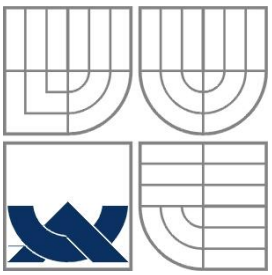
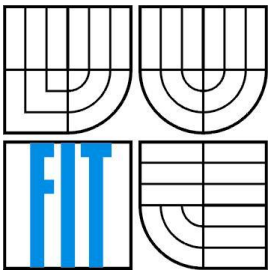


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

CRASH ANALYSIS PORTAL

CRASH ANALYSIS PORTAL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. MILAN JANEČEK
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. JAROSLAV ZENDULKA, CSC.
SUPERVISOR

BRNO 2008

Abstrakt

Práce představuje problematiku analýzy chyb softwaru v provozním prostředí. Zavádí pojem provozní chyby a systému pro analýzu provozních chyb, jehož jedné části, centrále, je věnována další část práce, která obsahuje analýzu požadavků a návrh centrály systému pro analýzu provozních chyb. Zvolená část návrhu centrály je dále implementována a na realizovaném výstupu je provedeno zhodnocení správnosti návrhu a praktické využitelnosti centrály.

Klíčová slova

provozní chyba, systém pro analýzu provozních chyb, centrála, analýza požadavků, návrh

Abstract

This work presents problems of software products' field failure analysis and introduces terms a field failure and a system for field failures analysis. Rest of this work contains requirements analysis and design of one part of the system for field failures analysis – a central. A part of the central is then implemented and design correctness and practical usability of the central is evaluated.

Keywords

field failure, system for field failures analysis, central, requirements analysis, design

Citace

Janeček, M. *Crash Analysis Portal* [diplomová práce]. FIT VUT v Brně, Brno, 2008

Crash Analysis Portal

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Jaroslava Zendulky, Csc. Další informace mi poskytli Ing. Karel Obluk, PhD., Ing. Vladimír Čech, PhD. a Mgr. Jiří Švec. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Milan Janeček

15. května 2008

Poděkování

Tato práce by nemohla vzniknout bez prvotní myšlenky na realizaci systému pro analýzu provozních chyb, za kterou děkuji technickému řediteli společnosti AVG Technologies, panu Ing. Karlu Oblukovi, PhD.

Dále bych chtěl poděkovat Ing. Vladimíru Čechovi, PhD., který mi byl po celou dobu řešení práce dobrým technickým vedoucím a především neúnavným optimistou dodávajícím mi sílu i ve zdánlivě bezvýchodných situacích. Mgr. Jiřímu Švecovi děkuji především za dohled nad správnou formulací požadavků na systém ze strany oddělení technické podpory.

Za kvalitní a pečlivé pedagogické vedení a zejména za pomoc při tvorbě teoretické části práce musím také poděkovat panu Doc. Ing. Jaroslavu Zendulkovi, Csc.

Obsah

1 Úvod

Všechny modely životního cyklu softwaru dnes obsahují etapu pojmenovanou jako provoz a údržba. Jedná se zpravidla o etapu poslední, ale co se týče délky trvání, o etapu nejdelší. Význam této etapy býval v minulosti často (zejména programátory) podceňován, větší společnosti jsou si ale dnes vědomy její důležitosti a hledají nástroje a postupy, jimiž by provoz a údržbu softwaru po jeho uvedení do provozního prostředí co nejvíce usnadnily. Tématem této práce je analýza, návrh a implementace systému, který se zaměřuje na automatizaci řešení jednoho z nejpalčivějších problémů, se kterým se softwarové společnosti mohou v etapě provozu a údržby setkat. Tímto problémem je analýza chyb softwaru v provozním prostředí.

Ačkoliv dnes existují velmi propracované metodiky návrhu a vývoje softwaru (např. Rational Unified Process nebo Model Driven Architecture), praktické zkušenosti vývojářů ukazují, že i přes pečlivé dodržování těchto metodik se v softwarových produktech stále vyskytují chyby. A i když se vyloučí chyby návrhářů a programátorů, pořád se zde bude vyskytovat množina chyb, s jejichž analýzou a odstraněním může mít společnost nemálo problémů, a pokud si chce u koncových uživatelů udržet dobré jméno a pověst, musí se i s těmito chybami umět vypořádat. Jedná se o chyby způsobené heterogenitou (různorodostí) provozního prostředí.

Při testování produktu je obvykle snaha nasimulovat testovací prostředí, které co možná nejvíce odpovídá budoucímu provoznímu prostředí. Při zakázkové tvorbě softwaru (např. dodávka podnikového informačního systému), kdy je provozní prostředí známo, není s jeho nasimulováním problém a v těchto případech je třeba výskyt chyb v etapě provozu a údržby vidět jako chybu dodavatele. Čím více však narůstá heterogenita provozního prostředí, tím více se zvětšuje pravděpodobnost neočekávaných chyb způsobených nemožností otestovat produkt ve všech možných provozních podmínkách. Při vývoji složitějšího softwarového produktu pro širší veřejnost (např. počítačová hra nebo kancelářský software) je tak využití systému pro analýzu provozních chyb velmi žádoucí.

