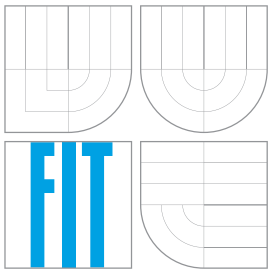
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ANALÝZA RIZIK JAKO DISTRIBUOVANÝ INTELIGENTNÍ SYSTÉM

RISK ANALYSIS AS AN DISTRIBUTED INTELLIGENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VOJTĚCH ORGOŇ
AUTHOR

VEDOUCÍ PRÁCE — Ing. FRANTIŠEK ZBOŘIL, Ph.D.
SUPERVISOR

BRNO 2007

Zadání

Analýza rizik jako distribuovaný inteligentní systém

1. Seznamte se s problematikou analýzy rizik a multiagentních systémů
2. Navrhněte model pro analýzu rizik jako systému s autonomními inteligentními entitami – agenty
3. Specifikujte role agentů v systému a navrhněte model komunikace.
4. Implementujte agenta v jazyce Java a ověřte funkčnost vašeho návrhu
5. Zhodnoťte přínos a dosažené výsledky.

Kategorie: Umělá inteligence

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně

Abstrakt

Tento text pojednává o použití multiagentového systému v analýze rizik. Po vysvětlení pojmů analýza rizik a multiagentový systém práce pojednává o návrhu a implementaci takového systému. Snahou je určit, zda a za jakých podmínek je vhodné nasadit multiagentový systém do analýzy rizik.

Klíčová slova

Analýza rizik, distribuovaný systém, multiagentový systém, aktiva, hrozby, zranitelná místa, protiopatření, rizika, chování, role agenta, zasílání zpráv, JADE, ACL

Abstract

This text describes use of the multiagent system in the risk analyses. After account of term risk analyses and multiagent system this work describes the proposal of such system. Final work is usability the multiagent system in the risk analyses.

Keywords

Risk Analyses, distributed system, multiagent system, asset, threat, vulnerability, countermeasure, risk, behaviour, agent role, sending messages, JADE, ACL

Citace

Vojtěch Orgoň: Analýza rizik jako distribuovaný inteligentní systém, bakalářská práce, Brno, FIT VUT v Brně, 2007

Analýza rizik jako distribuovaný inteligentní systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Zbořila, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vojtěch Orgoň
14. května 2007

Obsah

Kapitola 1

Úvod

Riziko. Slovo, skloňované snad v každé kapitole této práce, ale také v běžném životě. Nepřijatelné riziko, riziko podnikání, neriskuj, to riziko za to stojí. Známe jej z běžné mluvy jako dosti vágní pojem, který má většinou negativní význam.

Cílem postupů uvedených v kapitole 2 je snížit riziko za co nejlepších podmínek. Skládá se ze dvou částí – určení rizik a nalezení nejvýhodnějších protiopatření, které mají za úkol rizika eliminovat.

Najdete zde definici rizika, popis způsobů odhalování rizik pomocí hrozeb a zranitelných míst, jak rizika eliminovat a nakonec popis protiopatření. Dále je zde naznačeno, které části a za jakých podmínek lze automatizovat.

V další části mé práce (kapitola 3) se zabývám multiagentovými systémy (MAS), jedním ze způsobu, jak *analýzu rizik* podporovat pomocí informačních technologií. Nejde o popis celé problematiky MAS, ale jen vysvětlení postupů, které využívám v dalších kapitolách.

Kapitola 4 je pokusem navrhnout systém pro podporu analýzy rizik. Snažím se určit hlavní části takového systému a důležité komunikační protokoly. V návaznosti na to kapitola 5 doplňuje části návrhu, které jsou nutné pro provoz systému, ale nemají obecný charakter. Rozhodnutí učiněná v této kapitole se budou lišit v různých implementacích systému.

Kapitola 2

Analýza a management rizik

Úkolem této kapitoly je představit způsob vyhledávání a eliminaci rizik. První část, která se nazývá *analýza rizik*, se snaží popsat systém a určit možná rizika. Ve druhé části, nazvané *management rizik*, je úkolem zavést do systému takové změny (*protiopatření*), které sníží rizika pod přijatelnou úroveň.

Je třeba upozornit, že veškeré informace uvedené v této kapitole pocházejí z [2] a [7].

2.1 Slovníček pojmů

Každá lidská činnost si přizpůsobuje význam obecně používaných slov, nebo si vytvoří slova vlastní. Analýza rizik není výjimkou. Pokusím se představit základní termíny, jak je budu používat v této práci.

Aktivum Označuje věc, informaci, myšlenku, člověka, prostě cokoliv cenného co je potřeba ochránit. Z pohledu analýzy rizik je nejdůležitějším údajem jeho cena. Ocenění může být velmi obtížné hlavně u nehmotných aktiv.

Zranitelné místo a hrozba Dva pohledy na jednu věc. U zranitelného místa sledujeme vše z pohledu aktiva. Snažíme se odhalit a ohodnotit všechny problémové části. U hrozby hodnotíme vše z pohledu útočníka. Heldáme způsoby jak ohrozit aktivum.

Riziko V riziku se musí sejít všechny informace od aktiva, hrozby a zranitelného místa. Rizikem míním hrozbu, která útočí na zranitelné místo, které náleží určitému aktivu.

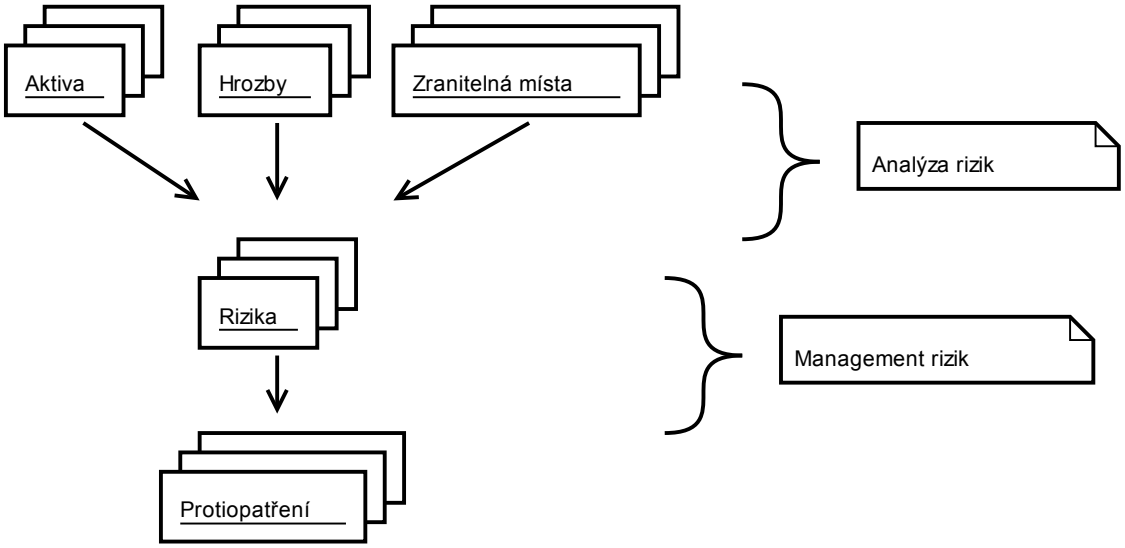
Protiopatření Je prvek, který naopak rizika ve zkoumaném systému zmenšuje. Jedná se o jakýkoliv postup, normu, fyzickou věc, počítačový program, nebo i člověka jehož úkolem je zmenšit riziko, nebo skupinu rizik.

2.2 Stručný scénář analýzy a managementu rizik

Posloupnost analýzy a managementu rizik jsem již naznačil. Shrnu tedy všechny dosavadní informace. Pro lepší názornost vkládám obrázek 2.1, který je převzatý z [7].

1. Identifikace aktiv ve zkoumaném systému a jejich ohodnocení

2. Identifikace zranitelných míst na aktivech. Určení jejich vlivu na aktivum
3. Identifikace hrozeb a pravděpodobnosti útoku na dané zranitelné místo
4. Nalezení všech možných rizik
5. Nalezení vhodných protiopatření na tato rizika
6. Implementace protiopatření do systému



Obrázek 2.1: Analýza a management rizik

2.3 Analýza rizik

Hlavním úkolem je určit všechna rizika v systému. Podívejme se nejprve na způsoby, jak mohou rizika narušit bezpečnost aktiv.

