých termínů:

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 obecním 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 platformě závislý a tak tuto 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	Zkratkové slovo pro označení této metodiky.
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 obsah interaktivního simulátoru, který není dán stavovým prostorem a není klíčový pro fungování jednotlivých procesů. 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. 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, jenž by bez wrapperu nebyly použitelné/dosažitelné.
XML	Extensible Markup Language. Značkovací programovací jazyk pro uchování konzistentních dat nezávisle na počítačové platformě.

Jak s metodikou pracovat

Metodika je obecným návodem pro implementaci simulátoru. Tento návod je však nutné adaptovat na podmínky konkrétní instituce a konkrétního simulačního modelu. Jednotlivé simulátory se významně liší rozsahem, vnitřním uspořádáním, mají odlišné úrovně detailů (mikro, mezo a makro) odlišné možnosti popisu vnitřních procesů. Adaptaci metodiky lze provést před vlastní přípravou implementace nebo jako jednu z prvních částí.
Pokud je implementace prováděna v organizaci řízené zvláštními standardy a požadavky (např. z hlediska bezpečnosti, způsobu nakládání s daty, systémem řízení kvality aj.), je vhodné metodiku uzpůsobit zavedeným standardům a jim i implementaci podřídit. V opačném případě je vhodné metodiku použít jako výchozí bod pro vytvoření plánu implementace a řízení jejího průběhu.
Pro užití metodiky je nutné splňovat kritéria stanovená v části Předpoklady pro aplikaci metodiky, případně podmínky dalších zvláštních požadavků vývojářů a zadavatelů. Metodika je navržena tak, aby byla transparentně rozšiřitelná o další požadavky především v oblasti zabezpečení – například o šifrované přenosy, nebo o přenosy v rámci QoS nebo po vyhrazených linkách.
Metodiku je možné použít komplexně, nebo využít jednotlivé samostatné fáze, a to vždy, když je možné dosáhnout požadavků na jednotlivé vstupy.

Doporučená forma řízení

V případě, že je metodika používána jako komplet, i v případě, že jsou požívány jen dílčí části metodiky, je doporučeno používat inkrementální iterativní metodiky vývoje, v ideálním případě v agilní formě. Metodika byla navržena tak, aby bylo výhodné využití agilních technik. Metodika byla testována a řízena holistickou, flexibilní technikou odpovídající metodice SCRUM.[1]
Charakteristické je řízení v akademickém prostředí, kde není možné dosáhnout symetrických požadavků a stand up v délce jednoho dne. Akademické prostředí, kde jsou ve vývoji zapojeni akademičtí pracovníci, místo pracovníků vědeckých, je výhodnější použít jiný systém řízení projektu. Agilní metodiky nejsou pro takovéto týmy výhodné (vyjma agilní techniky Událostně řízeného vývoje) a bude třeba použít rigorózní metody vývoje. Výhodné se v takové situaci jeví aplikace RUP nebo OOSP. Pro potřeby této metodiky se důrazně nedoporučuje využívat ASD. ASD velmi často degraduje a její uplatnění v akademickém prostředí je jen velmi obtížně možné. Pro potřeby vytvoření uceleného přehledu o metodikách je možné využít například materiál doc. Buchalcevové (2015) z Vysoké školy ekonomické v Praze.

Struktura 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

[1] Pro středně velký projekt tak lze doporučit aplikaci SCRUM s délkou sprintu 14 dní, stand up je stanoven na 1 den. Doporučením stanoveným Product Ownerem je odpovědný analytik, SCRUM Masterem pak vedoucí celého projektu. Strukturu vývojového týmu je tak vhodné vytvořit jako vyčleněnou s vyčleněním právě Product Ownera. Terminologie dostupná na: https://www.scrumalliance.org/learn-about-scrum

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

- **Metodický postup** zahrnuje 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** představují strukturální části bloků. Krokování je provedeno pro bloky, které jsou samy svým rozsahem netriviální. Účelem dalšího rozpadu bloků na kroky je zvýšení přehlednosti konkrétního bloku.

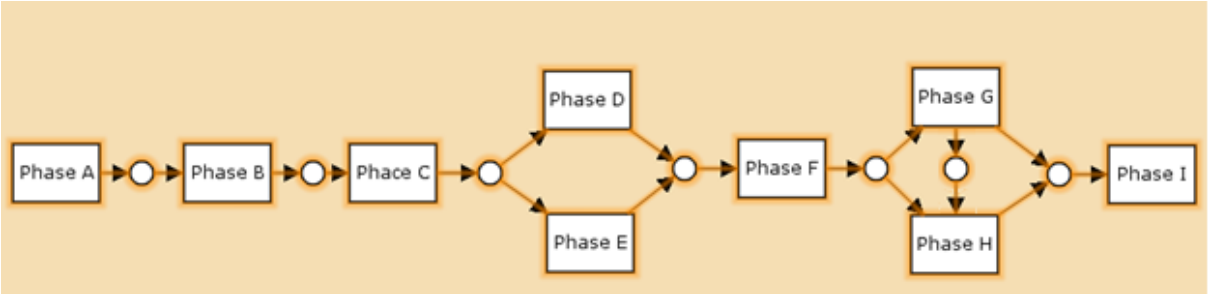
Konkrétní fáze metodiky

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 příslušných blocích. 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).



Na obrázku je zobrazena grafická reprezentace metodického postupu, kde obdélníky značí jednotlivé fáze metodiky ve formě síťového grafu. Vlastní fáze jsou reprezentovány vrcholy grafu, jejichž návaznosti jsou značeny orientovanými hranami a graf obsahuje také milníky (které mohou sloužit jako body synchronizace). Jednotlivé fáze označeny klíčovým slovem Phase a identifikátorem příslušné fáze. Body synchronizace (označeny kružnicemi) jsou klíčové především v případě větvení (paralelizace) fází. V případě změn je nutné vždy pracovat od milníku bezprostředně následujícím po fázi změn. Pokud by například na základě problémové implementace (fáze D) muselo dojít k architektonickým změnám (fáze C), je znovu nutné provést realizaci všech fází následujících po milníku, který následuje po fázi C. 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.
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.

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. Tedy, 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 k dosažení cíle. Při vysoké složitosti je blok dále rozdělen do kroků.

- **Vstupy** pro jednoduché a transparentní napojení jednotlivých fází uvádějí soupis požadovaných vstupů. Vstupy je možné nahradit jejich substituty.
- **Postup** vysvětluje konkrétní činnosti, které vedou k zpracování vstupů na 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.

Zájmy a zainteresované osoby

Při přípravě jednotlivých fází implementace je nutné zohledňovat zájmy zainteresovaných subjektů. V rámci definice vize projektu nebo později v době jeho přípravy je nutné dotčené zájmy a jednotlivé zainteresované subjekty identifikovat.

Konkrétními příklady zájmů ve vztahu k Metodice:

- ekonomika/finanční hledisko,
- legislativa a metodika (zahrnuje i různá vnitřní pravidla instituce),
- lidské zdroje iniciované na řešení projektu,
- bezpečnost (data, přístup k nim, provádění změn),
- podoba rozhraní,
- vyhodnocovací kritéria (na úrovni dílčích simulačních běhů),
- další uživatelská kritéria.

Konkrétní příklady zainteresovaných subjektů:

- objednatel simulátoru,
- instituce provozující simulátor,
- uživatel simulátoru,
- instituce, která určuje konkrétní uživatele simulátoru,
- autorská instituce simulátoru,
- servisní instituce simulátoru,
- příslušné instituce státní správy,
- další partnerské instituce pro instituci provozující simulátor,
- ředitel instituce,
- pracovník IT.

Klíčové subjekty

Popis klíčových zainteresovaných subjektů uvádí výčet projektových rolí tak, jak je chápe metodika. Jednotlivé role mohou splývat – tedy, jedna osoba může mít více rolí. Stejně tak je přípustné, aby jednotlivé role byly realizovány samostatnými týmy – v takovém případě je v metodice označena konkrétní role, avšak zamýšlen je celý tým/celá instituce, která dané roli odpovídá. Metodika připouští kombinaci rozkladu role na týmy a splývání rolí. Pojem osoba v rámci metodiky označuje osobu, nebo sdružení osob, které mohou být právnické, nebo fyzické.

Řešitelský tým

Řešitelským týmem se v rámci metodiky rozumí osoba, nebo sdružení osob, které mezi sebou mají legalizovány vztahy, proto se dalším vymezením metodika nemusí zabývat. Vymezenými vztahy je myšleno především určení klíčových, dílčích odpovědností, vymezení autorského podílu na výsledném díle a způsob distribuce odměny za dílo.
Řešitelský tým je pro potřeby metodiky vymezen jako osoba, která je uživatelem této metodiky a jejímž cílem je vytvořit vlastní simulátor jako vlastní autorské dílo. Tvorbu simulátoru autorský tým vytváří svépomocí, nebo prostřednictvím subdodávek. V případě realizace subdodávkami metodika zapouzdřuje subdodavatele jako jakéhokoli jiného interního řešitele.

Objednatel

Objednatel je v rámci metodiky osoba, která je primárně odpovědná za přebírání jednotlivých dílčích, nebo souhrnných výstupů. V praxi velmi často splyne osoba objednatele s osobou zadavatele nebo investora. Objednatel je osobou, která je klíčová pro akceptaci celého předkládaného řešení a předpokládá se u něj odborná způsobilost pro revizi akceptačních kritérií a kompetence rozlišení chyb implicitních (stav, který je v rozporu s objednávkou) a chyb explicitních (stav, který fakticky existuje, není v souladu se simulovaným systémem, avšak je v souladu s objednávkou).

Zadavatel

Zadavatel je v rámci metodiky osoba, která je primárně odpovědná za zadání problému (resp. Spojení s aplikační doménou řešeného simulátoru). K zadání systému může přistoupit relativně volně (může jít například o uživatele dispečerského stanoviště – dispečera – v situaci, kdy je úkolem vytvořit virtuální dispečerské pracoviště). Od osoby zadavatele nejsou očekávány znalosti z oboru simulací, návrhu softwaru, nebo informačních technologií obecně. Naopak se u zadavatele předpokládá velmi hluboká znalost problematiky, která je předmětem vlastního simulátoru.
Zadavatel je zároveň osoba, která dodá klíčová data pro nastavení simulátoru, nebo zajistí prostředky proto, aby si tyto údaje mohl zajistit řešitelský tým. Těmito klíčovými údaji pro nastavení simulátoru jsou myšleny časové údaje pro procesy, nastavení vstupních a výstupních proudů atp.

Investor

Investorem se pro potřeby metodiky rozumí osoba, která je odpovědná za financování celého řešení. Pokud osoba investora nesplývá s osobou zadavatele, objednatele, provozovatele, nebo uživatele, je třeba přihlížet k této osobě jen v několika málo specifikacích. Jde především o respektování zvláštních finančních požadavků investora (např. způsob fakturace, způsob předávání zdanitelného plnění). Pokud investor není přímým účastníkem celého projektu (tj. pokud neexistuje přímý vztah mezi investorem a řešitelským týmem, nýbrž existuje vztah například pouze mezi objednatelem a investorem), je možné od zvláštních požadavků investora abstrahovat a při řešení projektu vůbec neuvažovat.

Provozovatel simulátoru

Provozovatelem simulátoru se rozumí taková osoba, která bude vlastní simulační řešení fakticky provozovat. Provozovatel se zpravidla stává odpovědnou osobou za chod simulátoru a jeho bezproblémový provoz. Provozovatel zpravidla odpovídá za úpravu jednotlivých scénářů a jejich provoz. Pokud systém obsahuje systém After Action Rewiev, tak i jeho zprovoznění a případně účelový rollback je součástí kompetencí a odpovědností provozovatele simulátoru. V případě, že simulátor obsahuje prostředí pro uživatelské vstupy (emulaci prostředí, počítačový vstup, virtuální realitu), předpokládá se, že provozovatel simulátoru je odpovědný také za tuto hardwarovou infrastrukturu. Základní vlastnosti simulátoru dané jeho programovou implementací jsou pro provozovatele simulátoru neadaptovatelné.

Servisní instituce simulátoru

Servisní institucí simulátoru se rozumí taková osoba, která má k dispozici zdrojové kódy simulátoru a je schopna zasáhnout do běhu simulátoru především v situaci, kdy pro úpravu simulátoru nestačí pouhá editace scénáře. Servisní instituce je také odpovědná za aktualizace řešení a udržování řešení v souladu s dalšími technologiemi, které mohou probíhat technologickou evolucí. Servisní instituce je také odpovědná za technické závady na softwaru.

Uživatel simulátoru

Uživatel simulátoru je taková osoba, která bude simulátor ovládat. Pokud je v rámci metodiky navržen interaktivní simulátor, je uživatelem taková osoba, která dynamicky parametrizuje simulační výpočet během jeho běhu. Pokud jde o simulátor bez interaktivních vstupů, pak uživatel vytváří parametrizaci před spuštěním simulací, případně intepretaci simulačních výstupů.

Fáze A – Přípravná fáze

Cíle

Cíle fáze A jsou:

- legální zahájení realizace projektu budování simulátoru,
- vymezení oboru zájmu,
- stanovení základních kritérií,
- příprava analýzy,
- příprava akceptačních kritérií.

Způsob dosažení cíle

Tato fáze se zaměřuje na obslužná témata nezbytná k úspěšnému řešení projektu. Po ukončení fáze bude k dispozici jasné vymezení projektu, legální podklady k realizaci projektu a seznam akceptačních kritérií – tedy veškerých základních podkladů, které umožní vyhodnocení úspěšné/neúspěšné realizace celého díla.
Fáze má dvě nosné části – legalizaci projektu a základní vymezení projektu. V praktické situaci často nelze tyto dva procesy provádět sekvenčně, nýbrž dochází k určitému paralelismu, který je způsoben tím, že oba výstupy jsou na sobě volně závislé. Komplexní analýzu není možné vytvářet bez znalosti simulovaného systému a požadavku objednatele. Projekt však zároveň nejde legalizovat, pokud nejde vymezit jasnou problematiku – součástí legalizace musí být zároveň rozpočet.
Metody použité během fáze jsou především rozhovor a analýza (resp. rozbor). V rámci metodiky je připuštěno a předpokládáno, že v této fázi budou využity služby externích institucí např. právních kanceláří.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi A.