Jednou ze společností, které se zabývají vývojem softwarových produktů nasazovaných do silně heterogenního provozního prostředí, je společnost AVG Technologies. Tato společnost je jedním z největších světových dodavatelů bezpečnostního softwaru určeného jak pro domácí, tak i pro firemní uživatele. V současné době podporuje společnost AVG Technologies nasazení a provoz svých produktů pod většinou operačních systémů rodin Microsoft Windows a Linux (viz [1]). Konfigurace pracovní stanice každého uživatele produktů společnosti AVG Technologies se tak může lišit. Provozní prostředí těchto produktů může být zdrojem mnoha dopředu neznámých problémů a chyb a je tak ideálním kandidátem pro nasazení systému pro analýzu provozních chyb. Při analýze a návrhu tohoto systému se díky ochotě ze strany společnosti AVG Technologies bude moci vycházet z požadavků, které na analýzu provozních chyb mají její zaměstnanci, především pak zaměstnanci z oddělení vývoje a technické podpory.

Kapitola *Problematika analýzy chyb softwaru v provozním prostředí* podrobně popisuje problematiku provozních chyb a představuje některá z existujících řešení používaných pro jejich analýzu. Na závěr této kapitoly jsou definovány vlastnosti a části systému pro analýzu provozních chyb. Kapitola *Formulace cíle práce* obsahuje definici cíle práce a blíže popisuje vztah společnosti AVG Technologies k tématu práce. V následující kapitole s názvem *Analýza požadavků* budou podrobně specifikovány a analyzovány požadavky na centrálu systému pro analýzu provozních chyb. Kapitola *Návrh* se věnuje návrhu centrály systému pro analýzu provozních chyb a jejími

hlavními výstupy budou návrh fyzického schématu databáze a návrh zpracování zpráv o provozních chybách. Kapitola *Implementace* definuje rozsah systému, který bude v rámci této práce realizován. V kapitole *Zhodnocení výsledků a rozšíření* bude uvedeno zhodnocení dosažených výsledků a možná budoucí rozšíření systému. Poslední kapitola *Závěr* je věnována závěrečnému shrnutí a přínosu práce. Za touto kapitolou je uveden seznam použité literatury a přílohy.

2 Problematika analýzy chyb softwaru v provozním prostředí

Tato kapitola se bude podrobně zabývat problematikou analýzy chyb softwaru v provozním prostředí. Nejprve bude neformálně definován pojem provozní chyby, který se bude v dalším textu této práce často vyskytovat a jeho přesný význam tak bude třeba znát. Další část kapitoly bude věnována procesu analýzy provozních chyb a kategoriím provozních chyb. V následující části kapitoly budou představena některá v současnosti používaná řešení pro analýzu provozních chyb. V závěru kapitoly pak bude definováno pět základních vlastností, které musí splňovat systém, aby jej bylo možné označit za systém pro analýzu provozních chyb.

2.1 Neformální definice pojmu provozní chyby

Obecně může být provozní chyba chápána jako chyba softwaru, která se projeví až po jeho nasazení do provozního prostředí. Tato teze vychází z potřeby softwarových společností odhalovat a analyzovat chyby softwaru vzniklé jeho používáním koncovými uživateli, která byla diskutována v kapitole 1.

Vývoj softwaru často probíhá v iteracích, kdy výstupem každé iterace je nějaký spustitelný kód, označovaný jako *build*. Ne vždy je ale build nasazen do provozního prostředí k užívání koncovými uživateli. Častěji se jedná pouze o mezivýsledek, který je používán vývojáři k evaluaci dosud implementovaných funkčností softwaru. Přestože se zde nejedná o nasazení do provozního prostředí, část softwaru je již používána a může generovat chyby, jejichž automatická analýza může vývojářům pomoci urychlit celkový proces vývoje softwaru.

Dalším běžně používaným postupem odstraňování chyb softwaru je tzv. *beta-testování*, kdy je produkt před svým finálním uvedením na trh dán k dispozici komunitě uživatelů označovaných jako *beta-testeři*. Tito uživatelé se často nacházejí mimo prostředí společnosti a rekrutují se z řad běžných uživatelů koncového produktu. Je tak zajištěno otestování funkčnosti produktu v prostředí, které se co nejvíce blíží výslednému provoznímu prostředí. Při beta-testování se díky větší heterogenitě provozního prostředí často vyskytne velké množství chyb. Automatická analýza těchto chyb může značně urychlit čas potřebný k přechodu od beta-testování produktu k jeho uvedení na trh.

Z předchozích odstavců vyplývá, že za provozní chybu lze považovat obecně jakoukoliv chybu, která vznikne používáním části softwarového produktu (případně již hotového produktu) označené jako oficiální výstup některé z provedených iterací životního cyklu softwaru.

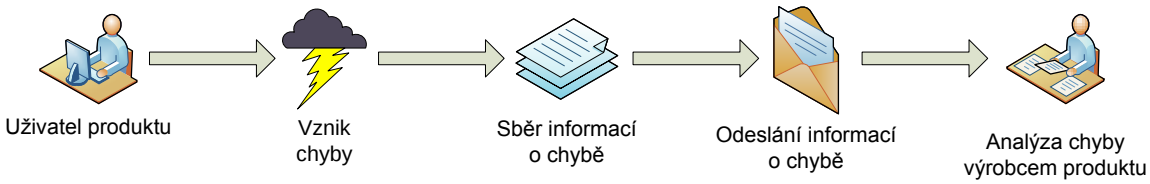
2.2 Analýza provozních chyb

Provozní chyby jsou odhalovány při používání softwarového produktu nebo jeho části. Tímto se liší od chyb ostatních, které se projevují a bývají odhaleny již při vývoji nebo při validačních testech prováděných před ukončením iterace životního cyklu – u těchto chyb je snadné najít jejich

příčinu, problematické místo ve zdrojovém kódu je často identifikováno přímo vývojářem a validační testy jsou psány s důrazem na otestování konkrétní funkčnosti produktu.