2.3.1 Bezpečnost aktiv

Posuzuje se ze tří hledisek. *Důvěrnost*, *integrita* a *dostupnost*. Útoky na aktiva se snaží narušit některé, nebo všechny tyto hlediska.

Důvěrnost Zajišťuje nám autorizovaný přístup k aktivu.

Integrita Z hlediska celého systému zajišťuje platný stav aktiv a systému.

Dostupnost Každý autorizovaný uživatel má přístup k aktivu, nebo službě za všech okolností.

Úkolem analýzy a managementu rizik je zajistit platnost těchto podmínek. Identifikace a ohodnocení zranitelných míst se zaměřuje na tato hlediska.

Zároveň je však těžké zajistit tyto podmínky vždy. Proto se zjišťuje nebezpečnost (zranitelná místa) a pravděpodobnost ohrožení (hrozby).

2.3.2 Identifikace a ohodnocení aktiv

Základním kritériem je cena aktiva. Pro fyzická aktiva obvykle nebývá problém cenu určit. Bývá jí skutečná aktuální cena aktiva (např. soustruhu). Cena logických aktiv se stanovuje hůře. Tento odhad bývá založen na důležitosti aktiva pro vlastníka. Pro podporu tohoto rozhodnutí se sestavují dotazníky, které se snaží obsáhnout všechny důležité dopady aktiva na zkoumaný systém.

Pro další analýzu se cena aktiva neudává v konkrétní měně, ale na k tomu účelu sestavené stupnici. Tím se zjednoduší vyhodnocování dotazníků i pozdější zpracování. Příklad takovéto stupnice převzatý z [7] je tabulka 2.1.

Hodnota	**Rozsah ceny aktiva v £**
1	0 - 10
2	10 - 50
3	50 - 200
4	200 - 1 000
5	1 000 - 10 000
6	10 000 - 50 000
7	50 000 - 200 000
8	200 000 - 500 000
9	500 000 - 1 000 000
10	1 000 000 a více

Tabulka 2.1: Stupnice cen aktiv

Pro pozdější zpracování se obvykle aktivu přiřadí také kategorie. Jaké kategorie jsou přesně zvoleny záleží na zkoumaném systému. Jako základní rozdělení uvádí [7] toto:

Hardware Kromě významu v IT je možné použít tuto skupinu i pro jiné druhy technického vybavení.

Software Ať už zakoupené, nebo ve firmě vyvinuté aplikace, operační systémy, aj.

Data Data uchovávaná na různých nosičích. Důležité je stanovit příležitosti při kterých data systém získává.

Lidé Zaměstnanci, hosté, externí spolupracovníci.

Dokumentace Jak programová dokumentace, tak různé druhy výkresů, administrativních návodů, smluv, aj.

Zásoby Prázdná média pro uložení dat, aj.

2.3.3 Identifikace a ohodnocení zranitelného místa

Zranitelné místo umožňuje hrozbě vytvořit rizika. Je proto důležité správně zranitelná místa aktiva identifikovat. Zranitelné místo hodnotíme jako rozsah poškození aktiva. Pro toto hodnocení se opět používá stupnice.

Jako způsob jak určit rozsah zranitelného místa se v analýze rizik opět používají dotazníky. Otázky jsou voleny tak, aby odhalily dopad zranitelného místa na aktivum a systém ze tří základních hledisek uvedených v kapitole 2.3.1.

Aby se zranitelná místa lépe zpracovávala, rozdělujeme je do několika kategorií podle toho, jaký dopad mají na systém (viz [2]).

2.3.4 Identifikace a ohodnocení hrozby

Základním údajem je pravděpodobnost uplatnění této hrozby. Jako obvykle se pro podporu stanovení této pravděpodobnosti používá dotazníků. Otázky jsou typu "Co když ..." a možné odpovědi označíme pomocí nějaké stupnice.

Pro identifikaci hrozeb se stanovují kategorie, které pomáhají odhalovat hrozby. Seznam obvyklých kategorií je v [2].

2.3.5 Podpora pomocí IT

Až doposud byla podpora informačních technologií pro analýzu rizik poněkud problematická. Jistě lze vytvořit nástroje, které budou vyhodnocovat dotazníky, shromažďovat informace o aktivu atd. Takové nástroje již existují, např. systém vytvořený pro britskou vládu *CRAMM* ([2]). Stále je však potřeba odborníka na analýzu rizik, který je schopen správné odpovědi získat.

Některé odpovědi do dotazníků lze jistě získat i pomocí automatických nástrojů.

2.4 Management rizik

Úkolem managementu rizik je navrhnout co nejlepší sadu protiopatření, která budou hrozby neutralizovat. Jeho činnost je založena na výsledcích předchozí části. Zpracovává tedy aktiva, hrozby a zranitelná místa. Z nich vytváří rizika a právě rizika dokáží protiopatření potlačovat.

2.4.1 Rizika

Jak jsem již uváděl, existuje vazba mezi zranitelným místem a aktivem či skupinou aktiv. Další vazba je hrozba – zranitelné místo či skupina zranitelných míst. Riziko je tedy kombinací těchto tří údajů.

Hodnota rizika by měla vyjadřovat ztráty, které může způsobit. Jako základní vzorec pro výpočet hodnoty rizika lze použít například tento (převzatý z [7]):

$$\text{riziko} = \text{pravděpodobnost hrozby} \times \text{rozsah zranitelného místa}$$

Problém je v pravděpodobnosti, tedy spíše v tom, že hodnocení je v nějaké stupnici. Pro vyhodnocení rizika se proto sestavují různé tabulky, jako tabulka 2.2 převzatá z [7]. Tak jak je tabulka navržena, je riziko průměrem hrozby a zranitelného místa.

ZM/H	1	2	3	4	5	6	7	8	9	10
1	1	1	2	2	3	3	4	4	5	5
2	1	2	2	3	3	4	4	5	5	6
3	2	2	3	3	4	4	5	5	6	6
4	2	3	3	4	4	5	5	6	6	7
5	3	3	4	4	5	5	6	6	7	7
6	3	4	4	5	5	6	6	7	7	8
7	4	4	5	5	6	6	7	7	8	8
8	4	5	5	6	6	7	7	8	8	9
9	5	5	6	6	7	7	8	8	9	9
10	5	6	6	7	7	8	8	9	9	10

Tabulka 2.2: Určování hodnoty rizika z rozsahu zranitelného místa a pravděpodobnosti hrozby

Do hodnocení zbývá zahrnout ještě cenu aktiva. Tato informace se však většinou uvádí samostatně jako dodatková informace. Používá se k prvnímu třídění rizik před samotným hledáním protiopatření.

Některé společnosti, které se analýzou rizik zabývají, zahrnou cenu přímo do hodnocení rizika.

Zde přichází moment, kdy lze s výhodou využít služeb informačních technologií. Úkolem bude prozkoumání všech vazeb mezi aktivy, zranitelnými místy a hrozbami a nalezení všech rizik. Určit hodnotu rizika podle tabulky nebo vzorce je již snadné.

2.4.2 Hledání protiopatření

Nalezených rizik může být velmi mnoho. Eliminace všech rizik by potom byla velmi nákladná. K dalšímu zpracování se většinou posílají rizika splňující nějaká kritéria. Literatura zavádí tzv. *level rizik* tedy minimální úroveň rizika, kterou je potřeba řešit. Někdy bývá tento *level* ještě doplněn informacemi o minimální ceně.

Dříve (v kapitole 2.1) jsem se snažil definovat protiopatření, jako prostředek ke snížení rizika. Každé protiopatření má svou účinnost. Tato účinnost je vyjádřena tzv. *příspěvkem* k bezpečnosti. Aby mohlo být protiopatření akceptováno jako vhodné na dané riziko, musí být výsledné riziko (po zmenšení o *příspěvek* protiopatření) menší než *level rizik*.

Dalším důležitým údajem u protiopatření je jeho cena. Obvykle není možné za ochranu zaplatit více, než za chráněná aktiva. Záleží na přístupu, jestli nás zajímá celková cena všech protiopatření, nebo dílčí protiopatření a jimi ošetřovaná rizika.