Externí dokumenty

- Legislativa související s legalizací příslušné zakázky,
- právní informace (např. výpisy z obchodního rejstříku) které specifikují osoby.

Interní dokumenty

- Interní metodické pokyny pro zpracování objednávek.

Architektonické dokumenty

- Slovní popis objektu zkoumání a simulovaného systému.

Postup

Základní analýza simulovaného systému

Simulovaný systém by měl být popsán formou volného slohového útvaru jako popis, odborný popis, nebo referát. Simulovaný systém by měl být popsán celý, stejně jako kontext ostatních systémů. Tento dokument je povinným vstupem a je předpokládán objednatelem.
Na základě dokumentu provede řešitelský tým analýzu podstatných jmen a sloves (Ting 2004). Podstatná jména a slovesa je třeba specifikovat (určit jim funkční rámec a zařadit do budoucího systému). Na základě analýzy vytvoří řešitelský tým Předběžnou analýzu systému, která obsahuje především:

- výčet podstatných jmen specifikovaných jako mobilní entity,
- výčet podstatných jmen specifikovaných jako stabilní entity,
- výčet podstatných jmen specifikovaných jako atributy entit,
- výčet sloves specifikovaných jako vstupní a výstupní datové proudy (resp. rozhraní pro výstupní proudy/streamy),
- výčet sloves specifikovaných jako vnitřní procesy systému a
- výčet podstatných jmen a sloves tvořící ohraničení systému.

Dokument, který na základě výše uvedeného vznikne, to jest *Předběžná analýza systému* může obsahovat řadu nevyjasněných okolností – ve vztahu k okolí systému a ve vztahu k vnitřním procesům systému. Právě analýza podstatných jmen a sloves umožní další diskusi se zadavatelem a upřesnění systému.

Vymezení kompetencí v rámci řešení projektu

Pro úspěšnou realizaci simulátoru je klíčová korektní identifikace jednotlivých klíčových subjektů. Řešitelské strany, které jsou specifikovány v kapitole Klíčové subjekty, musí být identifikovány následovně:

- Zadavatel je osoba, která vydává zadání simulovaného systému, určuje kritéria přijatelnosti a následně přebírá dokončené dílo.
- Instituce provozující simulátor je instituce, u které bude později simulátor nasazen. Taková instituce bude v budoucnu odpovědná za provoz simulátoru. V případě, že je dle metodiky budován pouze simulátor, který bude sloužit jako prostředek pro získání výstupu ve formě simulační studie, může být touto institucí přímo instituce realizačního týmu.
- Uživatele simulátoru je třeba rozlišit dle účelu a zaměření simulátoru. V případě, že simulátor bude pouze prostředkem pro realizaci simulační studie, splývá role uživatele simulátoru s rolí realizačního týmu. Pokud půjde o simulátor parametrizovaný před během simulátoru, nebo simulátor typu trenažér (parametrizovaný dynamicky během simulačního výpočtu), je třeba najít charakteristického uživatele, který může svými názory pomoci kvalitnímu vypracování celého řešení.

Upřesnění vymezení simulovaného systému

Je třeba realizovat schůzku mezi zadavatelem a řešitelským týmem. Při této schůzce dojde k vyjasnění klíčových okolností, které umožní upřesnit vymezení. Vymezení je třeba dokončit především v rámci:

- časové závislosti na okolí systému – především hledání periodických okolností mající vliv na systém (např. dny v týdnu, státní svátky, letní prázdniny),
- základní časové intervaly – především elementární časový krok z pohledu uživatele (informace o časech sběru dat z online systémů, případně online zařízení),
- časové závislosti vnitřní systémové latence – především z pohledu bezpečnosti simulovaného softwaru (*ISO/IEC 25021, 2014*),
- ohraničení celkového simulovaného a simulujícího systému vůči okolí – věcné, technické i technologické,
- požadavky na vnitřní ohraničení jednotlivých částí simulujícího systému – uživatelský pohled na fragmentaci dílčích uživatelských rozhraní,
- rekapitulace vstupních proudů a jejich konzultace se zadavatelem, včetně rekapitulace atributů jednotlivých entit.

Specifikace uživatelských požadavků

Na základě schůzky se se zadavatelem, provozovatelem simulátoru a, optimálně také, s uživatelem, dojde k vymezení uživatelských požadavků. Schůzka by měla pomoci specifikovat uživatelské požadavky tak, jak budou později uvedeny v legalizační dokumentaci.
Základní požadavky specifikované uživatelem by měly obsahovat:

- účel systému,
- jednotlivé sledované veličiny (výstupy),
- jednotlivé veličiny sloužící k parametrizaci simulátoru,
- periodické děje ovlivňující simulátor,
- charakter systému a vnitřní děje,
- požadavky na rozhraní pro uživatele,
- požadavky na dekompozici uživatelských pracovišť,
- požadavky na napojení na další systémy,
- požadavky na hardware,
- požadavky na jakost systému.

Základní požadavky specifikované provozovatelem simulátoru by měly obsahovat:

- požadavky na rozhraní pro uživatele (především kompoziční a prostorové řešení),
- požadavky na hardware a prostředky pro monitoring hardwaru,
- požadavky na logovací systémy a systémy vzdálené údržby,
- požadavky na architekturu systému (centralizovanou, distribuovanou).

Základní požadavky specifikované uživatelem simulátoru by měly obsahovat:

- požadavky na výstupy ze systému (s důrazem na formu),
- požadavky na uživatelské rozhraní.

Na základě sesbíraných údajů vznikne *Předběžný soupis požadavků*.

Formalizace vymezení systému

Na základě *Předběžné analýzy systému* a *Upřesnění vymezení simulovaného systému* dojde k formalizaci těchto informací v jednoznačném dokumentu. Systém je formalizován prostřednictvím blokového schématu, který rámcově vystihne jednotlivé procesy a děje v systému. Součástí popisu tohoto blokového vymezení systému je také specifikace jednotlivých entit a jejich atributů. Vhodnou formou je například použití schémat ArchiMate, nebo specializovaných BPM schémat.
Kromě blokového schématu je třeba specifikovat veškeré údaje do formy dvou tabulek. První tabulka je specifikována jako tabulka entit – kde budou vyznačeny entity a jejich atributy. Druhou tabulkou je tabulka procesů, která uvádí pro jednotlivé procesy přípustné entity a zároveň popisuje daný proces.
Tato formalizace musí mít takový zápis, aby byla jednoznačná pro budoucí analytické zpracování, avšak zároveň uvedena v takové formě, aby byla přijatelná pro zadavatele systému. Dokument *Formální vymezení systému* se stane přílohou budoucí smluvní dokumentace.

Formalizace uživatelských požadavků

Na základě *předběžné specifikace uživatelských požadavků* dojde k formalizaci všech požadavků dle příslušných doporučení či norem, a to především ISO 25021:2014 (2014) a ISO 29148:2011 (2011). Uživatelské požadavky budou specifikovány libovolným zápisem, přičemž doporučení je kladeno na zápis ve formě tabulky, nebo za využití specializovaných softwarových prostředků. Povinnými údaji jsou především:

- unikátní identifikátor požadavku,
- název požadavku,
- specifikace požadavku,
- závislosti požadavku,
- kritéria ověřitelnosti požadavku.

Uvedení přesných a jasných kritérií ověřitelnosti pro každý požadavek je naprosto klíčové. Požadavky budou formalizovány v dokumentu *Specifikace uživatelských požadavků*. Tento dokument bude součástí budoucí smluvní dokumentace.

Kompletace smluvní dokumentace

Veškerá připravovaná dokumentace musí odpovídat právním předpisům, vnitřním směrnicím a dalším souvisejícím normám. Vnitřní směrnice jednotlivých institucí jsou vstupními dokumenty této fáze řešení. Soulad se zákony by měla ověřit právní kancelář či právní oddělení. V případě, že je uzavírán smluvní vztah mezi partnery z více zemí, je třeba klást důraz na interpretaci mezinárodního práva a tomu také adaptovat všechny klíčové dokumenty. Zvláštní zřetel na odlišnosti práva je třeba klást především v případě s uzavíráním smluv se stranou z jiného právního prostředí, nežli je prostředí evropské (právě v případě USA dochází k významným

odchylkám i v právu autorském a dalších předpisech souvisejících s průmyslově právní ochranou) a stranou, která se řídí zvláštními předpisy pro mezinárodní instituce.
Smluvní dokumentace je základní dokumentací, která zajistí práva a povinnosti všech smluvních stran a obsahuje především (ne však pouze):

- objednávku služeb (nebo obdobnou objednávku),
- vymezení povinností smluvních stran, a to především:
  - vymezení termínů realizace,
  - vymezení milníků realizace s postupným předáváním, přičemž je doporučeno, aby jednotlivé milníky odpovídaly fázím řešení dle této metodiky,
  - vymezení termínů pro jednotlivé milníky,
  - vyslovení součinnosti objednatele tak, aby nemohlo docházet k prodlevám z důvodu nesoučinnosti objednatele,
  - vymezení odměny pro realizační tým,
  - vymezení pokuty v případě nedodržení termínů,
  - vymezení pokuty v případě nedodržení jakosti (specifikovaných požadavků),
  - vymezení výpočtu vícenákladů v případě úpravy požadavků,
  - vymezení dalších smluvních stran,
  - odkaz na přílohy v podobě Formálního vymezení systému a Specifikace uživatelských požadavků,
- řešení problematiky fakultativního „doobjednání“ sady dalších simulačních experimentů, a to včetně časového rozsahu, ceny, nebo nároku na řešení problematiky třetí stranou.

Je důrazně doporučeno kompletaci smluvní dokumentace delegovat na právní kancelář/právní oddělení, nebo tuto dokumentaci s právní kanceláří/právním oddělením konzultovat.

Legalizace smluvní dokumentace

Fázi lze považovat za dokončenou až okamžikem, kdy je *smluvní dokumentace* legalizována všemi smluvními stranami uvedenými ve smluvní dokumentaci.
Kromě *smluvní dokumentace*, jak je vymezena výše, je třeba zajistit legalizaci Formálního vymezení systému a *Specifikaci uživatelských požadavků*.
V případě zakázky nad 1.000.000 Kč je metodicky doporučeno legalizaci provést formou ověření nezávislou státní institucí, to jest legalizované dokumenty dále ověřit úřední mocí (tzv. úřední ověření podpisu, případně úřední uznání podpisu za svůj).

Výstupy

Výstupy pro fázi A jsou (nikoliv výhradně):

- Výstupy využité během fáze A, které nebudou dále využívány
    - předběžné analýzy systému,
    - předběžný soupis požadavků.
- Formální vymezení systému
    - blokové schéma návaznosti procesů,
    - tabulka entit,
    - tabulka procesů.
- Specifikace uživatelských požadavků
    - soupis požadavků,
    - soupis kritérií ověřitelnosti každého z požadavků.
- Legalizovaná smluvní dokumentace.

Fáze B – Analýza

Cíle

Cíle fáze B jsou:

- hluboké pochopení systému, jeho formalizace,
- získání přesných údajů pro simulátor,
- stanovení míry abstrakce,
- tvorba logické kompozice simulátoru,
- příprava na implementační práce.

Způsob dosažení cíle

Klíčovou fází pro úspěšnou realizaci simulátoru je provedení vysoce kvalitní analýzy. Analýza vytvoří komplexní náhled na systém, jeho rozhraní a jednotlivé vnitřní děje v systému. K analýze je třeba přistupovat z logické úrovně. Pro definici dějů a proměnných je vhodné používat obecnou definici dat, včetně ohraničení přípustných hodnot. Konkrétní specifikaci datových typů není v této fázi vhodné provádět.
Analýzu by měl provádět pracovník s odpovídající kvalifikací a nadhledem nad problematikou. První fáze analýzy probíhá bez stanovení míry abstrakce, a proto je sběr dat relativně podrobný. Proto aby nedocházelo ke zbytečným ztrátám sběrem takových dat, které nebudou pro systém relevantní, je třeba určité množství zkušeností. Ani velmi kvalitně zpracovaná dokumentace z fáze A nemusí v této situaci pomoci proces analýzy optimalizovat.
Ohraničení systému je nezbytné dodržet. Během procesu analýzy je třeba absolvovat akceptační schůzku se zákazníkem. Akceptační schůzka je posledním okamžikem, kdy je možné pozměnit akceptační kritéria (stanovená a legalizovaná ve fázi A), a tím změnit podmínky validace. Zároveň na základě akceptační schůzky lze stanovit míru abstrakce a finalizovat analýzu celého systému.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi B.

Externí dokumenty

- Dokumentace použitých standardů.

Interní dokumenty

- Formální vymezení systému,
- specifikace uživatelských požadavků.

Postup

Analýza na úrovni simulátoru

První částí metodického postupu při tvorbě kompletní analýzy je analýza na úrovni simulátoru. Jde o rozpracování dokumentu Specifikace uživatelských požadavků tak, aby byl použitelný pro implementaci řešení. Na rozdíl od *předběžné analýzy* je možné využívat vnitřní stavové entity, nebo atributy. Analýza musí být vystavěna tak, aby byly splněny všechny uživatelské požadavky formalizované v dokumentu Specifikace uživatelských požadavků. Metodika počítá s dalšími diskusemi se zadavatelem, pokud to pro odstranění nejasností bude nutné.
Níže je uvedena typická struktura prvotní analýzy a obsah jejích jednotlivých částí:

- Analýza vstupního rozhraní:
  - základní požadavky – emulace, virtuální rozhraní, GUI,
  - rozmístění ovládacích prvků vstupního rozhraní,
  - typ ovládacích prvků, krok ovládacích prvků, závislosti a blokace vstupních prvků,
  - označení atributů/proměnných přímo řízených vstupními prvky,
  - rozhodnutí, zda bude simulátor čerpat data z dalších systémů, a to offline nebo online.
- Analýza výstupních rozhraní:
  - základní požadavky – online animace, statistické výstupy, AAR,
  - požadované hodnoty – typy sledovaných hodnot, jejich vzorkování, způsob, interpretace,
  - zvláštní požadavky online animace – rozhraní, forma, synchronizace,
  - uživatelské požadavky na rozložení grafického výstupu.
- Analýza transparentních datových proudů v rozsahu rozhodnutí vymezující, zda:
  - Bude simulátor obsahovat jednotky pro vizualizaci (nebo dohledové systémy)
  - Bude simulátor obsahovat systém logování (a na jaké úrovni)
  - Bude simulátor připojen na online systémy
- Analýza entit:
  - Určení mobilních entit
    - zkoumané entity,
    - uživatelský avatar,
    - oponentské avatary,
    - zdroje,
    - produkty/obsluhované entity,
    - další mobilní entity.
  - Určení stabilních entit (jen pro simulátory, kde se takové entity vyskytují)
    - překážky,
    - externí vlivy (teplota, počasí, denní doba).
  - Klasifikace mobilních i stabilních entit
    - dle temporarity v systému (dočasné, stálé),
    - dle výskytu v systému.
  - Analýza atributů
    - přiřazení atributů jednotlivým entitám,
    - analýza sdílených atributů (společné pro třídu entit),
    - klasifikace atributů dle úrovně abstrakce.