U provozních chyb je detekce zdroje chyby problematičtější. Úspěšnost nalezení zdroje chyby přímo závisí na množství informací, které jsou o chybě k dispozici. Není-li o chybě známo dostatečné množství informací (např. uživatel produktu pouze napíše, že došlo k pádu aplikace, když si na svoji stanici nainstaloval novou počítačovou hru, a nedodá další informace), nelze předpokládat úspěšné nalezení zdroje chyby.

Analýza provozních chyb je proces začínající vznikem chyby u uživatele produktu. Následně jsou sesbírány všechny relevantní informace související se vznikem chyby. V dalším kroku jsou pak tyto informace přeneseny na místo, kde proběhne jejich analýza (nejčastěji to je u výrobce produktu). Proces analýzy provozních chyb je znázorněn na obrázku 1.



Obrázek 1: Proces analýzy provozních chyb

2.3 Kategorie provozních chyb

Jak bylo uvedeno v části 2.2, pro úspěšné odhalení příčiny provozní chyby se vyžaduje přítomnost dostatečného množství informací o chybě. Podle druhu provozní chyby jsou pro odhalení příčiny chyby vyžadovány odlišné informace. Lze rozeznat tři základní kategorie provozních chyb, které se liší způsobem, jakým ovlivňují správnou funkčnost softwarového produktu:

- *Pád aplikace*. Chyba se projevuje neočekávaným ukončením aplikace. Informací, která je nezbytná k úspěšnému odhalení zdroje chyby, je zde tzv. *crash dump soubor*, ve světě operačních systémů rodiny Linux označovaný jako *core dump soubor*, generovaný operačním systémem při pádu aplikace (viz [2]). Tento soubor obsahuje informace o obsahu paměti, která byla operačním systémem aplikaci přidělena, obsahu zásobníku a další informace, které mohou být použity ke zjištění příčiny pádu aplikace.
- *Nesprávná funkčnost aplikace*. Chyba se projevuje špatnou realizací některé funkčnosti aplikace (např. přehrávač médií přehrává místo videa pouze šum). Informace potřebné k úspěšnému odhalení zdroje chyby se v tomto případě mohou lišit podle druhu aplikace. Často je například potřeba znát výpis obsahu adresářů, do kterých aplikace za běhu přistupuje.
- *Nedostupná funkčnost aplikace*. Chyba se projevuje nedostupností některé funkčnosti aplikace (např. nemožnost stažení aktualizací). Opět se zde nedá přesně určit, které informace jsou nezbytné pro odhalení zdroje chyby. V největším množství případů výskytů chyb z této kategorie se jedná a kolizi aplikace s jinou instalovanou aplikací nebo zásah škodlivého softwaru do korektního běhu aplikace, proto se při analýze těchto chyb často využívá seznamu instalovaných programů a běžících procesů.

2.4 Řešení používaná pro analýzu provozních chyb

Tato podkapitola se zabývá současným stavem problematiky analýzy provozních chyb. Budou zde představena řešení používaná pro analýzu provozních chyb některými velkými společnostmi a jedno řešení vzniklé na akademické půdě, které se nezabývá přímo analýzou provozních chyb, ale zaměřuje se především na sběr informací o provozních chybách (viz proces analýzy chyb popsaný v části 2.2).

2.4.1 Systémy pro crash reporting

V této části práce budou představena tři nejvýznamnější řešení používaná pro analýzu provozních chyb. Tato řešení se někdy označují jako systémy pro crash reporting, protože jejich hlavním úkolem je právě detekce chyb (a to převážně pádů aplikací – angl. crash) a odesílání zpráv o chybách (dále jen zpráv). Většina z níže uvedených řešení však slouží nejen k tomuto účelu, mezi další vlastnosti, které lze u těchto systémů vypozorovat, patří zejména analýza zasílaných informací a poskytování výsledků analýz uživatelům. Všechny níže uvedené systémy mají problém s detekcí provozních chyb, které vznikly jinak než pádem aplikace.

Microsoft Online Crash Analysis

Microsoft Online Crash Analysis (dále jen MOCA) je zřejmě nejznámějším řešením pro analýzu provozních chyb pod operačními systémy rodiny Windows od společnosti Microsoft. Téměř každý uživatel, který někdy měl na své pracovní stanici nainstalován některý z operačních systémů rodiny Windows, se jistě setkal se známým hlášením oznamujícím výskyt chyby a vyzývajícím jej k odeslání informací o vzniklé chybě do společnosti Microsoft k podrobné analýze.

Po odeslání zprávy je uživateli předán odkaz na webovou stránku systému MOCA, která obsahuje informace o výsledku analýzy. Pokud je chyba rozpoznána jako známá chyba, kterou lze odstranit, je uživateli zobrazen podrobný návod na odstranění chyby. V případě, že se jedná o novou chybu nebo o chybu, jejíž příčina nebyla dosud odhalena, má uživatel možnost se na stránky systému MOCA v budoucnu vrátit a sledovat stav jím odeslaných zpráv. K identifikaci uživatele a jeho zpráv využívá systém MOCA technologii .NET Passport.