Tím nejdůležitějším údajem je však informace, jaká rizika je protiopatření schopno ovlivnit. Většinou je tato množina definována množinou hrozeb, kterým zabraňuje. Někdy však bývá tato informace doplněna ještě o zranitelné místo.

Pro informační technologie se najde uplatnění hlavně v této části analýzy rizik. Problém, jak podle hodnotících kritérií vybrat nejvhodnější množinu protiopatření, je klasickou úlohou o prohledávání stavového prostoru. Datové struktury vhodné pro analýzu rizik diskutuje [7].

2.4.3 Implementace protiopatření

Seznam navržených protiopatření je výsledkem celého procesu. Pokud budou všechna protiopatření implementována, dojde ke zmenšení rizik na stanovenou úroveň. Takový je aspoň předpoklad.

Před konečnou implementací se provede ještě závěrečná kontrola návrhu.

Pokud jde o automatizované systémy, musí být i protiopatření možné implementovat automaticky. Výhodou je možnost použití jednorázových protiopatření. Například pokud se uživatel obvykle přihlašuje z jedné IP adresy, ale momentálně se pokouší přihlašovat z jiné, aplikuje se právě na toto přihlášení dodatečné protiopatření.

Kapitola 3

Multiagentový systém

Tato kapitola obsahuje souhrn informací o agentových systémech. Snažím se zde přiblížit přístup k problému pomocí agentů. V poslední části kapitoly představuji několik návrhových technik, které používám dalších částech této práce.

Informace uvedené v této kapitole jsem čerpal z knihy [1] a z [8]. Pro další zdroje uvádím i užitečný portál zaměřený na agentové systémy [4].

3.1 Co je to agent?

Jako vždy, když se jedná o IT, je agent jakousi abstrakcí reality. Agent je entita, která dokáže z velké části autonomně existovat a plnit cíle v adekvátním prostředí. Slouží ji k tomu senzory, které zprostředkovávají informace o prostředí a aktuátory, kterými mění stav prostředí. To důležité je agentova přímá závislost na prostředí a automnomita.

Tento přístup pramení z robotů s různou mírou autonomity. Z tohoto pohledu je dán význam prostředí a jeho ovlivňování. Také se zde uplatňuje cíl nebo úkol, kterého se snaží agent–robot dosáhnout.

3.1.1 Chování agenta

Agenta definuje sada senzorů a aktuátorů a k nim se přidává sada chování. Způsob, jakým agent dosahuje svých cílů, je abstrahován jako chování. Opět se to pokusím vysvětlit na příkladu agenta–robota.

Mějme agenta–robota, který se dokáže pohybovat v prostředí. Pokaždé, když senzory zaznamenají překážku, agent–robot změní směr, aby se jí vyhnul. Tento agent–robot v sobě nese dvě chování. To první je velmi jednoduché. Agent–robot jede stále vpřed. To druhé chování jej nutí měnit směr tak, aby se vyhnul překážce.

Je vidět, že velmi důležitým úkolem je zajistit, aby se jednotlivá chování aplikovala ve správný okamžik.

Z toho pohledu lze agenty rozdělit do několika kategorií.

Reaktivní agent Jednotlivé druhy chování se spouštějí jako reakce na vnější podnět, nebo vnitřní stav agenta. Neexistuje zde žádné plánovaní ani žádná representace prostředí. Typickým příkladem je agent–robot zmíněný dříve.

Deliberativní agent Tento agent se podle svých představ o prostředí snaží vytvářet plány na dosažení svých cílů. Významným znakem těchto agentů je vytváření představ o prostředí, vlastnostech a schopnostech jiných agentů. Pro bližší zkoumání doporučuji [1].

Sociální agent U těchto agentů zaujímá podstatnou roli komunikace s ostatními agenty. Důvodem je odstranit omezení vlastních zdrojů. Agent vytváří do jisté míry náhodné a krátké koalice s jinými agenty. Většinou si uchovávají informace o ostatních agentech. Např. spolehlivost, cena za kooperaci, aj.

Hybridní agent Tento způsob se snaží kombinovat předchozí kategorie. Většinou je takovýto agent složen z vrstev do kterých se rozloží jednotlivé schopnosti (z pohledu komunikace, plánování či reaktivity). Otázkou zůstává, má-li mít každá vrstva přímí přístup k senzorům a aktuátorům.

3.1.2 Multiagentový systém

Samostatný agent může v prostředí narazit na jiného agenta, který může plnit jiné cíle. Dokonce cíle obou agentů mohou být protichůdné.

Při takovém "setkání" dochází mezi agenty ke komunikaci. Za komunikaci se považuje ovlivňování prostředí, které je příjemce schopen rozlišit jako zprávu. Tímto je možné považovat všechny účastníky komunikace za agenty podle definice uvedené výše v této kapitole.

Jak jsem již uvedl, cíle agentů v jednom prostředí nemusí být totožné. To však nevylučuje možnost spolupráce mezi agenty. Právě tato spolupráce umožňuje celému systému dosáhnout výsledků, kterých by samotný agent nebyl schopen.

3.2 Softwarové multiagentové systémy

V této oblasti ztratí senzory a aktuátory svůj původní význam a do popředí se dostane samotná komunikace. Samozřejmě senzory a aktuátory jsou důležité. Např. agent, který poskytuje ostatním data, která jsou uložena v konkrétním souboru, je využije. Ale ostatní agenty budou pro svoji činnost využívat právě data získaná komunikací.

Jako malou odbočku a výjimku z toho pravidla uvedu jeden open source projekt [5]. Úkolem je zde naprogramovat bojový tank tak, aby dokázal porazit ostatní tanky v aréně. Zde je kladen velký důraz na senzory a aktuátory.

3.2.1 Prostředí

Jako prostředí se u softwarových agentů používají různé platformy, které většinou zajišťují doručování zpráv mezi agenty. Ty nejobecnější, jako například JADE ([6]), zajišťují jednotné doručování zpráv i mezi agenty na různých počítačích.

Toto se ukazuje jako velmi důležité. Dosáhneme tím možnost distribuovat výpočetní výkon na více počítačů.

Mimo to platformy umožňují, nebo alespoň neznemožňují, přístup ke zdrojům konkrétního počítače. Pro obecné použití získáme dosti obsáhlé a variabilní virtuální prostředí.

Je otázkou pro vývojáře multiagentového systému, jestli pro svou aplikaci použije toto prostředí a adaptuje na něj své agenty, nebo se pokusí adaptovat prostředí tak, aby vyhovovalo jeho záměrům.

3.2.2 Komunikace

To, v čem softwarové agenty excelují, je právě komunikace. Je to dáno zaměřením informačních technologií. Metody sdělování informací jsou velmi propracované. Agentové technologie se snaží jít ještě dále. Ke každé zprávě přidávají informaci o druhu zprávy (informační, nabídka, žádost, ...).

Do této oblasti směřuje řada standardů (jako například FIPA ACL [3]). Jejich cílem je formalizovat komunikační akty.

Komunikace vytváří v systému místo pro agenta a jeho schopnosti.

Distribuovaný systém Klasický přístup, kdy problém rozdělíme na části a každou část svěříme jinému agentu. Důležitým předpokladem je, že se agenty podřídí globálnímu cíli (předpoklad *benevolence*). Řízení takového systému může být centralizované i decentralizované.

Decentralizovaný systém Agent zde dostává větší volnost. Nepředpokládá se, že by agenty byly benevolentní. Počítá se s výskytem agentů s protichůdnými cíli. Díky omezeným zdrojům jsou agenty nuceny kooperovat. (V multiagentových systémech se používají pojmy jako kooperace, koalice .... Do této práce jsem je však nezařadil.) Zásadním problémem je řízení takového systému, kde neexistuje nadřízená autorita.

Většinou se věci mají tak, že jeden agent nabízí nějakou službu (např. poskytnutí informace), kterou jiný agent chce využít. Proto se spolu snaží dohodnout.

Z toho modelu se zrodila potřeba získat "kontakt" na agenta, který je schopen danou službu poskytnout. Jako řešení se nabízí vytvořit zvláštního agenta, který bude buď kontakty aktivně získávat, nebo se u něj budou poskytovatelé služeb registrovat sami. Při žádosti o službu agent jednoduše prohledá svou databázi a poskytne seznam kontaktů. Takový agent se nazývá *matchmaker* (framework JADE [6] poskytuje toto jako službu *yellow pages*). Nebo může agentům přímo poskytnout danou službu. Sám požádá agenty ze své databáze o tuto službu a její výsledky zprostředkuje. Tento agent potom nese název *broker*.