- Analýza procesů:
  - vyhledání jednotlivých procesů,
  - atomizace procesů tam, kde je to nutné,
  - vytvoření vazby procesů a jejich zdrojů,
  - temporarita procesů (časové omezení provádění procesů),
  - atributy procesů,
  - klasifikace atributů procesů dle úrovně abstrakce.
- Vytvoření vazební tabulky zdroje-procesy pro přípustné kombinace procesů a zdrojů (entit),
- Tvorba základní typové mapy prostředí (obecný level design),
- Tvorba analytické mapy vyznačující:
  - systém a jeho části (procesy),
  - přípustný výskyt entit v jednotlivých částech systému,
  - dopad stabilních entit na systém,
  - vyznačení přípustných hodnot sledovaných atributů.

Zadavatelská konzultace a stanovení úrovně abstrakce

Po vytvoření analytické koncepce reprezentované Analytickou mapou je třeba realizovat schůzku se zadavatelem. Na schůzce je třeba řešit problematiku entit a jejich atributů ve vztahu k úrovni abstrakce. Zadavatel se tak má příležitost vyjádřit k tomu, které entity a atributy požaduje jako sledované. Na rozdíl od *předběžné analýzy* jde o značně rozpracovanou analýzu, která dosahuje mnohem vyšší přesnosti, proto obsahuje mnohem více klíčových údajů.
Závěr ze schůzky je třeba formalizovat tak, aby neodporoval zadávací dokumentaci. Je tedy třeba upravit analýzu, nebo zadávací dokumentaci a tuto změnu si nechat závazně potvrdit objednatelem. Na základě těchto údajů může dojít ke stanovení úrovně abstrakce.

Analýza vstupních dat

V okamžiku, kdy jsou vymezeny klíčové entity, procesy a jejich atributy je možné přejít ke zpracování vstupních dat (zpravidla časových řad). Při tomto zpracování je třeba vycházet buď z dokumentace dodané zadavatelem, nebo pokud tak bylo sjednáno v zadávací dokumentaci, je třeba analyzovat požadovaná data a provést vlastní měření.
Po získání surových dat je možné přejít k jejich analýze. Na základě analýzy jednotlivých (nejčastěji bodových) historických datových proudů je možné provést implementaci generátorů příslušných stochastických proudů, přičemž je možné použít metodiku ISO 22514:2014 *Statistical Methods in Process Management*, graficko-statistické zpracování dat (např. regresními modely), nebo provést zavedení stochastických kalendářů. Je třeba neopomenout data, která vykazují významné odchylky (pokud je systém časově závislý, může jít např. o státní svátky atp.).
Provedenou transformaci měření dat pro sesbíraných požadavky a jejich převod na vstupy stochastických procesů je třeba verifikovat. Doporučenou verifikační metodou je například statistická metoda prostřednictvím testu dobré shody.

Detaily jsou uvedeny ve fázi G verifikace. Během zpracování je nutné soustředit se na získání především následujících údajů:

- Časové údaje:
  - Mobilní entity:
    - časy vstupů do systému,
    - časy pohybu po systémů (mezi vstupem a procesem, mezi procesy, mezi procesem a výstupem ze systému).
  - Stabilní entity:
    - časy a typy (případně podmínky) provádění dynamických změn,
    - interakční parametry mobilních a stabilních entit.
  - Procesy:
    - časy provádění procesů pro všechny přípustné parametry (tabulkově, nebo funkcionálně je třeba vyjádřit časové náročnosti procesů).
- Údaje pro stanovení omezení:
  - fronty – časové i rozsahové limity,
  - procesy – časové i rozsahové limity,
  - jakost (z pohledu simulovaného systému) – časové i rozsahové limity.
- Údaje pro tvorbu virtuálních prostředí:
  - parametry prostředí (rozměr, tvar, výšková mapa atp.),
  - parametry stabilních entit,
  - parametry mobilních entit,
  - parametry rozhraní (pro všechna rozhraní).
- Analýza požadovaných technologických rozhraní.

Dekompoziční analýza na logické úrovni

Jelikož požadavky na vyvíjený simulátor mohou obsahovat zvláštní specifikace na dekompozici na logické úrovni, je třeba tuto dekompozici zachytit. Nejde o zachycení celkové dekompozice, ale pouze dekompozice funkční. Při této analýze je kladen důraz především na dekompozici:

- Realizovanou na základě požadavků na interaktivní systémy. Každá soustava uživatelských rozhraní, které mají sloužit pro jednoho uživatele, to jest každý seat, by měl odpovídat jedné samostatné dekomponované části. Tentýž požadavek platí pro rozhraní pro sledování celého systému jako je například globální mapa, jejímž účelem je vizualizovat data pro jednoho i více uživatelů.
- Realizovanou dle požadavků na rozhraní na externí systémy, pokud není zvolen jiný způsob systémové integrace. Každá vazba na externí offline, nebo online systém by měla být obsluhována vlastní, dekomponovanou, částí simulátoru.
- Zvolenou na základě predikce významných výpočetních režií, nebo dalších relevantních výkonových požadavků. Je třeba odlišit dekompozici na logické úrovni od budoucí architektonické dekompozice. Přesto však situace, kdy analytik předpokládá oblasti příliš vysoké hustoty výskytu procesů, nebo entit, a s tím spojeným problémů, jako například složité sledováním systému, nebo komplikovanou verifikaci či validaci takové části, je vhodné tyto vnitřně složité systémy dále dekomponovat.
- Navrženou dle zvláštních požadavků zadavatele, přičemž tuto dekompozici je třeba zmínit i v případě, že jde o dekompozici na jiné, nežli logické úrovni.

Analýza vnitřní logiky procesů

Při budování simulátoru je jedním z nejnáročnějších úkolů vybudování vhodné vnitřní struktury procesů. Kromě budování vnitřní logiky procesů je možné procesy dále klasifikovat (například na řídící a výkonné).
Přestože samotný proces není triviální, *Metodis* využívá pro řešení této problematiky faktu, že:

- již při prvotní analýze došlo ke specifikaci procesů a jejich popisu formalizovaným způsobem,
- v prvním kroku tohoto bloku byla zdůrazněna potřeba provést atomizaci a analýzu procesů.

Fakt, že je některý z procesů náročný na kvalifikaci personálu a lidské zdroje tak metodika nepopírá, avšak vzhledem k tomu, že je koncipována jako obecná, dále se tomuto tématu nevěnuje.

Analýza na úrovni implementovaného softwaru

Analýzu na úrovni implementovaného softwaru je třeba provést jen pokud bude celé řešení implementováno řešitelským týmem. Pokud bude použit software pro rychlé prototypování simulátorů lze tento krok vynechat. Analytická dokumentace, především tvorba analytických tříd, případů užití a sekvenčních/funkčních diagramů je třeba realizovat v souladu s ISO12207. Dále je možné použít i další unifikované způsoby/doporučení pro návrh softwaru vč. dílčích částí metodik, jako např. RUP.
Vhodnými výstupy, které by měly vzniknout, jsou především:

- model analytických tříd,
- use – case model a
- data – flow diagam.

Formalizace analýzy

Formalizací analýzy se rozumí tvorba uceleného dokumentu s analýzou. Tento ucelený dokument je povinným výstupem celé fáze. Musí obsahovat výstupy ze všech dílčích kroků (vč. označení verze).

Analýzy demonstračních scénářů

Během analýzy dochází k časté interakci se zadavatelem. Je velmi výhodné, pokud se podaří pro simulátor vytvořit jednoznačný kalibrační scénář. Respektive stanovit takové nastavení systému a vstupních proudů, pro které jsou známé hodnoty výstupů. Tvorbu demonstračního scénáře je možné zajistit například prostřednictvím anonymizovaných historických dat z reálného provozu. Dalším způsobem, jak je možné zajistit údaje pro demonstrační scénář, je zajištění takových vstupních dat a nastavení procesů, pro které bude možné definovat průchodnost entity systémem (resp. jeho jednotlivými větvemi) a zároveň pomocí numerických hodnot ověřit výsledný stav systému. Demonstrační scénáře formalizované ve *Specifikaci demonstračních scénářů* budou využity při validaci řešení.

Výstupy

Výstupy pro fázi B mohou být (nikoliv výhradně):

- Analytický model systému:
  - popis simulátoru na logické úrovni,
  - struktura simulátoru,
  - popis entit, procesů, jejich atributů a časových závislostí,
  - analýza na úrovni dalšího použitého softwaru/jeho rozhraní,
  - charakter proudu výstupních dat,
  - specifikace uživatelských rozhraní,
- Specifikace demonstračních scénářů,
- Zákaznické potvrzení rozsahu systému.

Fáze C – Architektura

Cíle

Cíle fáze C jsou:

- volba vhodné provozní architektury simulátoru,
- vytvoření architektury na úrovni simulátoru, vč. dekompozice procesů,
- definice všech proměnných, atributů a interakcí,
- předpis pro rozhraní,
- tvorba architektonického modelu softwaru vč. případné dekompozice,
- formalizace programátorské dokumentace.

Způsob dosažení cíle

Architektura simulujícího systému je klíčovou části metodického postupu realizovanou mezi obecným zadáním a vlastní systémovou implementací. Právě během architektonického návrhu jsou konečně ukotveny veškeré vlastnosti systému a tyto vlastnosti jednoznačně specifikovány. Obecným hodnotám jsou tak přiřazeny konkrétní datové typy, meze, jsou stanovena kritéria pro jednotlivá data a procesy. Navržený simulátor získává během architektonického zpracování konkrétní podobu – například co do počtu výpočetních uzlů, na kterých bude realizován, ve vztahu k dalšímu hardwaru atp.
Při architektonickém návrhu je třeba vycházet z analytického řešení a postupně rozvíjet vlastní architektonický návrh. Dílčími kroky jsou:

- Architektonické řešení se zaměřením na simulátor globálně:
    - architektonický návrh ve vztahu k provozní infrastruktuře,
    - přijatelná dekompozice simulátoru,
    - způsob práce s časem (resp. způsob implementace času),
    - použitelná běhová infrastruktura.
- Architektonické řešení na úrovni jednotlivých softwarů,
- Architektura systémové integrace,
- Architektura ve vztahu k hardwaru.

Kromě specifikace dílčích postupů je však v kapitole specifikováno několik základních přístupů, které na rozdíl od zbytku metodiky, nejsou specifikovány žádným standardem, nebo nejsou dostatečně kvalitně popsány v žádné literatuře.

Přípustné metody dekompozice 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. Rozšiřující informace jsou proto uvedeny v příloze B: Vybrané metody dekompozice.

Přípustné běhové infrastruktury

Protože je volba architektury klíčová a literatura často neposkytuje dostatečné zdroje, metodika se k problematice dále vyjadřuje v příloze C: Vybrané běhové infrastruktury.

Techniky pro implementaci interaktivních zásahů

Techniky pro implementaci interaktivních zásahů jsou pro oblast simulátorů velmi specifické svým chováním proto, že je nutné zajistit vhodné zapojení této formy externí parametrizace simulátoru za běhu simulace do simulačního výpočtu (stejně tak je nutné zajistit zaznamenání tohoto vstupu). Při implementaci je nutné dále zohlednit metodu práce s časem a další okolnosti. Vzhledem k tomu, že není snadné najít kvalitní dokument, na který by mohla metodika odkázat, jsou v Příloze D uvedeny vybrané vhodné způsoby pro implementaci interaktivních zásahů.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi C.

Externí dokumenty

- Předpisy související norem.

Interní dokumenty

- Analytický model systému.

Postup

Architektonické řešení na úrovni návrhu simulátoru

Kromě standardní softwarové architektury je třeba klást zvláštní pozornost na architekturu na úrovni simulátoru. Simulátor je software relativně specifický a před tvorbou vlastního návrhu tříd je nezbytné vytvořit specifická rozhodnutí o architektuře simulátoru, těmito rozhodnutími jsou především:

- volba běhové architektury,
- určení dekompozičních kritérií,
- určení způsobu provedení dekompozice procesů,
- volba metody práce s časem a volba synchronizačních metod,
- příprava předpisu rozhraní pro uživatelská rozhraní,
- volba způsobu uživatelských interakcí, pokud je toto nutné.

Proces architektonického řešení na úrovni simulátoru je vhodné zahájit tvorbou konceptu simulátoru a jeho základní dekompozice včetně určení dekompozičních kritérií (např. 1 výpočetní

uzel pro každý seat; 1 výpočetní uzel pro RTI; 1 výpočetní uzel pro mapový server). Určení konceptu simulátoru a jeho struktury efektivně poslouží jako výchozí bod pro další vývoj.
Z konceptu simulátoru je možné dále vytvářet dekomponované procesy, a to vždy tak, aby pro jednotlivou dekompozici zvyšovaly efektivitu provádění simulačních výpočtů, nebo snižovaly náklady na servis a udržitelnost a dále podpořily přehlednost řešení.
Na základě konceptu simulátoru je možné také začít rozpracovávat jednotlivá uživatelská rozhraní/emulovaná rozhraní/další řešení pro uživatelské vstupy. Zároveň lze na základě koncepce vytvořit komunikační schéma, které ilustruje přenášené informace mezi federáty – tyto informace lze v této fázi formalizovat včetně uvedení datových typů.

Architektonické řešení na úrovni implementace softwaru

Architektonické řešení na úrovni softwaru znamená rozpad předchozí architektury do ryze softwarového návrhu, u kterého dojde ke specifikaci tříd, struktur a atributů, případně dalších specifikací v souladu s ISO 12207. Architektonické řešení se nijak neliší od běžných architektur.