Do systému MOCA lze odeslat zprávu o jakékoliv chybě, která vznikne na pracovní stanici s operačním systémem rodiny Windows. Chyby netýkající se přímo produktů společnosti Microsoft, jsou předány výrobcům příslušných produktů (pro získání informací o těchto chybách se výrobce produktu musí do systému MOCA zaregistrovat).

Výrobci produktů pro operační systémy rodiny Windows mohou systém MOCA použít k analýze provozních chyb svých produktů a jeho prostřednictvím pak mohou svým uživatelům nabídnout řešení těchto chyb. Výhodou tohoto přístupu k analýze provozních chyb je jeho nenáročnost, hlavní nevýhodou je pak samotné použití systému MOCA, který nemusí vždy poskytnout dostatek informací ke zdárnému nalezení zdroje chyby. Více informací o systému MOCA lze najít na webových stránkách systému (viz [3]) nebo v článku [4].

CrashReporter

CrashReporter je program pro zasílání informací o chybách vznikajících na operačních systémech rodiny Mac OS X vyvíjených společností Apple. Na rozdíl od MOCA je CrashReporter určen pouze k odesílání zpráv, nikoli k prohlížení výsledků jejich analýz. Uživateli, který odeslal zprávu, není dána žádná možnost, jak se dostat k výsledku provedené analýzy. Zprávy lze v původní verzi programu CrashReporter odeslat pouze do společnosti Apple, existují však moduly, které tento nedostatek odstraňují a umožňují odesílání zpráv přímo výrobcům produktů pro operační systémy rodiny Mac OS X. Více informací o programu CrashReporter lze nalézt např. v [5].

Talkback

Talkback je zástupcem systémů pro crash reporting, které jsou svázány s konkrétním produktem, v tomto případě s produkty open-source komunity Mozilla.org (např. webový prohlížeč Firefox nebo emailový klient Thunderbird). Jedná se o program, který je přidáván k instalaci produktů komunity Mozilla.org a jehož úkolem je detekovat pády těchto produktů.

Po pádu aplikace zobrazí program Talkback dialog pro odeslání zprávy. Zpráva se odesílá na servery komunity Mozilla.org a na veřejně přístupných webových stránkách lze po zadání identifikátoru zprávy (vygenerován programem Talkback) zobrazit informace o výsledku její analýzy.

V současné době je snaha nahradit Talkback programem, který je vyvíjen v rámci open-source projektu s názvem Breakpad (viz [6] a [7]). Více informací o samotném programu Talkback lze nalézt např. v [8].

2.4.2 Ladění provozních chyb

Systémy uvedené v části 2.4.1 analyzují zprávy, které zpravidla obsahují crash dump soubory a další informace popisující stav aplikace v okamžiku jejího pádu. Na základě těchto informací ale nelze odhalit zdroj chyby vždy. Pokud se jedná o chybu, která vznikne jako důsledek několika po sobě následujících akcí, nelze analýzou těchto informací najít zdroj chyby vůbec. K úspěšnému odhalení příčiny těchto chyb by mohly pomoci takové informace, pomocí nichž by šlo průběh vzniku chyby zrekonstruovat. Na základě těchto informací by mohli vývojáři aplikaci ladit a předávat jí potřebné vstupy tak, aby mohli chybu zpětně navodit a zjistit její přesnou příčinu.

Sběrem informací, které by mohly vývojářům softwarových produktů umožnit rekonstrukci provozních chyb, se zabývá článek [9]. Autoři zde prezentují systém, který je schopný sledovat chování určité aplikace a toto chování zaznamenávat ve formě speciálních instrukcí do log souboru. Při vzniku chyby se pak systém snaží velikost log souboru co nejvíce zmenšit tak, aby chování v něm zaznamenané stále vykazovalo vzniklou chybu. Chování zaznamenané v log souboru pak může být přehráno a použito vývojáři při ladění aplikace.

Nasazení podobného systému pro ladění provozních chyb v praxi však brání několik faktů. Prvním, na který poukazují již sami autoři, je omezení systému pro zaznamenávání chování aplikací pouze na sledování chování aplikací napsaných v jazyce C/C++. Dalším problémem, který musí být vyřešen, je způsob, jak zaznamenávat chování aplikace s pokročilým grafickým uživatelským rozhraním (v dalším textu bude používána zkratka GUI z anglického Graphical User Interface). V současné době je systém schopen sledovat práci aplikace se vstupně/výstupními proudy dat (např. soubory, klávesnice, síť).

Všechny výše uvedené problémy mohou být jistě v budoucnu vyřešeny. Největší omezení této techniky však plyne z její podstaty. Pokud ji budou chtít společnosti vyvíjející softwarové produkty používat, budou muset společně s finálním produktem distribuovat aplikaci pro záznam jeho chování. Tato aplikace bude muset být spuštěna zároveň s finálním produktem a v závislosti na chování finálního produktu může její běh neúměrně snížit výkon pracovní stanice uživatele a tím samozřejmě i běh finálního produktu. Použití této techniky tak ani po vyřešení problémů popsaných v předchozím odstavci nebude vhodné pro aplikace, které již sami o sobě poměrně výrazně zatěžují výkon pracovní stanice uživatele (např. antivirové programy nebo počítačové hry).