3.2.3 Senzory a aktuátory

Pouze nástroje, které poskytuje daná platforma. Kromě možnosti komunikace se jedná o využívaní zdrojů počítače, na kterém zrovna agent běží.

3.3 Návrh multiagentového systému

Návrh MAS je velmi podobný jako návrh jakéhokoliv jiného systému. Nejprve je nutné podrobně prostudovat zadání a potom za pomoci dostupných prostředků postupně zjemňovat návrh systému. Tato kapitola obsahuje část zjemňovacích technik a dostupných prostředků, které využívají agentový přístup.

Techniky uvedené v této kapitole vycházejí z organizačního návrhového stylu GAIA [8] a jsou uplatnitelné hlavně na distribuované multiagentové systémy malého rozsahu.

3.3.1 Role agentů

Agent jako autonomní celek může sledovat více cílů. Protože ale při návrhu systému je obtížné sledovat simultánně všechny cíle jednoho agenta, přistoupíme k problému jinak. Nejprve definujeme cíle jednotlivých částí systému, o kterých se domníváme, že je může převzít agent. Postup, který vede ke splnění toho cíle, se jmenuje role. Po většinu doby návrhu můžeme pracovat s těmito rolemi.

O každé roli lze rozhodnout jaké prostředky bude potřebovat, se kterými rolemi bude komunikovat a co by mělo být výsledkem této komunikace. Důležitým údajem je očekávaný počet těchto rolí v systému.

Nakonec je vhodné pokusit se přiřadit role agentům. Lze si tak promyslet jak nejlépe spojit různé vlastnosti agentů a rolí.

Výsledkem může být např. diagram rolí a interakcí (viz obrázek 4.1).

Jako poznámku si dovolím předložit následující úvahu. Bylo by možné, aby agentu jeho role nebyla přiřazena již při návrhu. Agent by na sebe roli vzal až za běhu systému. Bylo by tak možné sestrojit universální agenty, kteří by přijímali role podle svých schopností (viz [8]).

3.3.2 Komunikační protokoly

Z existujících rolí a interakcí mezi nimi lze v této fázi návrhu specifikovat komunikační protokoly. Při návrhu vycházíme z požadavků definovaných při návrhu rolí.

Na první úrovni je snaha definovat protokol co nejobecněji. Neřeší se jaké přesné informace musí požadavek obsahovat, jak bude vypadat zpráva, aj. Hlavní pozornost se obrací na časovou posloupnost žádostí a odpovědí s ohledem na řešení daného problému.

Schéma takového protokolu je na obrázku 4.3

3.3.3 Chování

Ze znalostí protokolů a rolí pokračuje návrh ve specifikaci chování. Ještě jednou se projde předchozí návrh a tentokrát si již můžeme všímat detailů typu: “Kde vezme tento agent informaci o . . . ” a doplníme návrh o tyto náležitosti, čímž získáme pro každou roli a agenta dosti přesný seznam požadavků.

Podle požadavků se navrhnou chování, která odpovídají základům jednotlivých rolí a zvolí se způsob jejich spouštění.

Nakonec se zvolí způsoby vnitřní representace stavů agenta, prostředí, aj. a definují se konkrétní zprávy.

Kapitola 4

Návrh systému pro analýzu rizik

V této kapitole představuji návrh systému pro analýzu rizik. Snažím se o ucelený a postupný popis analýzy a návrhu systému a vysvětlení návrhových rozhodnutí.

Diskutuji zde požadavky na systém pro analýzu rizik, identifikace rolí a interakcí, návrh komunikačních protokolů a neformální popis chování agentů. Úkoly zde řešené jsou nutné v každém systému pro analýzu rizik. Konkrétní návrh implementovaného systému je v kapitole 5.

Za některými výrazy uvádím jejich přepis používaný na obrázcích a ve zdrojových textech. Tento text je vysázen `strojopisně`.

4.1 Požadavky na systém pro analýzu rizik

Pokusím se tedy definovat očekávání od tohoto systému. V kapitole 2 jsem jako jeden z hlavních výstupů analýzy rizik zmínil seznam protiopatření. Protiopatření, která je třeba implementovat, aby byl zkoumaný systém ochráněn. Volba hlavního produktu systému pro analýzu rizik proto padla na tento seznam.

Dalším logickým krokem je definice vstupů systému. Analýza rizik uvádí jako vstup pro management rizik jejich seznam a známá protiopatření. Dále zavádí způsob určení rizik pomocí aktiv, zranitelných míst a hrozeb. Což vymezuje další vstupy do systému pro analýzu rizik.

Nakonec, a toto jsou požadavky zadání, by systém měl být implementován jako distribuovaný, založený na autonomních agentech v jazyce Java.

4.2 První krok v návrhu

Už při čtení požadavků jsem rozdělil systém pro analýzu na dvě části. Na část zjišťující a shromažďující rizika a na část hledající vhodná protiopatření na tato rizika.

4.3 Objekty v modelu analýzy rizik

Analýza rizik definuje množství objektů. Úkolem je určit, které objekty modelovat jako agenty, které jako představy agentů a které úplně vypustit. Zároveň s tím vzniká i model systému.

Rizika Figurují ve všech částech procesu analýzy rizik, bylo by výhodné je modelovat jako samostatné agenty. Však ostatní agenty si i tak budou muset o riziku vytvářet své představy. Zároveň se rizika vytvářejí v rámci vlastní analýzy rizik. Také činnost takového agenta, by se omezila pouze na předávání informací o sobě.

Na druhou stranu bude nutné zabudovat do komunikačních protokolů celkovou informaci o riziku, i když ji konkrétní agent třeba nebude potřebovat. I přesto zatím bude výhodnější riziko jako agenta nemodelovat.

Aktivum Je základním stavebním kamenem první části analýzy rizik. Při modelování jako agent bude moci produkovat rizika. K tomu potřebuje informace o hrozbách a zranitelných místech.

Při modelování aktiva jako představu agenta by riziko generoval jiný agent (třeba zranitelné místo) za pomoci informací o aktivu. V tomto okamžiku by to byla cena aktiva.

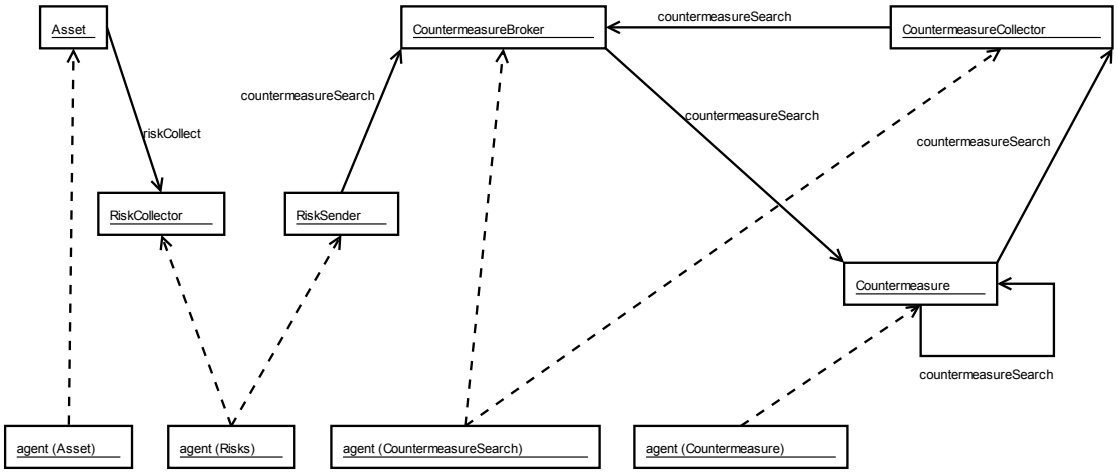
Obě konstrukce jsou srovnatelné. Rozhodl jsem se pro modelování aktiva jako samostatného agenta.

Hrozba a zranitelné místo Existují jako představy agentů aktiv. Není proto důvod je modelovat jako samostatného agenta.