Architektura systémové integrace

V softwarovém inženýrství je pojem systémové integrace používán nejčastěji k popisu stavu, kdy jsou vzájemně (volně nebo těsně) spojeny heterogenní systémy, které původně nebyly určeny ke vzájemné spolupráci. Pro potřeby této metodiky je systémová integrace chápána jako potenciální jednostranné řešení, které je vždy optimalizováno vzhledem k potřebám konkrétních systémů. Tedy i vstupní rozhraní/brána pro data z externího systému je touto metodikou považována za součást systémové integrace.

Při tvorbě architektonického modelu rozhraní systémové integrace je třeba zohlednit především:

- jaký je charakter systému, který je integrován (např. online, databázový systém),
- jakým způsobem je systém integrován (vstupní/výstupní),
- jaké má být chování simulátoru v případně nedostupnosti integrovaného řešení,
- technické specifikace integrace (latence, typy připojení atp.),
- typ datových přenosů (XML, stream, JSON),
- konkrétní datové specifikace (vymezení dat vč. názvů a datových typů).

Architektura ve vztahu k hardwaru

Architektonický přístup k hardwaru je v rámci metodiky vymezen jen pro hardware, který není běžnou součástí počítače (tj. monitor, myš a klávesnice jsou považovány za implicitní hardwarová řešení a není třeba pro ně vytvářet explicitní architektonické návrhy). Za hardware jsou považovány v širším slova smyslu veškeré technické prostředky, které budou součástí simulátoru. V užším slova smyslu jsou pak myšleny pouze ty prostředky, jejichž stav je možné interpretovat/převést na elektrický (digitální) signál, nebo takové prostředky, které mohou elektrický (digitální) signál interpretovat, pakliže jde o výstupní zařízení. Tato zařízení je možné specifikovat do dvou základních typů a to:

- unifikovaná zařízení pro ICT,
- zvláště zhotovená zařízení pro ICT.

Přičemž unifikovanými zařízeními pro ICT je, v rámci této metodiky, myšleno takové zařízení, které má pro standardní operační systémy dostupné vlastní ovladače. Jedná se například o běžná USB zařízení (volanty, pedály, joysticky, snímače pohybu, snímače polohy atp.) na vstupu, nebo USB zařízení (signalizace, zvuková zařízení) a grafická zařízení (virtuální realita, projektory) na výstupu.

Pro tuto třídu zařízení je nutné architektonicky definovat především:

- specifikaci typu externího zařízení,
- specifikaci knihovny, je-li to nutné,
- určení konkrétního chování v kontextu se softwarem,
- definovat obslužnou třídu, přiřadit atributy a uvést přípustné hodnoty.

Zvláštní pravidla platí pro zvláště zhotovená zařízení pro ICT, kterými jsou pro potřeby této metodiky myšlena všechna zařízení, která nemají dodávány implicitní ovladače pro poskytování veškerých dostupných technických funkcí. Tradičními příklady jsou individuální zhotovená zařízení využívající preprocesoring prostřednictvím mikropočítače Arduino, které následně ovládá konkrétní emulované rozhraní.

Pro tuto třídu ICT zařízení je nutné uvést, kromě údajů definovaných pro unifikovaná zařízení, zároveň ještě:

- specifikaci elektrických, mechanických a dalších fyzikálních veličin,
- specifikaci rozměrů, poloh a dalších technických údajů,
- specifikaci ovládacích rozhraní a knihoven pro tato ovládací rozhraní,
- architektonický návrh ovládacího softwaru – je-li součástí takového řešení,
- konkrétní převodní vztahy mezi hardwarem a softwarem, vč. přiřazení hodnot jednotlivým atributům řídícího softwaru,
- technické specifikace zabezpečení, třídy bezpečnosti a omezení pro provozování (tyto údaje je nutné promítnout také do uživatelské dokumentace).

Návrh architektury logovacích systémů

Vzhledem k rozsahu softwarového díla je součástí řešení reportovací systém. Architektura reportování může vycházet z použitého programovacího jazyka a využívat jeho vnitřních funkcí a procedur (např. pro Java vnitřní rutiny Logger). Logování je vždy třeba zavádět do souboru. Přípustná je architektonická specifikace více úrovní logování.

Jednotlivé logovací záznamy obsahují především:

- klasifikaci záznamu (např. chyba, upozornění, vytvoření),
- iniciační třídu,
- iniciační čas,
- hodnoty klíčových atributů.

Kompletace architektonických dokumentů

Po realizaci činností specifikovaných v předchozích blocích fáze je možné přejít ke kompletaci programátorské dokumentace. Programátorská dokumentace by měla mít standardní členění pro potřeby metodiky rozšířené o vyznačení:

- lokálních částí,
- globální části,
- integrovaných systémů a rozhraní,
- systémů pro práci s externím hardwarem.

Explicitní určení jednotlivých částí pomůže budoucí implementaci řešení a usnadní také testování. Odlišením jednotlivých částí dojde k efektivnímu rozdělení implementačních prací na další úrovni (nikoli pouze na úrovni tříd, tak jak je zvykem při standardním architektonickém návrhu).

Celá architektonická dokumentace již musí obsahovat:

- zvolené programovací jazyky pro implementaci konkrétních částí systému,
- vyznačenou verzi, datum a identifikátor pro každou třídu/strukturu,
- pro globální část:
  - předpis jednotlivých rozhraní (vč. konkrétních datových typů),
  - popis datových toků, vč. datových typů,
  - technické požadavky na zajištění spojení a ověřování spojení,
  - technickou specifikaci chování globální části,
  - specifikaci dalších technologií/brán/wrapperů pro externí systémové vazby,
  - specifikaci pro integraci dalších technologií,
- pro lokální části:
  - diagramy tříd, specifikaci rozhraní, metod a atributů,
  - předpis chování (formou psaného textu, diagramů, aj.),
  - další schémata nutná pro úspěšnou implementaci,
- pro funkční celky a sestavy:
  - stavový diagram,
  - popis fungování funkčního celku,
  - definice způsobů práce s časem,
- pro uživatelská rozhraní:
  - návrh rozmístění ovládacích prvků uživatelských rozhraní,
  - popis konkrétních ovládacích prvků, vč. přípustné množiny hodnot/stavů,
- pro externí systémy a hardware:
  - specifikaci rozhraní po HW i SW stránce,
  - zvolený přístup, technologii, programovací jazyk,
  - specifikaci proměnných,
  - technické požadavky na latenci, konzistenci, časové údaje,
- specifikaci pro testy (s důrazem na limitní stavy):
  - jednotkové,
  - integrační,
  - funkční.

Je třeba si uvědomit, že architektonický model musí být komplexní a kompletní. V okamžiku ukončení fáze je třeba znát veškerá rozhraní, veškeré specifikace a použité technologie. Veškeré hodnoty (proměnných i atributů) musí být určeny pro konkrétní datové typy, stejně jako v oblasti přípustných hodnot. Všechny tyto definice jsou klíčové nejen pro úspěšnou realizaci implementace, ale zároveň také pro úspěšný průběh verifikace celého řešení.

Výstupy

Výstupy pro fázi C mohou být (nikoliv výhradně):

- Programátorská dokumentace.

Fáze D – Implementace lokálních částí

Cíle

Cíle fáze D jsou:

- zajištění implementace jednotlivých lokálních částí.

Způsob dosažení cíle

Součástí aplikačních vývojových prací je nezbytná také vlastní programátorská implementace. Tato implementace by měla být řízena vybranou technikou tak, jak bylo zmíněno v úvodu celé metodiky (doporučena například SCRUM). Vlastní implementace je, v případě kvalitní programátorské dokumentace, realizovatelná libovolně paralelně i sekvenčně. Pro úspěšné dosažení cílů této fáze je tedy nezbytné kvalitní realizování fáze předchozí. Vzhledem k minimálním požadavkům na kvalifikaci uživatelů *Metodis* není konkrétní atomizace implementace součástí metodiky a metodika se zabývá pouze doporučeními, která jsou nad rámec kvalifikace programátorů – to jest doporučeními, které se vztahují k budování distribuovaných simulátorů.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi D.

Interní dokumenty

- Programátorská dokumentace.

Postup

Implementace vnitřních rozhraní

Přestože kvalitní architektonický návrh, který vznikl jako výstup fáze C, umožňuje širokou škálu uplatnitelných postupů programování, je možné vyslovit několik doporučení. Nejvýznamnějším doporučením je zahájení práci na vnitřních síťových rozhraních (stejné doporučení lze nalézt např. v HLA). Je třeba si uvědomit, že distribuovaný simulátor je distribuovanou aplikací, která má jasně určeny komunikační kanály, které však nejsou určeny v žádné knihovně. Tedy není možné použít generickou obsluhu simulátoru (až na výjimky, např. při architektuře postavené na zasílání zpráv). Nejvýhodnější proto je začít implementační práce realizací vnitřních rozhraní. Toto je výhodné především u DIS, nebo HLA, kteréžto architektury mají vlastní nástroje, prostřednictvím kterých je možné provést generování těchto rozhraní.

Implementace uživatelských rozhraní

Uživatelská rozhraní se mohou významně lišit svým provedením. Standardní GUI je třeba realizovat, avšak není třeba mu věnovat zvláštní pozornost. Pokud je však uživatelské řešení postaveno na emulaci prostředí (fyzické emulaci, nebo emulaci prostřednictvím virtuální reality), je žádoucí vyvíjet tato uživatelská rozhraní od raných částí realizace fáze. Často se jedná o softwarové části, jejichž odladění je časově velmi náročné. Právě časová náročnost a riziko spojené s prodlením je důvodem, proč je výhodné zahájit realizaci právě touto částí.

Implementace dalších systémů

Implementace dalších simulujících podsystémů lze považovat za standardní v souladu s běžnými programátorskými postupy.

Způsob uveřejňování výsledků

Každá implementovaná a kompilovaná část je považován za samostatný výsledek, který je třeba samostatně archivovat, verzovat a (na programové úrovni) dokumentovat.

Výstupy

Výstupy pro fázi D mohou být (nikoliv výhradně):

- Software jednotlivých lokáních částí.

Fáze E – Implementace globální části

Cíle

Cíle fáze E jsou:

- implementace softwarové části propojující jednotlivé lokální části.

Způsob dosažení cíle

Pokud je systém navržený jako distribuovaný (platí též pro paralelní systémy), je třeba vyřešit problematiku vzájemné komunikace federátů.
Provádění fáze E je zcela podřízeno technologii, která byla zvolena ve fázi C. Implementace musí proběhnout v souladu se zvolenou technikou, nebo technologií.

Pro nejrozšířenější technologie to znamená:

- Pro DIS je třeba vytvořit předpisy PDU packetů.
- Pro HLA je třeba vytvořit OMT.
- Pro centralizovanou architekturu je třeba vytvořit centralizovaný prvek a jeho rozhraní.
- Pro další programátorské techniky je třeba vytvořit rozhraní a jeho obslužnou část.

Vlastním způsobem dosažení cíle jsou programátorské práce.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi E.

Externí dokumenty

- Technické dokumenty příslušných doporučení a standardů v jejichž souladu proběhne implementace.

Interní dokumenty

- Architektonický návrh simulujícího systému.

Postup

Postup pro případ využití DIS

O komunikaci mezi federáty se v případě DIS stará vždy konkrétní kompetentní federát. Klíčové práce řešené během fáze budou souviset výhradně s datovými transakcemi. V případě DIS jsou data přenášena formou PDU – tzv. Protocol Data Units. Jejich předpis je dán standardem IEEE1278. Z programátorského hlediska se jedná o přímý datový stream, který je interpretován

vždy na konkrétním federátu bez ohledu na platformu či programovací jazyk. Při přípravě jednotlivých PDU packetů je třeba klást důraz na správnou kategorizaci PDU a správné členění informací do jednotlivých částí PDU packetu (hlavičky, PDU family a PDU body). Je třeba dodržet stromový systém tvorby jednotlivých PDU packetů a vhodně navrhnout jednotlivé packety tak, jak vyžaduje IEEE1278.

Postup pro případ využití HLA

Komunikace je v případě využití architektury HLA řízena prostřednictvím aktivní komponenty zvané RTI. Touto komponentou je nutné celé řešení vybavit nezávisle a není třeba ho implementovat. RTI plní většinu funkcionalit předpisu IEEE1516:2010, fakticky doimplementovat je třeba například datové nosiče časových údajů atp. Klíčové práce, které je třeba realizovat během této fáze budou soustředěny na definice datových typů a jejich formalizaci. HLA využívá pro přenos dat mezi federáty souborový formát XML. Architektonické oddělení dat na vrstvu, která přechází mezi federáty a na vnitřní stavové proměnné federátů zůstává relevantní. Avšak přesto je v příslušném XML předpisu nutné definovat obojí, tedy také vnitřní datové typy jednotlivých federátů tak, aby bylo možné externě sledovat rozvoj vnitřního stavového prostoru. Komunikační XML předpis, který bude RTI využíván se nazývá OMT (Object Model Template) a jeho podoba je standardizována v předpisu IEEE1516:2010.
Na rozdíl od definování prosté datové vrstvy je třeba dále rozpracovat přípustné interakce mezi jednotlivými federáty. To jest, prostá změna stavových proměnných je možná, avšak může k ní docházet jen na základě volání majitele takové stavové proměnné. V případě potřeby federátu zasahovat do jiného federátu, musí k takovéto operaci docházet výhradně prostřednictvím RTI. Pro RTI však platí, že může provádět jen takové interakce, které jsou předem definovány v OMT.

Obecně je tedy v případě využití třeba připravit OMT, který obsahuje:

- Rozlišení objektů na objekty, interakce a atributy definované pro celou federaci.
- Rozlišení objektů na objekty, interakce a atributy definované pro jednotlivé federáty.
- Definice všech přípustných interakcí, včetně interakčních atributů.
- Definice všech přípustných objektů, včetně jejich atributů.

Samotný soubor OMT je třeba fyzicky přenést ke každému federátu. Soubor je třeba verzovat a řádně dokumentovat.