2.5 Vlastnosti systému pro analýzu provozních chyb

Aby bylo možné nějaký systém označit za systém pro analýzu provozních chyb, musí mít tento systém následujících pět vlastností:

1. *Detekce provozních chyb*. Systém musí být schopen detekovat provozní chyby sledovaného produktu. Čím více kategorií provozních chyb (viz část 2.3) je schopen detekovat, tím lépe.
2. *Generování zpráv*. Systém musí být schopen generovat zprávy, které obsahují co možná nejvíce relevantních informací.
3. *Přenos zpráv*. Systém musí být schopen přenosu zpráv na místo, kde budou tyto zprávy analyzovány.
4. *Analýza zpráv*. Systém musí poskytovat automatizovaný proces analýzy zpráv a prostředky pro sledování výsledků provedených analýz.
5. *Zpětná vazba na uživatele produktu.* Systém musí poskytnout koncovému uživateli řešení, které vede k odstranění provozní chyby (pokud je takové řešení k dispozici).

Z výše uvedených vlastností plyne, že se každý systém pro analýzu provozních chyb musí skládat ze tří úzce propojených částí:

- *Monitor*. Program, který sleduje chování produktu, detekuje jeho provozní chyby a generuje zprávy (zajištění vlastností 1 a 2). Je nasazen v provozním prostředí společně se sledovaným produktem.
- *Centrála*. Hlavní část systému pro analýzu provozních chyb. Poskytuje prostředky pro automatickou analýzu zpráv a pro sledování výsledků provedených analýz (zajištění vlastnosti 4). Je nasazena u výrobce sledovaného produktu.
- *Přenosový kanál*. Protokol pro přenos zpráv mezi monitorem a centrálou (zajištění vlastnosti 3).

Zajištění vlastnosti 5 může být v systému realizováno různými způsoby. Často je tato vlastnost zabudována do všech tří částí systému tak, že pokud řešení vedoucí k odstranění provozní chyby bylo centrálou nalezeno, je toto řešení uživateli zasláno prostřednictvím přenosového kanálu jako odpověď na odeslání zprávy a následně se objeví v GUI monitoru (např. systém MOCA, viz

část 2.4.1). Jiným způsobem může být odeslání řešení vedoucího k odstranění chyby na emailovou adresu uživatele (tato adresa je pak jednou z informací uváděných ve zprávě).

3 Formulace cíle práce

Cílem této práce je analyzovat požadavky na centrálu systému pro analýzu provozních chyb (viz část 2.5), na základě požadavků vytvořit návrh centrály a realizovat vhodnou část návrhu tak, aby mohla být vyhodnocena jeho správnost a praktická využitelnost centrály. Přestože se při analýze a návrhu bude vycházet z požadavků společnosti AVG Technologies, neměl by být výsledný návrh vytvářen pouze s ohledem na potřeby této společnosti, všechny požadavky by měly být co možná nejvíce zobecněny, aby mohla být centrála nasazena k analýze provozních chyb produktů libovolné společnosti zabývající se vývojem softwaru. Společnost AVG Technologies zde působí jako konzultant, který má s problematikou analýzy provozních chyb praktické zkušenosti, bez nichž by návrh centrály systému pro analýzu provozních chyb nebyl možný. Společnost AVG Technologies bude prvním uživatelem centrály a provozní prostředí jejích produktů bude prvním zdrojem chyb, na kterých bude moci být ověřena funkčnost implementovaného řešení.

4 Analýza požadavků

V této kapitole se nejprve objeví neformální specifikace požadavků na centrálu systému pro analýzu provozních chyb, ze které bude dále vycházeno při modelování funkčních požadavků pomocí techniky případů použití a při specifikaci požadavků nefunkčních. Na základě specifikovaných požadavků bude na konci kapitoly uveden model konceptuálních tříd.

4.1 Neformální specifikace požadavků

Neformální specifikace požadavků slouží jako východisko pro jejich další analýzu. Porozumění neformálnímu popisu požadavků je důležité pro pochopení základní funkčnosti centrály systému pro analýzu provozních chyb. Při tvorbě neformální specifikace požadavků se vycházelo z potřeb zaměstnanců společnosti AVG Technologies, zároveň byl ale kladen důraz na odstranění, případně přeformulování požadavků, které by mohly způsobit komplikace při nasazení centrály do prostředí jiných společností.

4.1.1 Integrace centrály s monitorem a přenosovým kanálem

V současné době existuje poměrně velké množství nástrojů zajišťujících crash reporting (viz část 2.4.1), jedná se tedy o nástroje, které by mohli v systému pro analýzu provozních chyb vystupovat jako monitor nebo přenosový kanál. Společnosti zabývající se vývojem softwaru si ale při analýze provozních chyb pouze s nástroji pro crash reporting nevystačí. Stále častěji je vyžadována přítomnost mechanismu pro automatizaci analýz zpráv o chybách, přičemž ideálním řešením by byl systém, který by bylo možné jednoduše propojit s existujícími nástroji pro crash reporting a vytvořit tak plnohodnotný systém pro analýzu provozních chyb obsahující všechny jeho základní části (monitor, přenosový kanál i centrálu). Vazba pouze na jeden konkrétní typ monitoru a přenosového kanálu by byla vážným nedostatkem výrazně omezujícím možnosti využití centrály.

Referenčním monitorem, který bude použit při vývoji centrály, bude program AVG Diag, který slouží k detekci a zasílání zpráv o provozních chybách produktů společnosti AVG Technologies. Přenosovým kanálem, který program AVG Diag používá, je protokol SMTP.