Protiopatření Patří do druhé části analýzy rizik. Modelování proto bude záviset na způsobu prohledávání stavového prostoru. Z pohledu agentových systémů nám jednotlivé agenty budou nabízet jako službu eliminaci rizika. Výsledkem je právě protiopatření. Z tohoto hlediska je protiopatření modelováno jako představa agenta.

4.4 Model rolí a interakcí

Výsledek úvah shromážděných v této kapitole ukazuje obrázek 4.1.



Obrázek 4.1: Model rolí a interakcí

V tomto stádiu jsem schopen definovat role jednotlivých agentů a interakce mezi nimi. Začnu tím jediným pojmem, který jsem z analýzy rizik použil pro agenta, *aktivem* (role `Asset`). Tato role slouží jako zdroj rizik. Pro další zpracování je třeba rizika ze všech aktiv

shromáždit. Roli pojmenuji *sběrač rizik* (`RiskCollecor`). Protokol, který popisuje tuto činnost jsem nazval ***`riskCollect`***.

Tímto jsme splnili první část úkolu systému pro analýzu rizik. Máme shromážděná všechna rizika na jednom místě. Následuje hledání nejvhodnějších protiopatření.

Celou akci zahajuje agent s rolí *odesilatel rizik* (`RiskSender`). Rizika určená ke zpracování zašle agentu s rolí *broker protiopatření* (`CountermeasureBroker`), který je zodpovědný za nalezení nejlepšího řešení. Ke své práci využívá agenty s rolemi *protiopatření* (`Countermeasure`). Zde je důležité upozornit, že tyto agenty uchovávají informaci, jak vyřešit nějakou sadu rizik. Nemusí to být jenom jedno protiopatření definované analýzou rizik. Protokol ***`countermeasureSearch`*** definuje cílovou roli celého snažení – *sběrač protiopatření* (`CountermeasureCollector`). Tato role vybere z návrhů ten nejlepší a postará se o výstup ze systému.

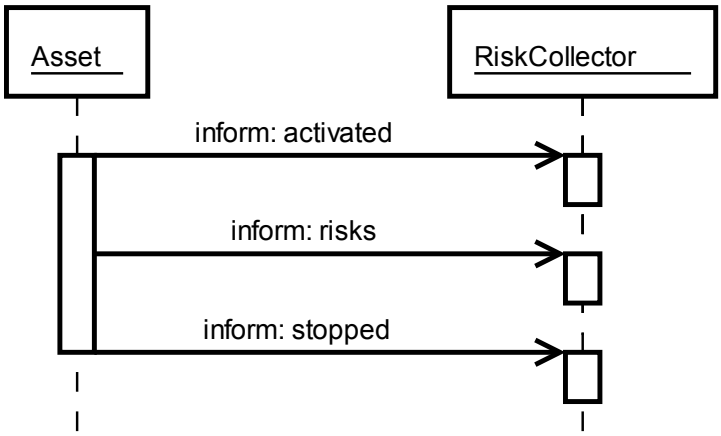
4.5 Komunikační protokoly

U každého komunikačního protokolu uvedu cíl jeho snažení, schéma a role agentů.

4.5.1 Protokol `riskCollect`

Cílem protokolu je shromáždit rizika. Rizika produkuje agent s rolí *aktivum*. Příjemce je *sběrač rizik*. Na obrázku 4.2 je schéma protokolu.

Aby *sběrač rizik* měl informaci o tom kolik rizik již sesbíral, může se postupně dotazovat *aktiv*, nebo mu budou *aktiva* zasílat tuto informaci sami. Druhá možnost podporuje snažší přidávání dalších *aktiv* za běhu systému. Stále je nutné mít na paměti, že navrhujeme distribuovaný systém a můžeme předpokládat benevolenci agentů (viz kapitola 3.2.2).

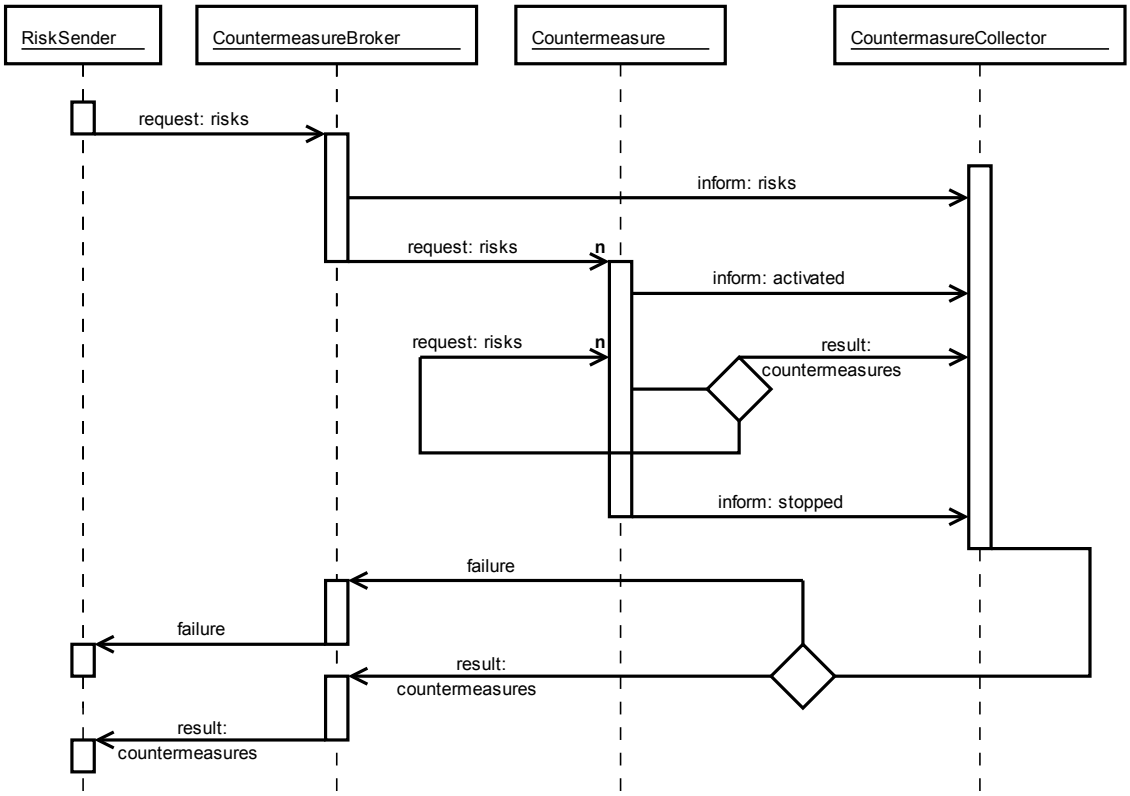


Obrázek 4.2: Protokol `riskCollect`

Aktivum nejprve zašle informaci o tom, že má nějaká nezpracovaná rizika (že je aktivní). Potom odesílá rizika. Není omezeno, kolik zpráv s riziky odešle. Nakonec oznámí *sběrači rizik* konec odesílání rizik (informaci o zastavení).

Popis syntaxe zprávy o aktivitě / ne-aktivitě (`stateInform`) popisuje kapitola 5.5.1. Syntaxe informace o riziku (`riskInform`) zase kapitola 5.5.2.

4.5.2 Protokol `countermasureSearch`



Obrázek 4.3: Protokol `coutermeasureSearch`

Tak jako byl předchozí protokol orientován na rizika, je tento orientován na protiopatření. Prvotní požadavek (viz obrázek 4.3) na vyhledání nejlepší množiny protiopatření, vychází od role *odesílatel rizik*. (Jako malou odbočku si dovolím podotknout, že tuto roli pravděpodobně bude zastávat stejný agent jako roli *sběrače rizik* protokolu `riskCollect`, viz obrázek 4.1.) Žádost bude obsahovat seznam rizik (zpráva je popsána v 5.5.2) a je zaslána roli *broker protiopatření*.

Úkolem *brokeru protiopatření* je zprostředkovat vlastní vyhledání nejvhodnějších protiopatření. Z jemu známých agentů s rolemi *protiopatření* vybere část a tu požádá o řešení. Ve zprávě dále určí další *protiopatření*, které lze kontaktovat při hledání řešení.