Postup pro případ volby centralizované architektury

Pokud není zvolena žádná obecně známá architektura/doporučená architektura, ale dojde k čistému řešení na úrovni client – server, je třeba předem určit, zda půjde o aplikačně, nebo službově orientované řešení (SOP). V případě volby servisně orientovaného prostředí je pro vývoj výhodné využít další literaturu, neboť tato architektura je poměrně novou koncepcí poprvé publikovanou až v roce 2002 (Silliti a kol., 2002), ve které je možné čerpat informace o tomto konceptu.
Přímé programování řešení klient – server je v dnešní době nevýhodné z důvodu nároků na datové přenosy. V rámci metodiky je doporučeno, aby se řešitelský tým ujistil, že řešení klient-server je skutečně vhodným řešením.

Prvním krokem při volbě serverově orientované architektury je ustanovení centralizovaného serveru za federát. Tedy, přestože se server aktivně nepodílí na simulačním výpočtu, ale slouží pouze jako infrastruktura, je třeba, pro zjednodušení prací, prohlásit řešení za nezávislý federát. Pro tento federát je třeba realizovat fáze C a E. Vzhledem k přímému přístupu architektury klient – server by mělo dojít k implicitnímu vyřešení komunikačních protokolů.

Postup v případě využití jiných metod

Metodika připouští využití libovolných dalších programátorských technik. Vzhledem k tomu, že řešení bez centralizovaného prvku je předepsáno DIS, užijí se pro všechny techniky obdobně právě předpisy pro DIS. Ke změnám dojde jen ve způsobu komunikace, která nemusí být unifikována (např. náhrada PDU za CORBA).
V případě využití řešení s virtuálním centralizovaným prvkem se obdobně využijí pravidla pro HLA. Opět je přípustné nahradit komunikační formáty v podobě XML jiným způsobem. Stejně tak je přípustné odejmout funkcionalitu řízení času z virtuálního centralizovaného prvku a převést ho přímo na jednotlivé federáty. Avšak pro všechny tyto kombinace lze využít obdobné předpisy pro HLA.

Doporučení v případě kombinace více technik

V některých situacích je výhodné použít více technik současně. V některých případech – například rozšiřování již existujícího řešení, nebo propojení dvou, doposud nezávislých řešení, bude nutné provést systémovou integraci. Integrace je vždy realizována na úrovni globální části tak, aby jednotlivé simulátory těmito úpravami nebyly zasaženy.
Pro řešení je třeba zvolit aktivní softwarový prvek (wrapper), který bude na hranici obou systémů. Wrapper bude vždy implementován na aplikační úrovni. Úkolem wrapperu bude transparentně interpretovat informace z jedné federace, do federace druhé, a naopak. U rozsáhlejších systémů je vhodné wrapper řešit jako software běžící na vlastním výpočetním uzlu, nebo na uzlu, který sdílí s dalšími ve federaci obsaženými subsystémy.

Nasazení wrapperu by se mělo řídit těmito pravidly:

- Pokud je jednou z federací taková federace, která je řešena jako klient – server řešení, potom bude wrapper umístěn ve výpočetním uzlu serveru.
- Pokud je jednou z federací taková federace, která je řízena prostřednictvím RTI, potom bude wrapper umístěn na výpočetním uzlu RTI.
- Pokud jsou obě federace řízeny formou přímých datových přenosů s odpovědností na úrovni federátů (DIS a variace), potom by wrapper měl být umístěn na nezávislém výpočetním uzlu, pro který je nastavena vysoká priorita QoS.

Obecná kritéria realizace fáze E

Vzhledem k široké škále odlišných možností, které jednotlivé typy implementace globální části poskytují, je žádoucí stanovit jednotné kritérium realizace celé fáze tak, aby bylo jednoznačně možné určit, zda byla fáze dostatečně kvalitně realizována. Obecně jde prohlásit, že fáze byla úspěšně realizována, pokud existuje/bylo implementováno takové softwarové řešení, které

umožní komunikaci všech federátů, přičemž v rámci těchto federátů může dojít ke všem požadovaným typům interakcí, synchronizací a předání informací o změně vnitřního stavu atributů.

Výstupy

- Softwarové řešení pro komunikaci mezi federáty.

Fáze F – Propojení lokální a globální části

Cíle

Cíle fáze F jsou:

- propojení lokální a globální části,
- kompletace softwaru,
- tvorba uživatelské příručky.

Způsob dosažení cíle

Pro systémy s homogenní architekturou bude fáze značně omezena na pouhé propojení jednotlivých lokálních částí. Systémy navržené jako heterogenní, a zvláště pak systémy navržené jako heterogenní s centralizovaným uzlem využijí fázi F v celé její škále.
Cíle je dosaženo faktickým nasazením řešení v rámci testovací (plné nebo virtuální) infrastruktury za využití dílčích softwarů implementovaných jako samostatné SW během předchozích fázích. V případě potřeby dojde k úpravě softwaru/rozhraní. Součástí kompletního nasazení bude jeho ozkoušení (zkušební zprovoznění) a tvorba uživatelské příručky.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi F.

Externí dokumenty

- Dokumentace softwaru zajišťující běhové rozhraní.

Interní dokumenty

- Architektonický návrh řešení.
- Programátorská dokumentace.

Software

- Jednotlivé implementace pro federáty.
- Komunikační rozhraní/technologie federace.

Postup

Příprava infrastruktury

Celé řešení se bude zkoušet na infrastruktuře, dle momentálních dispozic řešitelského týmu, přičemž základní klasifikace možností je:

- Provoz na finálním hardwaru v případě, že řešitelský tým je dodavatelem nejen softwarové části, ale zároveň hardwarových komponent.
- Provoz na hardwarové konfiguraci totožné, jako hardwarové specifikace fyzických zařízení, kde bude řešení výsledně provozováno, které nejsou součástí dodávky.
- Provoz na substituovaném hardwaru.
- Provoz ve virtuálním prostředí.
- Provoz v kombinovaném prostředí, kde část hardwaru odpovídá budoucím specifikacím, část hardwaru je substituován, nebo vytvořen virtuálně.

Přičemž výše uvedené body platí pro veškerou informační techniku – výpočetní uzly, vizualizační zařízení, další výstupní zařízení i vstupní systémy/emulátory.

Vlastní příprava infrastruktury by měla reflektovat následující požadavky:

- Zohlednění budoucí provozní infrastruktury, pokud je známa.
- Zohlednění počtu a typů jednotlivých federátů (je třeba zajistit paralelní provoz minimálně jednoho federátu každého typu).
- Specifikace běhového rozhraní/infrastruktury.

Samotná infrastruktura musí být připravena do stavu, ve kterém je možné provést zavedení systému. Tedy, musí proběhnout instalace operačních systémů, běhových rozhraní, ovladačů, nastavení výjimek ve firewallu a případně dalších technických opatření. Doplňkové systémy a hardware je třeba fyzicky připojit a otestovat. Rozhraní externích, nebo online systémů je přípustné vytvořit virtuálně prostřednicím wrapperů s databázovými vstupy, nebo prostřednictvím datových pump (data dump). Během realizace a ověřování fungování propojení není přípustné tyto systémy nahradit konstantní hodnotou.

Zavedení řešení

Propojení jednotlivých softwarových částí je realizováno prostou instalací dílčích částí, které byly implementovány. Právě touto instalací a zprovozněním bude ověřeno, že je vzájemné propojení v pořádku, stejně jako dojde k ověření celého řešení.
Na připravenou infrastrukturu je možné zavést jednotlivé softwarové komponenty. Komponenty se zavádí vždy v poslední dostupné verzi a vždy tak, aby způsob zavádění na připravenou infrastrukturu reflektoval podmínky pro budoucí nasazení ve skutečné infrastruktuře. Pokud je například kladena podmínka, aby dva federáty byly provozovány nezávisle na různých výpočetních uzlech, není možné je v rámci zkoušení zavést na totožný výpočetní uzel.
Po zavedení řešení není třeba tvořit další podmínky – vytížení sítě dalším provozem atp., neboť při zkoušení je možné tyto běhové parametry zanedbat.

Zkoušení řešení

Po úspěšném zavedení je třeba řešení spustit. Spuštění není dokumentováno, stejně jako nedochází k dokumentaci dílčích kroků zkoušení – toto bude provedeno až v následující fázi. Při zkoušení je třeba sledovat především úspěšné zavedení všech softwarových částí řešení a jejich vzájemnou komunikaci/propojení. Při zkoušení je třeba zavést všechny typy softwarových

komponent, i kdyby v běžném provozu nikdy nemělo dojít ke spuštění všech typů komponent naráz. Předmětem zkoušení jsou i hardwarová rozhraní, pokud jsou součástí simulátoru. Pro hardwarová rozhraní je třeba provést nejen zkoušky vůči odpovědnému federátu, ale také vůči celé federaci. Poslední částí, kterou je třeba zkoušet, jsou softwarové části odpovědné za adaptaci na externí rozhraní anebo online systémy.

Tvorba uživatelské dokumentace

V okamžiku, kdy je řešení zavedeno a odzkoušeno, je možné přistoupit k tvorbě uživatelské dokumentace. Uživatelská dokumentace by měla především:

- Popsat základní rozhraní:
  - popsat jednotlivá rozhraní simulátoru,
  - popsat způsob ovládání a jednotlivé ovládací prvky,
  - popsat výstupy z rozhraní.
- Popsat způsob zavedení a spuštění simulátoru:
  - popsat postup spuštění, vč. pořadí spouštění, je-li stanoveno,
  - popsat výběr scénáře a inicializačních kritérií,
  - popsat způsob parametrizace vstupů a procesů,
  - uvést přípustné rozsahy hodnot pro jednotlivé parametry.
- Popsat základní běhové chyby a způsob jejich řešení.
- Uvést povinné zákonné údajů.
- Uvést verze softwaru, pro kterou je uživatelská dokumentace uvedena.
- Uvést verzi certifikace, verze dokumentu a data, pokud k certifikaci softwaru, nebo jeho části došlo.

Před finální úpravou dokumentace je vhodné provést kontrolu dokumentace laikem, nebo členem řešitelského týmu, který na uživatelské dokumentaci nepracoval.

Výstupy

- Zkompletované softwarové řešení,
- Uživatelská dokumentace.

Fáze G – Verifikace

Cíle

Cíle fáze G jsou:

- Verifikovat řešení.

Způsob dosažení cíle

Verifikace je ověření aplikace proti technickému zadání. Tedy, během verifikace dochází k ověření jednotlivých částí řešení, ale i celku řešení vůči návrhům samotného softwarového řešení realizovaného řešitelským týmem. K tomuto ověření dochází různými prostředky, které jsou definovány v rámci jednotlivých bloků této fáze řešení.
Verifikace vychází z ISO/EIC 29119 Software testing z roku 2014. Tato část metodiky byla aktualizována, neboť ISO/EIC v roce 2014 nahradil řadu standardů, jako ISO/IEC 33063: Process Assessment Model (at DIS stage), IEEE 829 Test Documentation, IEEE 1008 Unit Testing a BS 7925-2 Software Component Testing Standard.

Přípustné postupy

Tento blok uvádí doporučení obsahující především výčet základních doporučených metod.

Jednotkové testy

Jednotkové testy patří mezi základní metody testování v oblasti informačních technologií. Každá jednotka (třída/blok tříd) je testována samostatně. Doporučenou formou testování je tvorba přímých testovacích tříd, nebo modulů. Testování probíhá na programové úrovni a má odhalit chyby na implementační úrovni v samotných třídách nebo blocích. Testy jsou specifikovány v IEEE 1008 Unit Testing.

Integrační testy

Integrační testování probíhá na úrovni programovacího jazyka. Cílem integračního testování je ověření funkčnosti rozhraní. Provádění integračních testů má smysl až poté, co jsou úspěšně provedeny jednotkové testy. V opačném případě by nalezení chyby v integračních testech nebylo možné identifikovat co do jejich původu. Testy se zpravidla realizují v programovacím jazyku jako vlastní moduly, nebo testovací třídy. Integrační testy jsou nadstavbou integračních testů a jsou popsány v doporučení ISO/EIC/IEEE 29119.

Testy kalibrace vstupních proudů

Pro simulátory existuje specifická potřeba pro tvorbu zvláštních testů, které slouží ke kalibraci vstupních proudů. Kalibrace probíhá na aplikační úrovni (je tedy třeba mít dokončené softwarové řešení). Dochází ke srovnání vypočtených dat prostřednictvím numerických metod, nebo metod

operačního výzkumu, s praktickými výstupy softwaru (přičemž nastavení softwaru musí být známé a odpovídat hodnotám zadaným pro numerické metody). Porovnání dat získaných z testovaného simulátoru, s daty získanými alternativními prostředky (numerickými metodami, sběrem historických dat), je třeba statisticky vyhodnotit.

Testy kalibrace stochastických procesů

Pro simulátory je relativně specifická potřeba pro tvorbu zvláštních testů, které budou sloužit ke kalibraci jednotlivých stochastických procesů. Tyto testy jsou velmi podobné, jako testy jednotkové. Na rozdíl od testů jednotkových probíhají testy na aplikační úrovni (je tedy třeba mít dokončený software, který prošel příslušným testováním z nižších úrovní). K testování je třeba mít data ze softwaru a data získaná jiným způsobem (např. numericky). Porovnání dat získaných z testovaného simulátoru, s daty získanými alternativními prostředky (numerickými metodami, sběrem historických dat), je třeba statisticky vyhodnotit.

Testy datových proudů

Testy datových proudů jsou standardními testy dle ISO 29119-4, avšak pro potřeby simulátorů dochází k jejich částečné úpravě. Standardní testování probíhá na implementační úrovni. Testování pro simulátory probíhá na aplikační úrovni. Vhodné je provádět testování v několika fázích. V první fázi jsou všechny procesy parametrizovány jako deterministické a je kontrolován průchod entit simulátorem, přičemž je třeba postupně vytvořit také prostředí, které umožní testování každého procesu. Ve druhé fázi je třeba provést obdobné testy se stochastickým nastavením jednotlivých hodnot, v kombinaci s numerickými hodnotami. Porovnání dat získaných z testovaného simulátoru, s daty získanými alternativními prostředky (numerickými metodami, sběrem historických dat), je třeba statisticky vyhodnotit.