4.1.2 Různý formát zpráv

Existuje-li požadavek na schopnost spolupráce centrály s různými implementacemi monitoru a přenosového kanálu, pak také musí existovat požadavek na schopnost centrály zpracovat zprávy odlišných formátů, které tyto implementace mohou produkovat.

Pokud není vyžadována okamžitá zpětná vazba na uživatele (viz část 2.5), pak lze na celý problém integrace centrály s monitorem a přenosovým kanálem nahlížet jako na problém zpracování zpráv v různých formátech, protože právě zprávy jsou jedinou entitou, která musí být mezi jednotlivými částmi systému pro analýzu provozních chyb přenášena, a to navíc pouze ve směru od monitoru k centrále. Bude-li možné centrálu konfigurovat pro zpracování zpráv v různých formátech, bude tímto centrála na použitém monitoru a přenosovém kanálu nezávislá a problém její integrace s monitorem a přenosovým kanálem bude vyřešen.

4.1.3 Příjem zpráv

Produkty společnosti AVG Technologies jsou v současné době nainstalovány na více než 70 miliónech počítačů (viz [10]). V případě výskytu závažné provozní chyby lze očekávat příchod značného množství zpráv a celý systém pro analýzu provozních chyb by měl tento nápor vydržet. Vlastní řešení možného zahlcení zprávami by měl řešit použitý přenosový kanál, centrála by tedy měla předpokládat, že ve výše uvedeném případě zahlcení přenosového kanálu nenastane a o zpracování se skutečně bude hlásit velmi velké množství zpráv současně. Centrála by měla být schopna i v takovém případě zprávy zpracovat při zachování akceptovatelné doby zpracování (tj. doby od příjmu zprávy po dokončení její analýzy).

4.1.4 Typ aplikace centrály

GUI centrály by mělo být implementováno jako webová aplikace – portálové řešení pracující nad databází s informacemi získanými analýzou zpráv. Na základě tohoto požadavku byl také odvozen název práce – Crash Analysis Portal. Zpracování příchozích zpráv a ukládání získaných informací do databáze by ale mělo být zajištěno pomocí entit (např. pomocí služeb operačních systémů rodiny Windows nebo unixových démonů) nasazených na jiných strojích než samotná webová aplikace (z důvodu možného přílišného vytížení stroje webové aplikace).

4.1.5 Provozní prostředí centrály a její uživatelé

Protože zprávy obsahují citlivé informace o koncových uživatelích sledovaných produktů, měla by být centrála nasazena v prostředí interní sítě společnosti, do níž mají přístup pouze její zaměstnanci.

Centrála by měla poskytovat flexibilní mechanismus správy uživatelů pomocí oprávnění a rolí. Uživatel by měl mít vždy přiřazenu právě jednu roli, ke které se vážou oprávnění. Uživatel s přiřazenou rolí pak může vykonat pouze ty akce, ke kterým má jeho role oprávnění. Všechny významné akce, které bude možné v centrále vykonávat, by měly ke svému provedení vyžadovat speciální oprávnění.

4.1.6 Analýza zpráv

Zprávy mají vždy formu jednoho souboru, většinou se jedná o archiv v některém z běžně používaných kompresních formátů (např. ZIP, RAR, u programu AVG Diag se jedná o formát EML). Soubor se zprávou vždy obsahuje další soubory, které mohou obsahovat nejrůznější informace závisející obvykle na kategorii chyby (viz část 2.3). Může se jednat o informace o stavu aplikace v době jejího pádu, záznamy o konfiguraci pracovní stanice uživatele, informace z log souborů sledované aplikace apod. Tabulka 1 ukazuje soubory, které se mohou nacházet ve zprávě vygenerované programem AVG Diag pro produkty verze AVG 7.5.

Analýza zpráv je proces, kdy jsou ze souborů obsažených ve zprávě (případně z vlastního souboru zprávy) získávány důležité informace ve formě řetězců znaků (tyto informace budou v dalším textu označovány jako *analytické položky*), na základě nichž jsou poté vykonávány další akce, mezi které patří zejména vytváření problémů (viz níže). Soubory, ve kterých lze hledat analy-

tické položky, se od ostatních souborů liší a budou dále označovány jako *významné soubory*. Centrála by měla umožnit definování různých typů významných souborů a analytických položek.

Jak mezi významnými soubory, tak i mezi analytickými položkami, by měla centrála umožnit definování vztahu rodič-potomek. Při analýze zpráv jsou z původních souborů zprávy vytvářeny soubory jiné (potomci), typickým příkladem může být analýza crash dump souboru, při které je vytvořen soubor analýzy a teprve z něho má smysl získávat analytické položky. Vztah rodič-potomek by tedy měl vzniknout mezi crash dump souborem a souborem s jeho analýzou. U analytických položek vztah rodič-potomek vyjadřuje příslušnost některých položek k položce jiné, nadřazené. V souboru s analýzou crash dump souboru se mohou vyskytovat položky reprezentující moduly sledované aplikace a o každém modulu jsou v souboru přítomny další informace (položky) o stavu modulu v okamžiku pádu aplikace. Vztah rodič-potomek by měl v tomto případě vzniknout mezi položkou označující modul a položkami s dalšími informacemi o modulu.

Různé hodnoty analytických položek mohou často označovat stejnou entitu (např. označení shodné verze operačního systému v různých jazycích). Je proto vhodné, aby centrála umožnila definovat alias hodnot pro problematické hodnoty analytických položek.