Role *protiopatření* vyřeší takovou část problému, kterou zvládne. Pokud se mu podaří vyřešit celý problém, odešle své řešení agentu s rolí *sběrač protiopatření*. V opačném případě předá odpovědnost spolu s částečným řešením agentům specifikovaným v žádosti (sebe již ovšem neuvede). Ti se pokusí v rámci svých možností problém buďto vyřešit, nebo opět částečné řešení předají dál. (V testovací implementací je všechna tato komunikace obstarávána zprávou popsanou v 5.5.3.)

Posledním účastníkem tohoto protokolu je role *sběrač protiopatření*. Jak bylo uvedeno přijímá od *protiopatření* návrhy na řešení problému. Z nich vybere ten nejlepší a výsledek předá zpět *brokeru protiopatření*, který výsledek předá původnímu žadateli. Aby *sběrač protiopatření* poznal, že má všechny návrhy, používám zprávy o aktivitě *protiopatření*.

Tyto zprávy fungují stejným způsobem jako u protokolu `riskCollect` (4.5.1). Dále přijímá od *brokeru protiopatření* oznámení o spuštění řešení problému. Může tak reagovat i na požadavky bez řešení.

4.6 Chování agentů

Nastává vhodný okamžik specifikovat chování a úkoly, které musí agent zvládnout, aby mohl přijmout danou roli. Většina úkolů už byla specifikována při popisu protokolů a zbytek při vytváření modelu rolí a interakcí. Tato sekce je tedy shrnutím těchto požadavků.

Aktivum Shromáždí zranitelná místa a hrozby. Z nich vytvoří rizika, která odešle *sběrači rizik*. Ten je také informován o začátku a konci zpracování rizik.

Sběrač rizik Přijímá rizika a zprávy o aktivitě od *aktiv*. Rizika shromažďuje, může je jakkoliv zpracovávat (např. třídit stejná rizika apod.) a v okamžiku kdy dostane od všech *aktiv* informaci o ukončení hledání rizik, provede s riziky nějakou akci. Nejobvyklejší je předat je *odesílateli rizik*. Pokud by mezitím bylo přidáno nějaké další *aktivum*, jeho rizika budou přidány ke stávajícímu seznamu a celá akce se provede znovu s rozšířenou množinou rizik.

Odesilatel rizik Požádá *broker protiopatření* o nalezení nejlepší množiny protiopatření. Převezme výsledky a postará se o další zpracování, například zobrazí výsledky uživateli.

Broker protiopatření Vyřizuje žádosti od *odesilatele protiopatření*. O každé žádosti informuje *sběrače protiopatření*. Potom vybere skupinu nejvhodnějších *protiopatření* a tu požádá o řešení. Zpráva, kterou jim pošle, obsahuje rovněž další *protiopatření*, které může požádat o pomoc. Posledním úkolem je převzít od *sběrače protiopatření* výsledky a předat je žadateli.

Protiopatření Má informace o protiopatřeních (z pohledu analýzy rizik), tzn. jaká rizika ošetřuje. Přijímá zprávy, které obsahují částečné řešení, neošetřená rizika a seznam agentů s rolí *protiopatření*. Podle svých znalostí změní částečné řešení a množinu nešetřených rizik. Jestli je problém vyřešen (žádná neošetřená rizika), řešení pošle *sběrači protiopatření*. Jinak předá odpovědnost agentům určených ve zprávě. O své aktivitě informuje *sběrače protiopatření*.

Sběrač protiopatření Porovnává jednotlivá řešení a vybírá to nejlepší. Zároveň kontroluje aktivitu agentů *protiopatření*. Jakmile má jasno informuje o výsledku *broker protiopatření*.

Kapitola 5

Testovací implementace systému pro analýzu rizik

V kapitole 4 jsem diskutoval obecné prvky a chování systému pro analýzu rizik. Zde popisuji jednu implementaci tohoto systému. Rozhodnutí týkající se dat v systému: vstup, výstup a formát dat. Přiřazení rolí a popis zpráv.

5.1 Implementační jazyk

V zadání je specifikován jazyk Java. Do značné míry obhajuje volbu právě tohoto jazyka přenositelnost. Porovnání různých jazyků z hlediska agentových systémů je v [1]. Tamtéž je jako vývojové prostředí pro jazyk Java doporučen i framework JADE (viz WWW projektu [6]). Další nástroje lze nalézt na portále [4].

5.2 Rozdělení rolí agentům

Role *aktivum* a *protiopatření* je nejlépe implementovat jako samostatné agenty. Využijeme tak možnosti distribuovaného zpracování rizik i hledání protiopatření.

Role *sběrač rizik* a *odesilatel rizik* spojíme do jednoho agenta, jak bylo dříve naznačeno.

Stejně tak role *broker protiopatření* a *sběrač protiopatření*. Jejich separace by umožňovala zvýšit propustnost systému, kdy by mohl broker přijímat další požadavky a o třídění výsledků by se staral jiný agent. Však navrhovaný systém je testovací a proto nepředpokládám tak velikou zátěž.

5.3 Vstup dat

Je třeba zajistit aby agenty *aktiva* dostaly informace o ceně aktiva, zranitelných místech a hrozbách. Také je třeba agentům *protiopatřením* dodat informace o známých protiopatřeních.

Hned na začátku odmítnu nejzřejmější způsob a to předat agentům potřebné informace pouze přes parametry. Agentům je totiž třeba dodat tak velké množství informací, že se tento způsob stává dost nepřehledným. Lépe by bylo umístit informace do souboru, ze kterého bude agent čerpat. S výhodou využiji předpřipravených analyzátorů formátu XML.

Soubor s těmito daty pojmenovávám `data.xml` a je umístěn v aktuálním adresáři.

5.3.1 Aktiva, hrozby a zranitelná místa

Protože zranitelné místo může být ve více aktivech, bylo by vhodné navrhnout strukturu souboru tak, aby mohl být sdílen pro více *aktiv*, potažmo i *protiopatření*. Je proto nutné, aby bylo možno informace o aktivu v souboru identifikovat. Dobře poslouží číslo, které bude předáno jako parametr agenta. Dalším důležitým údajem je cena aktiva. Příklad zápisu:

```
<asset id="2" price="7" description="Manazerska data" />
```

Parametr `description` je důležitý pro tvůrce souboru. Agent jej sice přečte, dále s ním však nijak nepracuje.

Zranitelné místo popíšu podobě jako aktivum. Nelze zapomenout na vazbu na aktivum.

```
<vulnerability id="1" scope="5" description="Fyzicky pristup">
  <asset id="1" />
</vulnerability>
```

Stejný recept aplikuji potřetí na hrozby. Označím také vazbu na zranitelná místa.

```
<threat id="3" likehood="3" description="Prepeti v~siti">
  <vulnerability id="3"/>
</threat>
```

Je důležité podotknout, že parametry `id` u jednotlivých skupin xml elementů je nutné udržovat unikátní. Potom je lze použít pro porovnávání.

5.3.2 Protiopatření

Budu dále pokračovat stejným způsobem. U protiopatření, kromě identifikátoru, je zajímáva jeho cena a účinnost (přesně viz kapitola 2). A hlavně rizika na která působí. (Dávám přednost výčtu rizik, která protiopatření potlačuje před seznamem hrozeb. Více viz 2.4.1).

U rizik se ale zastavím. Bude dobré si uvědomit, co riziko identifikuje. Mohl bych použít opět parametr `id`, ale stejně bych potom musel riziko dále specifikovat. Ze všech hodnot, které riziko identifikují, vycházejí jako základní hrozba a zranitelné místo. Popis protiopatření bude tedy vypadat takto:

```
<countermeasure id="3" price="2" contribution="8"
                        description="Prepetovy chranic">
  <risk vulnerability="3" threat="3"/>
</countermeasure>
```

Tím jsem vyřešil vstup dat do systému. Zbývá specifikovat zprávy v systému a výstup ze systému. Ale nejdřív je nutné zjistit co všechno si pro svůj provoz musí agenti sdělit.

5.4 Informace nutné pro funkci agenta

Cílem je prostudovat všechny agenty a zjistit, které informace potřebuje ke své činnosti a navrhnout způsob jak mu tuto informaci poskytnout.

Aktivum Kromě informací zmíněných v kapitole 5.3.1 je pro správnou funkci nutné znát adresu agenta s rolí *sběrač rizik*. Zde jsem s úspěchem použil službu frameworku JADE nazvanou *yellow pages* (více v kapitole 3.2.2).