Testy mezních vstupních podmínek

Test mezních vstupních podmínek dle ISO 29119 je nutné provést pro zajištění stability řešení v odlehlých, přípustných hodnotách provozu simulátoru. Softwarové řešení musí být otestováno na podmínky, které jsou v krajích přípustných mezích a zároveň musí být provedeny testy mimo přípustné meze. Přípustné meze jsou určeny uživatelskou příručkou a zároveň by měla být omezena softwarovými prostředky. Testy je třeba provádět vzhledem k softwarové příručce dle ISO1063. Mezní testy je tedy možné provádět na implementační rozhraní bez ohledu na uživatelské rozhraní. Uživatelské rozhraní se doporučuje testovat samostatně v rámci funkčního testování dle ISO29119.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi G.

Externí dokumenty

- Dokumenty standardů použitých při verifikaci.

Interní dokumenty

- Programátorská dokumentace.

Ostatní interní materiály

- Software.

Postup

Jednotlivé postupy jsou využitelné opakovaně dle počtu aplikovaných testů. V rámci metodiky je doporučeno provést minimálně jednotkové testy, integrační testy, testy kalibrace vstupních proudů (jako parametrické rozšíření jednotkových testů), testy kalibrace stochastických procesů a testy datových proudů. Některé kroky verifikace (např. kalibrace stochastických procesů) je možné přeskočit za předpokladu, že byl ke konstrukci simulátoru využit hotový software pro rychlé prototypování simulátorů, kde jsou vnitřní funkce tohoto SW již ověřeny.

Příprava řešení

Pro každé kolo testování je třeba provést dokonalou kopii softwaru (nebo simulačního modelu, pokud byl pro konstrukci simulátoru využit software pro rychlé prototypování simulačních modelů). Tato samostatná kopie bude vždy datově uložena samostatně spolu s protokolem o daném testu. Toto opatření zajistí dohledatelnost testovací konfigurace a výsledku příslušného testu. Při úpravě softwaru tak bude relativně snadné zjistit, zda mohl být test ovlivněn úpravou konfigurace.

Provedení testů

Po vytvoření individuální kopie je možné přikročit k provedení testů. Veškeré softwarové úpravy musí být provedeny před zahájením testů (např. nastavení násady pro generátory pseudonáhodných čísel, výběr souboru pro test datových proudů atp.). V případě, že by bylo třeba jeden test třeba provést s více různými nastaveními, je třeba pro každý takový dílčí test provést vlastní přípravu testu, vlastní provedení a uložit test s vlastním protokolem.

Tvorba protokolu z testování

Pro provedení testu je třeba vždy formalizovat výsledky testu do samostatného protokolu – to jest provést zaznamenání testu. Tento protokol odpovídá požadavkům na jakost softwaru dle ISO 25010:2011. Archivace protokolu je povinnou součástí postupu, který podléhá jakémukoli stupni kvality řízení dle ISO 9000, resp. ISO 9001. Ideální (i když ne nezbytnou) formou pro evidenci testování odpovídá specifikaci IEEE829.

Protokol z testování by měl obsahovat minimálně (ne však výhradně) následující údaje:

- uvedení testované části softwaru,
- uvedení testovací metody,
- uvedení verze testu, pokud byla metoda použita s více různými parametry,
- uvedení verze testované/testovaných částí softwaru,
- výsledky testů a případná doporučení k testování,
- odkaz na překryté testy (předchozí neúspěšné testy, které aktuální testy nahrazují),
- datum provedení testů,
- identifikace odpovědné osoby za provedení testů.

Výstupy

V kapitole jsou uvedeny výstupy fáze G.

- Protokoly jednotlivých testů, vč. archivovaných částí softwaru.

Fáze H – Validace

Cíle

Cíle fáze H jsou:

- provedení validační testů,
- provedení závěrečného akceptačního testu.

Způsob dosažení cíle

Validace je poslední fází testování softwaru/zařízení, při které dochází k ověření veškerých funkcionalit. Na rozdíl od verifikace je ověřování provedeno oproti požadavkům zadavatele. Validace tak, na rozdíl od verifikace, odhalí také chyby v návrhu aplikačního řešení.
Cíle může být dosaženo jednofázově, nebo dvoufázově. Povinnou poslední fází je vždy provedení akceptačních testů proti dokumentaci stanovené během fáze A. Volitelnou činností během fáze je stanovení dalších, nadnormativních validačních testů. Doporučené testy, kterými je možné zvýšit jakost softwaru, nebo mohou být přímo vyžadovány při tvorbě simulátorů pro zvláštní určení, jsou uvedeny dále.

Přípustné postupy

Kapitola obsahuje především výčet základních doporučených validačních testů, které jsou použitelné v rámci metodiky. Libovolně je možné využít další testovací metodiky.

Expertní testování

Expertní testování patří k základním a pro autorský tým triviálním způsobům testování. Metoda je vhodná především v situaci, kdy je potenciální cesta průchodu entity simulátorem široce větvená a téměř každá entita prochází rozdílnými procesy, oproti další entitám, nebo jde o simulátor typu trenažér. Nevýhodou jsou personální předpoklady, neboť zajištění experta v oboru, který může vlastním testováním provádět validaci je často netriviální, nebo dokonce nemožné. Dalším významným problém je ten, že tato metoda testování není postačující metodou pro prohlášení celého systému za validní. Tuto nevýhodu jde obejít kombinací expertní metody a metody akceptačních kritérií – což však v tomto konkrétním případě není možné, neboť akceptační test je povinnou součástí plnění metodiky. Metoda expertního testování by měla odpovídat testování dle kapitoly 5.4 předpisu ISO/EIS/IEEE 29119-4.

Scénářové testování

Pro tvorbu scénářového testování je třeba v přípravné fázi vytvořit takový scénář simulátoru, pro který platí, že pro předem specifikované vstupy jsme schopni jednoznačně (se započtením směrodatné odchylky) určit hodnoty jednotlivých výstupů. Scénářové validační testy nejsou postačující metodou pro prohlášení celého systému za validní. Jsou však velmi často používanou validační metodou, neboť tvorba scénářů je pro řadu simulátorů relativně triviální. Existují však

systémy, pro které není možné reprodukovatelný scénář možné vytvořit (např. trenažéry), v takovém případě je nutné nasadit jiné metody testování. Scénářové testování je definováno v kapitole 5.2.9 předpisu ISO/EIS/IEEE 29119-4.

Ekvivalentní testování

Pro ekvivalentní testování je nezbytné mít k dispozici alternativní model k odpovídajícímu testovanému systému se známými vstupními parametry a interpretovatelnými výstupními parametry. Parametrizace validovaného systému pak musí být nastavena dle ekvivalentního systému. Ekvivalentní systém musí být považován za validní. Tedy, není možné tímto způsobem validovat dva nevalidní systémy. V případě, že pro stejné vstupní parametry budou u validovaného i ekvivalentního systému odpovídat výstupní hodnoty, je možné považovat systém za validní. Rozsah potenciální validovatelnosti celého řešení je závislý na možné škálovatelnosti ekvivalentního systému. Ekvivalentní validace (dána specifikací v kapitole 5.2.1 předpisu ISO/EIS/IEEE 29119-4) je velmi výhodná především v kombinaci s technikou kontroly všech dostupných stavů daných kapitolou 5.2.5.5 předpisu ISO/EIS/IEEE 29119-4. Tedy za předpokladu, že v ekvivalentním systému je možné nastavit vstupní hodnoty dle předpisu 5.2.5.5., můžeme ekvivalentní testování považovat za dostatečné testování pro validitu celého řešení.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi H.

Externí dokumenty

- Dokumenty standardů použitých při validaci.

Interní dokumenty

- Specifikace uživatelských požadavků,
- uživatelská dokumentace.

Ostatní interní materiály

- Software.

Postup

Postup prvních kroků (které jsou interaktivní) odpovídá testovacímu postupu pro verifikaci. Další kroky jsou nové a mají sloužit zároveň k přípravě dokumentů pro zadavatele.

Příprava řešení

Pro každé kolo testování je třeba provést dokonalou kopii softwaru (nebo simulačního modelu, pokud byl pro konstrukci simulátoru využit software pro rychlé prototypování simulačních modelů). Tato samostatná kopie bude vždy datově uložena samostatně spolu s protokolem o daném testu.

Toto opatření zajistí dohledatelnost testovací konfigurace a příslušného testu. Při úpravě softwaru tak bude relativně snadné zjistit, zda mohl být test ovlivněn.

Provedení validačních testů

Po vytvoření individuální kopie je možné přikročit k provedení testů. Veškeré softwarové úpravy musí být provedeny před zahájením testů (např. nastavení násady pro generátory pseudonáhodných čísel, výběr souboru pro test datových proudů atp.). V případě, že by bylo třeba jeden test provést s více různými nastaveními, je třeba pro každý takový dílčí test provést vlastní přípravu testu, vlastní provedení a uložit test s vlastním protokolem.

Tvorba protokolu z testování

Pro provedení testu je třeba vždy formalizovat výsledky testu do samostatného protokolu. Tento protokol odpovídá požadavkům na jakost softwaru dle ISO 25010:2011. Archivace protokolu je povinnou součástí postupu, která podléhá jakémukoli stupni kvality řízení dle ISO 9000, resp. ISO9001. Ideální (i když ne nezbytnou) formou pro evidenci testování odpovídá specifikace IEEE829.

Protokol z testování by měl obsahovat minimálně (ne však výhradně) následující údaje:

- uvedení testované části softwaru,
- uvedení testovací metody,
- uvedení verze testu, pokud byla metoda použita s více různými parametry,
- uvedení verze testované/testovaných částí softwaru,
- výsledky testů a případná doporučení k testování,
- odkaz na překryté testy (předchozí neúspěšné testy, které aktuální testy nahrazují),
- datum provedení testů,
- identifikace odpovědné osoby za provedení testů.

Akceptační validační test

K akceptačnímu testování je možné přistoupit až potom, co dojde k úspěšnému průchodu všech testů specifikovaných předchozími bloky fáze validace, pokud byly stanoveny.
Akceptační validační test proběhne dle předpisu v dokumentu *Specifikace uživatelských požadavků*. Jde o kriteriální test, kdy kritéria jsou stanovena právě v dokumentu *Specifikace uživatelských požadavků*. Z testování je proveden zápis, který má formu tabulky, kde položkami tabulky jsou jednotlivá kritéria přijatelnosti a hodnotami tabulky údaje o splnění, nebo nesplnění akceptačního testu. V případě, že je jediné kritérium nesplněno, je test označen jako nesplněný. V případě opakování testu je nutné celý test opakovat znovu. Výstupem testování je dokument Souhrnný protokol akceptačních testů, který se stane nedílnou přílohou při předání díla.

Výstupy

V kapitole jsou uvedeny výstupy fáze H.

- Protokoly jednotlivých testů, vč. archivovaných částí softwaru.
- Souhrnný protokol akceptačních testů.

Fáze I – Předání

Cíle

Cíle fáze I jsou:

- nasazení softwarového díla,
- nasazení řešení pro údržbu systémů,
- souhrnný protokol akceptačních testů,
- legální ukončení řešení.

Způsob dosažení cíle

Fáze I se soustředí na dokončení vlastního díla, legální ukončení řešení, nasazení řešení a případné nasazení pro vzdálenou údržbu systému.
Fáze legálního ukončení je formální, neboť po faktickém přijetí akceptačních kritérií během fáze G již prakticky není možné dosáhnout odmítnutí řešení ze strany zadavatele.
Klíčovým partnerem je, vyjma zadavatele, také instituce provozovatele simulátoru, u které dojde k nasazení kompletního řešení. Nasazené řešení je následně otestováno. Po úspěšném otestování (fáze G a fáze H) metodika doporučuje nasazení softwarových prostředků pro vzdálenou údržbu systémů tak, aby další servis bylo možné provádět distančně.
Celá fáze je realizovatelná sekvenčně, při respektování jednotlivých kroků během fáze. Tak jako v případě ostatních fází, v případě adaptace metodiky, nebo splynutí některých rolí, je možné některé kroky vynechat. V takovém případě dochází k sekvenčnímu postupu dále na nejbližší krok, který nebyl ve fázi vynechán.

Vstupy

V kapitole jsou uvedeny vstupy pro Fázi I.

Interní dokumenty

- Validační scénář,
- soupis technických komponent simulátoru,
- technická příručka pro simulátor,
- uživatelská příručka pro ovládání řešení.

Ostatní interní materiály

- Software simulátoru.

Externí dokumenty

- Licence softwaru pro vzdálenou údržbu.

Postup

Příprava nasazení

K faktické migraci simulátoru z laboratoří řešitelského týmu do instituce provozovatele simulátoru musí dojít vždy ve spolupráci s institucí provozovatele simulátoru. Forma a rámec této spolupráce měly být definovány již během fáze A, při přípravě smluvní dokumentace. Pokud k tomu nedošlo, je třeba tyto vztahy specifikovat v této fázi.

Při přípravě nasazení je nutné, kromě vyjasnění právních vztahů, vyřešit také otázky:

- instituce odpovědné za nasazení hardwaru,
- instituce odpovědné za nastavení hardwaru,
- podmínek přístupu do vnitřní sítě provozovatele simulátoru,
- termínů pro nasazení řešení,
- klíčových parametrů pro technické zajištění vzdálené správy systémů.

Technická příprava nasazení

Během technické přípravy nasazení je nutné vycházet z dokumentu *Soupis technických komponent simulátoru*, který předepisují zařízení, kterými je nutné pracoviště vybavit. Tato zařízení je třeba usadit, připravit a zakomponovat v souladu s dokumentem *Technická příručka pro simulátor*.
Součástí technické přípravy je také softwarová příprava z hlediska programů nesouvisejících se samotným simulátorem. Tímto jsou myšleny především instalace operačních systémů.

Technologická příprava nasazení

Po uskutečnění technické přípravy je možné zahájit prvotní nastavení souvisejících systémů. Jde především o zapojení technických řešení a jejich vzájemné softwarové testování. V případě komunikace jednotlivých výpočetních uzlů a případného hardwaru je možné přejít v testování dál k propojení jednotlivých výpočetních uzlů. Toto testování vyžaduje kontrolu nastavení sítě, jejích přenosových rychlostí a především parametrů (latence, přenosová rychlost). V případě, že to *Technická příručka pro simulátor* vyžaduje, je třeba připravit nastavení QoS, nebo vyhrazení linek v případě běhu simulačního výpočtu. V případě, že jsou Technickou příručkou pro simulátor požadovány další softwarové úpravy, je třeba tyto provést. Primárně se jedná o instalaci běhových prostředí pro simulace (např. RTI).
Součástí kroku technologické přípravy je také příprava prostředí na řešení pro vzdálenou správu systémů. Primárně jde o zajištění vzdáleného přístupu pro software v souladu s pravidly instituce provozovatele simulátoru – zřízení VPN, vyhrazeného datové kanálu, nebo prosté povolení nastavení softwaru ve firewallech.