Přítomnost některých typů analytických položek a souborů ve zprávě, případně jejich kombinace, může jednoznačně identifikovat problém a na základě těchto informací by pak měla být v databázi vytvořena *instance problému* (záznam o problému), přičemž vždy musí být jasné, ze kterých analytických položek, souborů a zprávy byl problém vytvořen. Každý problém má svou *hodnotu*, která je dána hodnotami analytických položek a názvy souborů, na jejichž základě byl problém vytvořen.

U vlastního souboru zprávy je nutné jeho uložení na úložiště s rychlým přístupem, ze kterého jej půjde pomocí GUI centrály získat pro případné manuální zpracování (viz část 4.1.15). Pro významné soubory by měla být možnost jejich uložení na takové úložiště volitelná.

Tabulka 1: Soubory zprávy vygenerované programem AVG Diag pro produkty verze AVG 7.5

Soubor	Obsažené informace
avg_info.xml	informace o instalaci produktu (číslo verze programu AVG Diag, jméno registrovaného uživatele produktu, jméno firmy, číslo verze produktu, licenční číslo a další)
system_info.xml	informace o operačním systému (typ operačního systému, verze operačního systému, instalovaný servisní balík, systémový čas a datum a další)
env_info.xml	informace o programovém prostředí (seznam běžících procesů, seznam procesů spouštěných při startu systému, seznam instalovaných aplikací a další)
<název počítače>(COMPUTER).cab	informace o konfiguraci produktu na pracovní stanici (konfigurace platná pro všechny uživatele)
<jméno uživatele>.<název počítače>(CURRENT USER).cab	informace o konfiguraci produktu nastavené pro aktivního uživatele
<jméno uživatele>.<název počítače>.cab	informace o konfiguraci produktu nastavené pro další uživatele pracovní stanice (pro každého uživatele pracovní stanice je vytvořen zvláštní soubor)
logs.cab	všechny dostupné protokolované záznamy (log soubory)
dumpAVG.cab	všechny dostupné záznamy o pádech produktu
dumpSYS.cab	všechny dostupné záznamy o pádech jiných produktů nebo pádech operačního systému
fwdiag.cab	diagnostické záznamy komponenty AVG Firewall

4.1.7 Doplňkové analýzy a modifikovatelnost

Během analýzy zprávy nemusejí být nutně všechny získané informace ukládány do databáze, u některých informací to nemusí být ani žádoucí. Analýzou zprávy mohou být získány informace, které jsou určeny pouze jako doplněk k získaným analytickým položkám a nebude z nich potřeba generovat statistiky ani vytvářet problémy. Jejich zobrazení v detailním pohledu na zprávu by ale mohlo pomoci při hledání problémů během manuálního zpracování (viz část 4.1.15). Během analýzy zprávy by měly být tyto informace sesbírány a vytvořen z nich úsek HTML kódu, který bude možné začlenit do detailního pohledu na zprávu. Tento HTML kód by měl být označen jako specifický typ významného souboru ukládaný na úložiště s rychlým přístupem. Takto vytvořený významný soubor bude v dalším textu označován jako *doplňková analýza*.

Častým požadavkem při provozu centrály bude jistě přidání či odebrání doplňkové analýzy z procesu analýzy zpráv. Podobně mohou vyvstat požadavky na vyhledávání nových analytických položek, zpracování zpráv o chybách v novém formátu a mnoho dalších požadavků, které vyžadují změnu procesu analýzy zpráv. Centrála by proto měla umožnit proces analýzy zpráv jednoduchým způsobem modifikovat.

4.1.8 Stavy problémů a komentáře

Pro usnadnění orientace v množství problémů, které se mohou v databázi centrály nacházet, by mělo být možné označit každý problém příznakem určujícím jeho stav. Centrála by měla umožnit definování různých příznaků, čímž bude zajištěna flexibilita při nasazení centrály do prostředí nové společnosti, kdy si tato společnost bude moci definovat příznaky označující stavy problémů, které se již v jejím prostředí (např. v systému pro sledování chyb) používají. Některé stavy by však měly být výchozí a neměnné, jedná se především o ty stavy, které nově vzniklým problémovým případům přiřazuje proces analýzy zpráv.

Při řešení problému by uživatelé centrály měli mít možnost přiřazovat k němu komentáře, jimiž by mohli blíže popsat průběh řešení konkrétního problému.

4.1.9 Propojení se systémem pro sledování chyb

Pokud problém představuje chybu sledovaného produktu, měl by být záznam o této chybě vytvořen v systému pro sledování chyb (pokud společnost nějaký podobný systém používá, ve společnosti AVG Technologies je tímto systémem open-source řešení Bugzilla). Problém by měl obsahovat jednoznačný odkaz na záznam o chybě v systému pro sledování chyb společně s dalšími informacemi o stavu chyby v tomto systému. U problémů, které mají tento odkaz nastaven, by měla centrála zajistit periodickou kontrolu stavu informací o chybě a udržovat informace v centrále v souladu s informacemi v systému pro sledování chyb.