Sběrač a odesílatel rizik Seznam přijatých rizik je znám v rámci agenta. Potřebuje znát adresu agenta s rolí *broker protiopatření*, tato informace je opět dostupná přes službu *yellow pages*.

Broker a sběrač protiopatření Jako broker potřebuje znát adresy a zpracovávaná rizika agentů *protiopatření*, aby mohl zvolit nejlepší skupinu agentů, které požádá o řešení. Šlo by opět využít *yellow pages*. Každý agent *protiopatření* by zaregistroval pro každé riziko, jeho řešení jako službu. Dohledat agenty pro konkrétní množinu rizik, by bylo snadné.

Řešením, které jsem použil já, je informovat *brokera protiopatření* o řešených rizicích pomocí zvláštní zprávy. Odpovědnost za toto připadne agentům s rolemi *protiopatření*. Všechny další informace jsou známé v rámci agenta.

Protiopatření Vezmu-li v potaz předchozí odstavec, potřebuje tento agent znát adresy dvou rolí. *Broker protiopatřením* a *sběrač protiopatření*. Adresa první role je zveřejněna na *yellow pages* a obě role přijímá jeden agent. Nic nebrání tomu zveřejnit stejným způsobem i *sběrače protiopatření*.

5.5 Syntaxe zpráv

V kapitole 4.5 jsem popisoval komunikační protokoly v systému. Nyní specifikuji konkrétní zprávy. Každá zpráva se skládá z identifikátoru zprávy, za kterým v závorce následuje parametr. Pokud je potřeba přenést více údajů použije se seznam tak jak je znám z programovacích jazyků Lisp, či Prolog.

5.5.1 Zpráva `stateInform`

Přenáší informaci o aktivitě agenta. Je zasílána rolemi *aktivum* a *protiopatření*. Více popisuje kapitola 4.5.

Přenášená informace je dvoustavová:

`activate` agent vykonává nějakou činnost

`stop` agent svou práci dokončil

5.5.2 Zpráva `riskInform`

Přenášena je informace o rizicích. Je využívána rolemi *aktivum* a *odesílatel protiopatření*. Ve druhém případě se jedná o žádost a označeni *inform* je trochu zavádějící.

Riziko je identifikováno pomocí `id` zranitelného místa (`v`) a `id` hrozby (`t`). Dále je potřeba přenášet informaci o hodnocení rizika (`r`) a ceně aktiva (`p`). Do zprávy dáme možnost přenést více rizik. Parametr potom bude mít tento formát:

[[v , t , r , p] , . . .]

5.5.3 Zpráva `countermeasureSearch`

Tato zpráva je využívána ve stejnojmenném protokolu (4.5.2). Jeho úkolem je přenést informace o nevyřešených rizicích, částečném řešení a agentech s rolí *protiopatření*. Je zasílána rolemi *protiopatření* a *broker protiopatření*.

Seznam rizik jsem popsal u zprávy `riskInform` (5.5.2). Použiji zde stejný způsob. Řešení obsahuje informaci o protiopatřeních a jeho dosavadní ceně. Dále seznam agentů, kterým lze předat odpovědnost za pokračovaní řešení(což je také pouze seznam jmen). Zpráva ještě musí obsahovat level rizika (viz. 2).

Zprávu však mohu zjednodušit, pokud nechám agenty s rolí *protiopatření* a konkrétní protiopatření mapovat v poměru 1 : 1, mohu ze zprávy vypustit seznam navrhovaných protiopatření. Ten dostanu jako doplněk k momentální množině agentů *protiopatření* a původnímu stavu, který vyslala role *broker protiopatření*.

Parametr zprávy by měl potom tento tvar:

```
[[[v,t,r,p],...],[contermeasureAID,...],price,level]
```

5.5.4 Zpráva `countermeasureInform`

Slouží k informovaní *brokera protiopatření* o protiopatření a rizicích, které řeší a k přiřazení této informace k určité adrese agenta. Odesílá agent s rolí *protiopatření*.

Opět obsahuje seznam rizik. Dále informaci o `id` protiopatření.

```
[countermeasureID,[[v,t,r,p],...]]
```

5.6 Poslední poznámky k implementaci

Nebyla zde nikde popsána zpráva, která výsledky pošle *odesílateli rizik*, protože se žádná taková zpráva neposílá. O vypsání výsledků se stará agent *broker protiopatření*. Je to proto, že úkolem systému je předat výsledky uživateli a tento poslední krok by byl zbytečný.

Výsledek je seznam protiopatření a celková cena. Aby výsledek byl pro uživatele co nejpřijatelnější vypisuje systém jména agentu *protiopatření*. Je tedy na něm jak si je pojmenuje.

Příklad dávkového spuštění systému:

```
java -classpath ../lib/jade.jar;../bin/ jade.Boot CountermeasureSearch:ra.ag
ents.CountermeasureSearchAgent Zamek:ra.agents.CountermeasureAgent(1) Sifrov
ani-Disku:ra.agents.CountermeasureAgent(2) Prepetovy-Chranic:ra.agents.Count
ermeasureAgent(3) Otisk-Prstu:ra.agents.CountermeasureAgent(4) VPN:ra.agents
.CountermeasureAgent(5) MagementNotebook:ra.agents.AssetAgent(1) ManagementD
ata:ra.agents.AssetAgent(2) Risks:ra.agents.RisksAgent
```

Příklad výstupu systému:

```
Problem: 1179140160109
cena: 12
Reseni:
Sifrovani-Disku
Otisk-Prstu
Prepetovy-Chranic
VPN
```

Kapitola 6

Experimenty k ověření funkčnosti systému

Tato kapitola popisuje několik experimentů, které jsem na systému provedl. Než se dostanu k samotným experimentům popíšu agenty v systému.

Aktivum (`ra.agents.AssetAgent`) Představuje aktivum v systému (role *aktivum*). Pro svůj běh potřebuje v aktuálním adresáři soubor `data.xml` s potřebnými informacemi (viz kapitola 5.3.1). Jako parametr dostane `id` aktiva v souboru. Počet těchto agentů je roven počtu zkoumaných aktiv.

Rizika (`ra.agents.RisksAgent`) Sdružuje v sobě role *sběrače rizik* a *odesílatele rizik*. V systému je jenom jeden.

Protiopatření (`ra.agents.CountermeasureAgent`) Tento agent je implementací role *protiopatření*. V systému existuje jeden agent pro každé protiopatření (název agenta by měl dané protiopatření co nejlépe vystihovat). Potřebná data jsou v souboru `data.xml` umístěném v aktuálním adresáři (může se jednat o stejný soubor jako pro aktiva). Parametrem je `id` protiopatření.

Vyhledávač protiopatření (`ra.agents.CountermeasureSearchAgent`) Jedná se o spojení rolí *brokera* a *sběrače protiopatření*. V systému se předpokládá pouze jeden tento agent.

6.1 Experiment 1: Jednorázové spuštění systému

6.1.1 Cíl

Otestovat schopnost systému jednorázově nalézt sadu protiopatření. Simuluji tím situaci, kdy např. společnost zabývající se analýzou rizik shromáždí všechny podklady (aktiva, hrozby a zranitelná místa) a požaduje od systému sadu protiopatření. Systém se spustí jednorázově pro jeden úkol a hned se zase vypne.

6.1.2 Postup experimentu a předpokládané výsledky

Agenty spustím v tomto pořadí: agenta *Vyhledávač protiopatření*, všechny agenty *Protiopatření*, agenta *Rizika* a nakonec všechny agenty *Aktiva*. Výsledkem by měl být seznam

jmen agentů *Protiopatření* a cena za celkové řešení. Pokud bude více srovnatelných řešení, měly by být uvedený všechny.

6.1.3 Průběh experimentu a výsledky

Experiment jsem několikrát opakoval s různými daty. Výsledné řešení odpovídalo předpokladům.

Někdy se ovšem stalo, že se některé *aktivum* poněkud zpozdilo. (Všechny agenty běžely na jednom počítači.) Agent *Rizika* dostal informaci o tom, že všechny agenty již ukončily svou činnost a odeslal rizika k řešení. Teprve potom se zpozděné *aktivum* nahlásilo k řešení (toto chování je předmětem Experimentu 2). Výsledky však bylo možné odlišit podle čísla problému.