Provedení nasazení

Vlastní nasazení softwaru simulátoru provede Řešitelský tým na připravenou infrastrukturu. Instalace proběhne podle *Technické příručky pro simulátor* tak, aby tento proces byl

reprodukovatelný. Provedení nasazení bude úplné – to znamená, že nasazení musí proběhnout na každý uvažovaný výpočetní uzel v případě potřeby běhu nejvyššího uvažovaného množství výpočetních uzlů. Toto nasazení je třeba realizovat i za předpokladu, že v běžném provozu nebude takové množství výpočetních uzlů prováděno. V tomto kroku je možné provést nasazení rezervních výpočetních uzlů, pokud je takové nasazení požadováno.

Kontrola nasazení

Po provedení nasazení je třeba nasazení ověřit. Pro toto ověření je třeba spustit na simulátoru validační scénář. Pro tento scénář bude sledován jeho úspěšný průběh, stejně jako provedena komparace předpokládaných výsledků s výsledky skutečnými.
V případě, že validační scénář neproběhne korektně, je třeba provést analýzu příčin tohoto neúspěchu. Na základě analýzy je třeba provést úpravy na úrovni hardwaru nebo softwaru a znovu provést kompletní kontrolu nasazení.
Součástí kontroly je také provedení všech kontrol s upraveným nastavením tak, že uzly, pro které je připraven rezervní uzel, jsou těmito rezervními uzly nahrazeny a je provedena kontrola s nimi. Po nahrazení základních uzlů uzly rezervními již nedojde ke zpětné výměně (tj. v ostrém provozu zůstávají rezervní uzly – vyjma případu, kdy toto neumožňuje hardwarová konfigurace a rezervní uzly nejsou hardwarově rovnocenné s primárními uzly).
Po kontrole nasazení je vystaven dokument *Protokol o provedení nasazení systému*, který vyžaduje legalizaci ze strany Instituce provozující simulátor.

Zavedení softwaru pro údržbu systémů

V okamžiku, kdy dojde k úspěšnému nasazení systému, je možné uvažovat nasazení servisních softwarů, nebo softwarů pro vzdálenou údržbu systému (tím je myšlen simulátor, operační systémy, software běhových rozhraní atp.). Implicitním předpokladem metodiky je to, že instituce zajišťující provoz simulátoru má k dispozici příslušné licence k instalovanému servisnímu software.

Požadavky kladené na servisní software jsou především:

- distribuce servisního softwaru na stejný výpočetních uzlech, na kterých je prováděn simulátor,
- distribuce servisního softwaru na výpočetním uzlu, na kterém je realizováno běhové prostředí pro simulátor (např. RTI),
- umožnění vzdálené správy výpočetních uzlů,
- umožnění správy a sledování sítě a jejího provozu (mimo stav offline),
- omezení, že při zásahu servisního softwaru nedojde k přerušení či ovlivnění prováděných výpočetních úloh,
- umožnění aktivovat software vzdáleně, nebo jeho neustálý běh v režimu démon.

Výše zmíněným požadavkům nevyhovují implicitní softwarové prostředky operačních systémů, v OS Windows – vzdálená plocha, která přeruší provádění výpočtu, nebo vyžaduje přímé potvrzení a povolení správy – není ji tedy možné aktivovat distančně a pokud ano, dojde k přerušení aktuálních výpočetních úloh. Proto není možné při použití pouhých prostředků OS

prohlásit podmínku za splněnou. Vyhovujícími prostředky jsou prostředky pro vzdálený přístup do OS Linux. Doporučeným softwarem třetí strany je například software TeamViewer od stejnojmenné společnosti (www.teamviewer.com, 2003).
Po zavedení systému na údržbu je třeba předat dokument *Protokol o nasazení systémů pro údržbu*. V tomto protokolu budou, kromě základních údajů o nasazení, údajích o distribuci, pojmenování jednotlivých servisních uzlů, také uvedeny klíčové přihlašovací a přístupové údaje do servisního softwaru.

Legalizace ukončení řešení

Pro úspěšném zavedení systému a po úspěšném ukončení všech předchozích kroků je možné provést formální ukončení řešení celého projektu. Toto ukončení je třeba opět provést legální formou s kontrasignací zadavatele.
Základním východiskem pro předání celého díla je kompletace předchozích dokumentů pro předání jednotlivých fází, potvrzené validační testy a potvrzené informace o nasazení celého řešení.
Dokument *Dodací list* legalizovaný všemi povinnými smluvními stranami formálně ukončuje projekt. Po dodání *dodacího listu* je nutné zajistit přípravu daňových dokladů pro dodavatele. K faktickému ukončení projektu dochází okamžikem úhrady vystavených daňových dokladů (faktur).

Výstupy

- Protokol o provedení nasazení systému,
- protokol o nasazení systémů pro údržbu,
- dodací list.

Novost postupů

Existence novosti postupů (tedy originálních postupů, přístupů a aplikovaných technik ve vybrané aplikační doméně) je nutnou podmínkou pro to, aby mohl být výstup klasifikován jako výstup VaVaI. *Metodis* splňuje tato kritéria v následujícím:

- **Unikátnost** znamená, že metodika je v rámci své aplikační domény jedinečná. Pro *Metodis* platí, že v ČR není dostupná další metodika, která by byla zaměřena na budování distribuovaných simulátorů.
- **Technologická volnost** znamená, že metodika umožňuje použití větší škály technologií, než je obecně (v zahraničních metodikách) přípustné. To je způsobeno tím, že *Metodis* vznikla s cílem vytvoření metodiky použitelné pro konkrétní účel (tj. budování simulátorů). Zahraniční metodiky vznikaly téměř výhradně s pevnou vazbou na konkrétní implementační technologie (např. metodika budování simulátorů prostřednictvím HLA, metodika budování simulátorů prostřednictvím DIS).
- **Zobecnění osvědčených principů** je jedním z důsledků předchozího bodu. Přestože není metodika pevně navázána na žádnou implementační technologii, je silně inspirována dobrými, ověřenými a fungujícími principy používanými v různých existujících technologiích. Toto platí například pro separaci tří úrovní simulátoru, kde *Metodis* přebírá schéma používané v HLA. Toto schéma navíc zobecňuje i pro použitelnost v jiných implementačních technologiích.
- **Doporučení ověřených technologií a technik** metodika umožňuje díky množství případových studií, které vzniky při/po tvorbě metodiky. Díky tomu bylo již během psaní metodiky možné procházet jednotlivými procesy (na úrovni fází, bloků i kroků) a optimalizovat je vzhledem k potřebám praktického použití.
- **Zaměření i nad rámec implementace simulátoru**, které uplatňuje ucelený pohled na celé řešení simulátoru, nikoli jen jeho softwarové části. To umožňuje aplikovat metodiku pro celý projekt, od okamžiku vymezení simulovaného systému až po faktické předání simulátoru.

Metodika *Metodis* přináším svým uživatelům především následující přínosy:

- **Usnadnění vytvoření a řízení řešitelského týmu**. *Metodis* umožňuje postavit řešitelský tým na pouze jedné osobě kvalifikované (znalé) v oblasti simulace, a dalších osobách u kterých postačuje znalost z oblasti tvorby softwaru. Právě respektování metodikou předepsaných postupů umožňuje izolovat jednotlivé práce tak, aby nebylo třeba využívat pouze vysoce kvalifikovaných specialistů. Zároveň se *Metodis* zabývá přístupem k řízení řešitelského týmu a navrhuje konkrétní řídicí metody týmu. Z této vlastnosti může opět řešitelský tým profitovat, neboť metodika, do určité míry, snižuje rizika způsobená nevhodným řízením projektu.
- **Třívrstvý logický model** separuje v simulátoru tři samostatné logické úrovně. Toto rozdělení značně usnadňuje analytickou a architektonickou fázi řešení, neboť jedna úroveň dekompozice je již provedena samotnými principy metodiky. Zároveň zvolené rozdělení vychází z dobré praxe v jedné, z potenciálních implementačních technologií (HLA). Zobecněním vznikl model, který je značně intuitivní v případě snahy využít některou konkrétní implementační technologii. Naprosto přirozené je rozdělení při použití HLA. Velmi intuitivní převod třívrstvého logického modelu je na implementační schéma DIS (s centralizovaným, i bez centralizovaného prvku).

- **Výhody plynoucí z nasazení metodiky** jsou výhody, které mají relativně obecný charakter. Definovat je lze při srovnání situace, kdy libovolná činnost probíhá za využití unifikovaného postupu (metodiky), nebo bez něj. Současný stav vědeckého poznání podporuje unifikaci / standardizaci. Pro metodiku jde především o výhody plynoucí z principů Zaznamenání a uchování postupu, Předvídatelnosti výstupů, Jednotného systému řízení a Jednotného systému vyhodnocení.

Seznam literatury

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

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

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.

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.

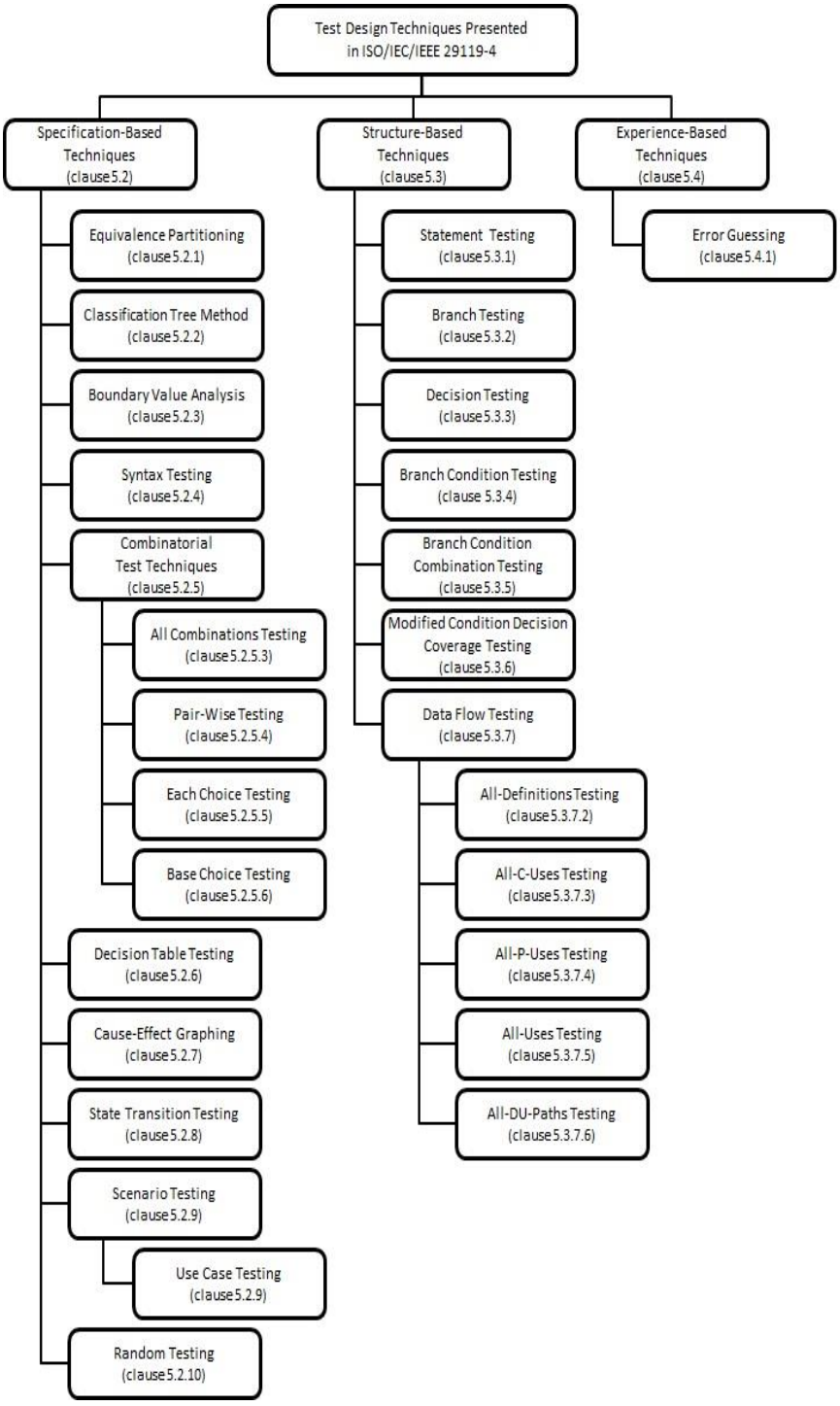
TeamViewer - Vzdálená podpora, vzdálený přístup, servisní služba, online spolupráce a schůzky [online]. 2003 [cit. 2014-07-11]. Dostupné z: https://www.teamviewer.com

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/C.13c. ISBN: 1-937218-33-1.

WITMER G., 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.

Příloha A: Struktura testů



Obrázek A.1: Struktura testů[2]

[2] Zdroj: ISO29119

Příloha B: Vybrané způsoby dekompozice logických procesů

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.
Vhodná forma rozkladu jest rozklad dle přirozených příčinných souvislostí. Na každou část modelu tak připadají (potenciálně) všechny funkce a mohou jimi (potenciálně) procházet všechny entity. Avšak logicky je model rozložen na více částí.

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 B.1. 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 B.1: Ukázka dekompozice[3]

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

[3] Zdroj: vlastní

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

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.

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.

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.
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.

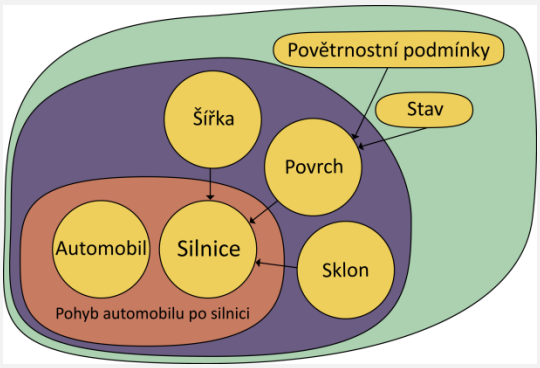
Dekompozice na mikroprocesy

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).
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.

Tvorba kompozitního simulátoru

Kompozitní simulátory jsou postaveny tak, aby byly postupně rozšiřitelné. Při jejich návrhu a implementaci je použit tzv. inkrementální přístup. Je tak možné velice snadno škálovat jednotlivé výstupní parametry na základě výsledků předchozích experimentů. Při implementaci takových řešení je třeba důsledně řešit problematiku vůči rozhraní. Každá funkcionalita je pak vlastně (potenciálně) vnořeným modelem.

Jednoduchý příklad je na obrázku B.2: 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 opatření je však možné jednotlivé parametry zlepšovat. Na obrázku B.2 je červeně vyznačen základní model, modře nižší míra abstrakce první úrovně, zeleně nižší míra abstrakce druhé úrovně. Tato opatření však byla realizována vždy jen nad jedním objektem. Základní parametry silnice je možné rozšířit o sklon, typ povrchu a šířku silnice. Výpočet pak neproběhne v objektu silnice, ale proběhnou tři delimitované výpočty. Při simulačním výpočtu pak bude bráno v potaz mnohem více parametrů a celý model může uplatňovat nižší úrovně abstrakce.



Obrázek B.2: Kompozitní simulační systém[4]

[4] Zdroj: vlastní

Příloha C: Vybrané běhové infastruktury

Příloha slouží k vytvoření přehledové znalosti vybraných běhových infrastruktur. Tato přehledová znalost nemůže být dostatečná pro úspěšnou implementaci simulátoru. Je však dostatečná pro možnost rozhodnout se, jakou běhovou infrastrukturu zvolit.

HLA

High Level Architecture (dále jen HLA) je velmi silná metodika pro vytváření a provoz distribuovaných simulací. Norma neomezuje aplikační doménu, způsob modelování, to jest je umožněno modelování prostřednictvím logických procesů pomocí principu dekompozice logických procesů atp. Na nižších úrovních HLA specifikuje způsob komunikace mezi jednotlivými uzly distribuované simulace, přičemž požadavky klade pouze na samotný přenosový formát (jde o komunikaci prostřednictvím XML). Díky tomu je možné vytvářet softwarově i hardwarově heterogenní simulační systémy (např. část implementovaná v jazyce Java, část v jazyce C), případně pomocí konverze vstupně výstupních dat do XML zapojovat již existující simulátory.
V HLA je nutné veškeré elementy (v tomto odstavci rozumějme objekty, atributy, interakce a parametry) definovat předem v takzvaném OMT (Object Model Template). Samotné OMT je strukturováno do dvou částí: FOM a SOM.
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.
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).
Při charakterizování architektury HLA je nutné mít na paměti, že její nejvytíženější komunikační částí je právě RTI. 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. Je proto třeba důsledně rozlišovat úroveň pohledu na celou architekturu.
Na fyzické úrovni má RTI dvě důležité části; první z nich je vlastní aplikace RTI, 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. Druhou částí je lokální RTI komponenta. Jde o knihovnu, kterou je nutné přilinkovat k vlastnímu softwarovému řešení, nebo aplikačnímu wrapperu. 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.

DIS

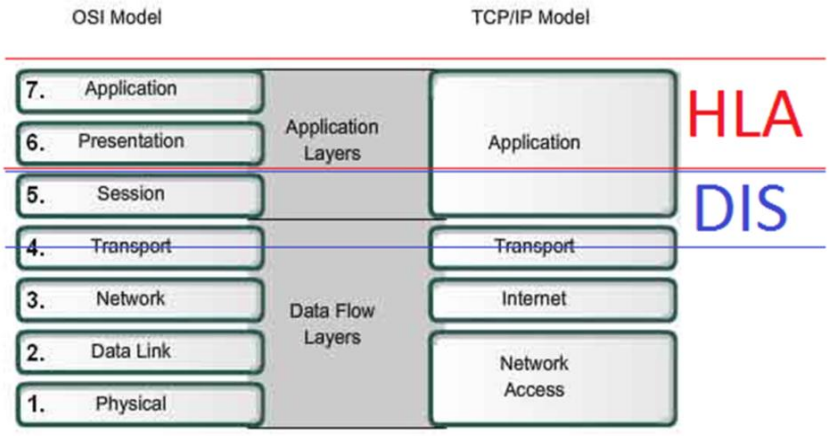
DIS, čili distribuovaná interaktivní simulace (původně, anglicky Distributed Interactive Simulation), je technologie, která vznikla historicky dříve než HLA, ale zároveň je dlouhodobě vyvíjena paralelně s HLA. Historicky šlo dokonce o úspěšnější technologii, neboť je na rozdíl od HLA definována na daleko 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 a historicky také několika armádními normami (např. STANAG 4484ED, SIMPLE STANAG 5602), a to v USA i Evropě, kde byla hojně používána, než byla nahrazena právě HLA.

Základní určení DIS je podpora tvorby 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 Family) jsou společně například všechny žádosti o vstup do procesu, výstup z procesu, problematiku logiky přesunu aj. 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 se zvýšila efektivita, 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, tedy PDU Family nese data, která jsou společná pro všechny členy 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.



Obrázek C.1: DIS a HLA v OSI a TCP modelu[5]

[5] Zdroj: vlastní

Protože jde o nízkoúrovňové řešení, bylo nahrazeno v praktických použitích právě HLA architekturou. Na obrázku C.1 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 (nízkoúrovňové řešení DIS jsou pro kodéry pracnější a časové náročnosti jsou až násobně vyšší než je tomu u HLA).

Programovací přístupy a paradigmata běhových infrastruktur

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 nasadit vlastní řešení lišící se 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. 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 (Interface) a vytvoření bran (Gate) pro jiné technologie, neboť struktura dat je předem jasně dána.

Mezi způsoby, jak budovat distribuované simulátory na nejnižší (implementační) úrovni patří také softwarová rozhraní pro vzdálené volání metod a funkcí.
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 simulátor 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.
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.
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.
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 tak chová, jako kdyby vzdálené objekty byly místními. CORBA zajišťuje, aby pro aplikaci bylo toto chování zajištěné a transparentní.
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.

Ani jedna z těchto technik však není primárně určena k tvorbě řešení s vysokou mírou interakcí mezi federáty a proto se příliš nehodí k tvorbě rozsáhlých 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.

Příloha D: Vybrané techniky pro interaktivní zásahy

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), (Granino, 2016), (Granino, 2010) 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.

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 kausální změně o velikost 0,8 až 1,2 času (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.

Pro online systémy (obou typů) platí, že proto, aby byla simulace reprodukovatelná, je nutné (kromě řádné implementace technik pro interaktivní zásahy) vždy evidovat konkrétní interakci.

Samotná technika interaktivních zásahů se může odlišovat. Interaktivní systémy můžeme klasifikovat na systémy tvrdě interaktivní a měkce interaktivní. Pro tvrdě interaktivní systémy existuje velmi pevná vazba mezi uživatelským vstupem a důsledky těchto vstupů. Literatura uvádí, že přijatelné latence uživatelských vstupů a projekce těchto vstupů do parametrů je 0,2 s reálného času. Tedy, od okamžiku uživatelského zásahu do provedení projekce uživatelské

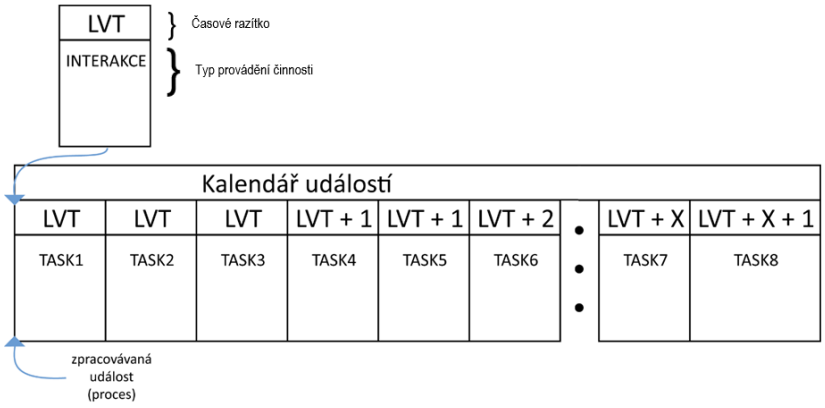
interakce do stavového prostoru simulátoru nesmí uplynout více než 0,2 sekundy. Tvrdě interaktivní systémy jsou charakteristické tím, že jejich simulační čas má pevnou vazbu s reálným časem, přičemž požadavek na tyto systémy jsou vazby odpovídající kausální změně o velikosti 0,9 až 1,1 reálného času (tj. čas v simulaci může krátkodobě plynout rozdílnou rychlostí, a to nejméně rychlostí 0,9 a nejvíce 1,1násobku reálného času).
Měkce interaktivní systémy jsou takové interaktivní systémy, u nichž není nutné požadovat striktní dodržení podmínek pro tvrdě interaktivní systémy. Neplnění tohoto požadavku může mít zpravidla tři základní příčiny:

1. Kauzální změny v simulujícím systému jsou tak rychlé že zvýšení, nebo snížení latence nemůže být uživatelem intuitivně identifikováno.
2. Simulátor funguje s vazbou na reálný čas, avšak uživatelské procesy svým charakterem (vysokou dobou propsání důsledku změny do části simulátoru, kterou může uživatel sledovat) překryjí absenci pevné vazby.
3. Systém nemá přímé kauzální vazby.

Po definici systémů a požadavcích na jejich vazby je možné uvést jednotlivá implementační řešení.
V níže uvedených částech je uveden ilustrativní příklad s kalendářem událostí. Je nutné podotknout, že implementace je možnost provést i na systémech nezaložených na kalendářích.

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), 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 na obrázku D.1.
K řádnému pochopení obrázku je třeba uvědomění, že TASK1 je aktuálně zpracovávaná úloha v čase LVT.

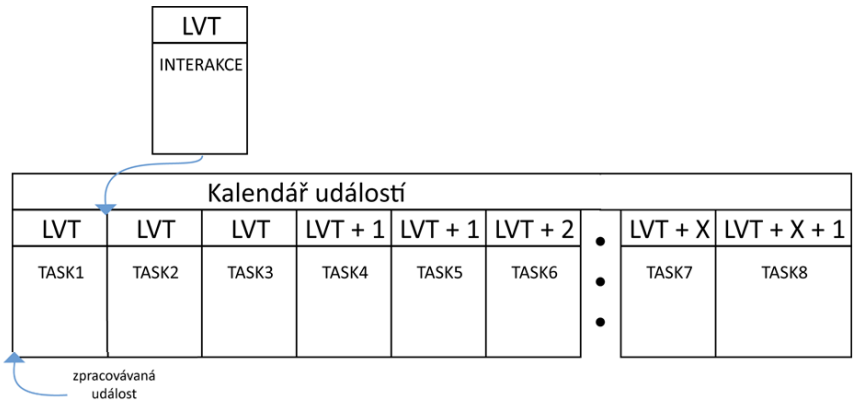


Obrázek D.1: Interaktivní systém s přerušením[6]

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.

Interaktivní zásahy bez přerušení

Nejjednodušší technikou je aplikace interaktivních zásahů bez přerušení. Na rozdíl od techniky 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 D.2. 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.



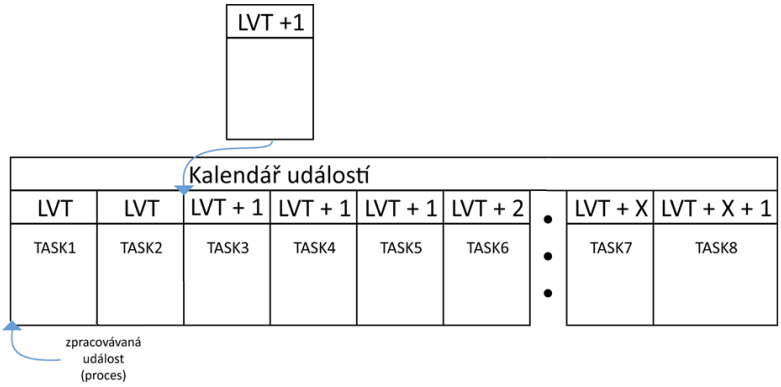
Obrázek D.2: Interaktivní zásah bez přerušení[7]

[6] Zdroj: vlastní
[7] Zdroj: vlastní

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 reakčního času 0,2 s zároveň platí, že jde o věc, kterou lze uživatelsky přejít bez povšimnutí.

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 (pro aktuální simulační čas). 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. 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 D.3.

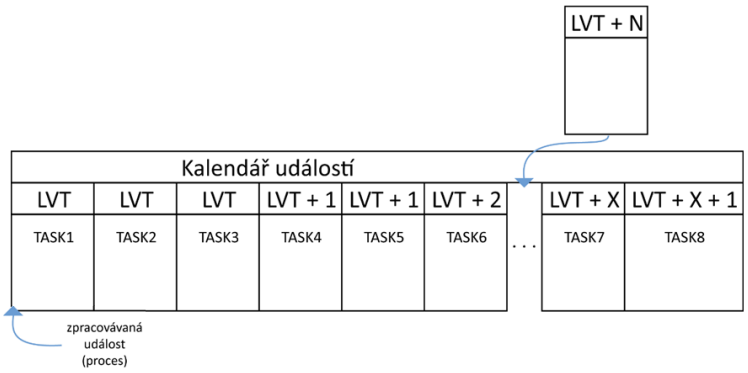


Obrázek D.3: Interaktivní zásah s elementárním odstupem[8]

Interaktivní zásah s výhledem

Poslední ze základních technik interaktivních zásahů je plánování interaktivního zásahu s výhledem. 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 realizaci této projekce až v čase pozdějším. Řešení je využitelné zvláště v případech, kdy simulovaný systém svou povahou, 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 čas *N* je možné vidět na obrázku D4.

[8] Zdroj: vlastní



Obrázek D4: Interaktivní zásah s výhledem[9]

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, které však má veškeré výhody vazby těsné (snazší verifikace a validace, lepší předvídatelnost, plnění ISO norem na jakost softwaru pro bezpečnost atp.).

[9] Zdroj: vlastní