Další nedokonalost spočívá ve zobrazování stejných řešení. Systém je považuje za odlišné, protože k nim došel jinou cestou. Znamená to úpravu algoritmů ve *sběrači protiopatření*.

6.2 Experiment 2: Přidání dalšího Aktiva za běhu systému

6.2.1 Cíl

Do běžícího systému přidat další aktivum tak, aby bylo zahrnuto do celkového výsledku. Simuluje se situace, kdy je další aktivum přidáno později za běhu (např. v systémech, které analyzují průběžně).

6.2.2 Postup experimentu a předpokládané výsledky

Agenty spustím v tomto pořadí: agenta *Vyhledávač protiopatření*, všechny agenty *Protiopatření*, agenta *Rizika* a nakonec agenty *Aktiva*. Poslední protiopatření je přidáno až nakonec.

Výsledkem by měli být dvě sady protiopatření. Jedna, bez zahrnutí dodatečného aktiva a druhá s ním.

6.2.3 Průběh experimentu a výsledky

Znamenalo to pokusit se opakovat zpoždění jednoho nebo více aktiv při vstupu do systému z předchozího experimentu. Po spuštění sady aktiv systém nalezl řešení. Po dodání dalšího aktiva agent *Rizika* zaslal novou žádost, která obsahovala navíc i rizika z přidaného aktiva. Nově nalezená řešení byla správně odlišena.

6.3 Experiment 3: Řešení více samostatných úloh

6.3.1 Cíl

Prozkoumat zda na jedné sadě protiopatření lze řešit i více samostatných úloh. Je to simulace situace, kdy *Vyhledávač protiopatření* se svými *Protiopatřeními* běží permanentně a do systému se připojují různé agenty *Rizika* s *aktivy*.

6.3.2 Postup experimentu a předpokládané výsledky

Spustíme agenta *Vyhledávač protiopatření* a všechny agenty *Protiopatření*. Potom spustíme agenta *Rizika* a agenty *Aktiva* z první úlohy. Agenta *Rizika* ze systému odstraníme a znovu spustíme. Spustíme novou sadu agentů *Aktiv* pro získání dalšího výsledku.

Základním nedostatkem je, že implementace systému neumožňuje agenty *Aktiva* spojit s konkrétním agentem *Rizika*. *Aktiva* se spojí s prvním nalezeným agentem s rolí *sběrač rizik*. Při uvedeném postupu budou výstupem dvě sady řešení každá k jedné úloze.

6.3.3 Průběh experimentu a výsledky

Při tomto experimentu se bylo občas velmi těžké přiřadit výsledek správné sadě aktiv. Nejvíce se na tom podepsalo rozhodnutí neodesílat výsledky zpátky roli *odesílatel rizik*. Při zpoždění některého aktiva (viz výsledky Experimentu 1) byly *brokeru protiopatření* zaslány až 4 žádosti o řešení sady rizik. Systém potom vygeneroval 4 různé návrhy řešení z nichž 2 byly neplatné, protože v systému nebyla zahrnuta všechna aktiva.

6.4 Experiment 4: Přidání dalšího Protiopatření za běhu systému

6.4.1 Cíl

Do běžícího systému přidat další protiopatření tak, aby další úlohy toto protiopatření akceptovaly. Jedná se o rozšíření předchozího experimentu, kdy mezi dvěma úlohami ještě přibude do systému další protiopatření.

6.4.2 Postup experimentu a předpokládané výsledky

Spustíme agenta *Vyhledávač protiopatření* a agenty *Protiopatření*. Potom spustíme agenta *Rizika* a agenty *Aktiva* z první úlohy. Přidáme dalšího agenta *Protiopatření*. Agenta *Rizika* ze systému odstraníme a znovu spustíme. Spustíme novou sadu agentů *Aktiv* pro získání dalšího výsledku.

Předpokládám opět dvě sady protiopatření, jedna však bude akceptovat i nové protiopatření.

6.4.3 Průběh experimentu a výsledky

Tento se mi nepodařilo provést dostatečně rychle. Proto byl výsledek první sady aktiv znám ještě před tím, než bylo do systému vloženo nové protiopatření. V tomto případě fungoval systém požadovaným způsobem, tedy při spuštění další sady aktiv bylo do výsledku zahrnuto i nové protiopatření.

6.5 Shrnutí experimentů

Výsledky experimentů dokazují, že systém je schopen nalézt správnou sadu protiopatření. Obtíže způsobují hlavně problémy se současným spouštěním agentů *Aktiv*. Znamená to, že protokol `riskCollect` nepočítá s přechodným stavem po zavedení skupiny aktiv.

Při spuštění více úkolů, bylo také obtížné přiřadit řešení správné skupině aktiv. Tento problém by spolehlivě vyřešilo odesílat sadu protiopatření zpátky roli *odesílatel rizik*, tak jak to navrhuje protokol *countermasureSearch*.

Kapitola 7

Závěr

Tato práce se snaží představit model distribuovaného systému pro podporu analýzy rizik. Procesy analýzy a managementu rizik tak jak byly popsány však stále vyžadují odborný dohled. Jedná se hlavně o ohodnocení aktiv, zranitelných míst a hrozeb.

Také pohled na ohodnocování rizika je velmi subjektivní. Firmy, které se analýzou rizik zabývají, volí různá kritéria. Např. do celkového ohodnocení rizika započítávají i cenu aktiva.

Proto jsem se snažil systém navrhnout tak, aby využíval plně možností distribuovaného zpracování, ale zároveň návrh stále nechával volnost v pohledu na ohodnocení rizika. Díky tomu je návrh rozdělen na dvě části. Část řešící obecnou strukturu systému, důležité protokoly a neformální popis rolí. Druhá část je zaměřená na detaily, které se mohou lišit v různých implementacích (jako definice zpráv, řešení konkrétních vstupů a výstupů).

Role aktivum, tak jak je definována v obecné fázi návrhu, předpokládá informace o hrozbách a zranitelných místech. Nebude problém při konečné definici napojit tuto roli na již existující systém, který žádané informace poskytne.

Zajímavou úpravou by bylo použití systému v tzv. risk–based autentizaci. Jedná se o způsob sledování zvyků uživatele, kde, pokud dojde ke změně, si systém vyžádá dodatečnou autentizaci. Zde by bylo nutno upravit role aktiva (sledování zvyků) a odesílatel protiopatření (musí ze seznamu protiopatření vygenerovat dodatečnou autentizaci).

Celý systém má však slabinu. Protože je navrhován jako distribuovaný, předpokládá pouze *benevolentní* agenty. Proto je velmi snadné tento systém kompromitovat. Další vývoj by měl proto směřovat k nalezení bezpečnějších protokolů a dalších bezpečnostních složek, které by se vypořádaly s tímto rizikem.

Další nevýhodou je přílišná jednoduchost protokolu na shromažďování rizik. Ten nepočítá se současným spouštění skupiny aktiv. Systém potom vyprodukuje sadu chybných řešení. Správné řešení je až to po odeznění přechodného děje.

Systém tedy najde využití hlavně při dlouhodobém sledování a analyzování nějakého systému.

Literatura

[1] Aleš Kubík. *Inteligentní agenty.* Computer Press, a.s., 2004. ISBN 80-251-0323-4.

[2] Security Service on behalf of the UK Government. *The CRAMM Risk Analysis and Management Metod.* London, 1996.

[3] WWW stránky. The foundation for intelligent physical agents. `http://www.fipa.org/`.

[4] WWW stránky. Portál zaměřený na agenty. `http://agents.umbc.edu/`.

[5] WWW stránky. Projekt robocode. `http://robocode.sourceforge.net/`.

[6] WWW stránky. Vývojový framework agentových systému v javě – jade. `http://jade.tilab.com/`.

[7] František Zbořil. *Risk analysis and management methods [Diploma Project].* Brno University of Technology, 2000.

[8] František Zbořil. *Návrh a modelování agentů [Podklady k přednášce].* Brno University of Technology, 2006.

Seznam příloh

Přílohy na CD:

- testovací soubor `data.xml` a výsledky testů
- zdrojové texty balíku `ra` s potřebnými třídami pro analýzu rizik
- programová dokumentace balíku `ra`
- neformální specifikace zpráv a rolí