4.1.10 Propojení se systémem pro sledování komunikace s koncovými uživateli

Jestliže společnost využívá nějaký systém pro sledování komunikace s koncovými uživateli svých produktů (ve společnosti AVG Technologies to je externí společností dodávaný a spravovaný sys-

tém Genesis), pak by mělo existovat propojení mezi zprávami v tomto systému a zprávami v centrále. Díky existenci tohoto propojení budou mít uživatelé primárně pracující se systémem pro sledování komunikace s koncovými uživateli přístup k výsledkům analýz, které pro zprávu provedla centrála, a tyto analýzy již nebudou muset vykonávat manuálně.

4.1.11 Analýza na vyžádání

Další žádanou vlastností, která umožní využití centrály především uživatelům nacházejícím se mimo interní síť společnosti, je tzv. *analýza na vyžádání*. Jedná se o analýzu zprávy (nebo její části), kterou externí uživatel odešle na speciální emailovou adresu. Z této adresy by měly být všechny příchozí zprávy periodicky odebírány a předkládány k procesu analýzy, jenž proběhne stejným způsobem jako analýza běžných zpráv přijatých od monitoru, ale výsledky této analýzy budou v databázi uloženy pouze omezenou dobu, která je potřebná k získání všech informací o případných problémech a stavech jejich řešení. Z těchto informací pak bude sestavena HTML stránka, která se odešle zpět na emailovou adresu uživatele, který zprávu odeslal.

4.1.12 Zpětná vazba na uživatele produktu

Přítomnost zpětné vazby na uživatele produktu je jednou z pěti základních vlastností systému pro analýzu provozních chyb. Jak bylo uvedeno v části 2.5, často bývá tato vlastnost zabudovávána do všech tří částí systému. Při vývoji centrály, která má spolupracovat s libovolnou implementací monitoru a přenosového kanálu, tato varianta možná není. Pokud je ve zprávě uvedena emailová adresa uživatele produktu, může být tento kontaktován, jakmile bude známo řešení jeho problémů (budou vyřešeny všechny problémy ve zprávě). Obecná centrála tedy problém zpětné vazby na uživatele produktu nemusí řešit, v případě přítomnosti emailové adresy koncového uživatele produktu ve zprávě jej mohou uživatelé centrály kontaktovat pomocí standardních prostředků elektronické pošty.

4.1.13 Sledování provozu centrály

Provoz centrály by měl být pečlivě sledován. Akce systémové, tj. provoz systému bez interakce s uživateli, by měly být protokolovány do systémových log souborů. Akce uživatelů, tj. akce, ke kterým je potřeba oprávnění, by měly být protokolovány do databáze centrály, aby byly přes GUI centrály rychle dostupné administrátorům systému a mohlo v nich být snadno vyhledáváno. Společně s typem uživatelské akce a identifikátorem uživatele, který ji provedl, by měly být protokolovány všechny důležité parametry akce tak, aby mohla být akce v případě potřeby stornována a databáze mohla být uvedena do stavu před provedením akce (manuálně, pomocí běžných operací, tj. jestliže některý uživatel změnil stav problému, pak z protokolovaných akcí půjde vyčíst, kdo změnu stavu provedl a jaký byl stav před provedenou změnou, a pomocí provedení operace změny stavu bude moci být problému nastaven stav původní).

4.1.14 Statistiky

Jedním z hlavních výstupů, které centrála svým uživatelům poskytuje, jsou statistiky. Centrála by měla obsahovat GUI pro zobrazení následujících statistik:

- *Distribuce počtu nalezených problémů přes hodnoty analytických položek za dané časové období*. Statistika udává velikost vlivu konkrétní hodnoty analytické položky na výskyt problémů v daném časovém období. Pro typ analytické položky musí být možné nastavit příznak označující jeho relevantnost pro tento typ statistiky.
- *Počty vytvořených problémů se shodnou hodnotou za dané časové období*. Statistika udává problémy, které se v daném časovém období vyskytovaly nejčastěji.

4.1.15 Automatické a manuální zpracování zpráv

Po přijetí zprávy by centrála měla automaticky provést její analýzu zahrnující detekci významných souborů, vyhledání analytických položek, provedení doplňkových analýz, vytvoření problémů a jejich nastavení do výchozích stavů. Navíc, pokud již v databázi centrály existuje problém mající stejnou hodnotu jako některý z nově vytvořených problémů, a tento problém má nastaven odkaz na chybu do systému pro sledování chyb, pak by měla centrála tento odkaz nastavit také u příslušných nově vytvořených problémů. Celý výše uvedený postup bude v dalším textu označován jako *automatické zpracování zpráv* a je řízen *konfigurací procesu zpracování zpráv*.

I přes výše popsané automatické zpracování zpráv je stále potřeba některé zprávy zpracovat manuálně (např. zprávy neobsahující žádný problém) a navíc některé činnosti týkající se zpráv automatizovat nelze vůbec nebo jen velmi obtížně. Centrála by proto měla poskytovat následující prostředky pro *manuální zpracování zpráv*:

- Snadné získání souboru zprávy a významných souborů (viz část 4.1.6).
- Přidávání komentářů k problémovým případům a změna jejich stavu (viz část 4.1.8).
- Manuální vytváření a rušení problémů s omezením, že zrušit lze pouze manuálně vytvořený problém.
- Manuální nastavení odkazu na chybu v systému pro sledování chyb pro konkrétní problém (viz část 4.1.9).
- Reanalýza zprávy. Centrála by měla na vyžádání umožnit opětovné provedení automatického zpracování zprávy (má význam pouze pokud byl proces automatického zpracování od poslední provedené analýzy dané zprávy modifikován, nebo pokud zpracování z nějaké příčiny selhalo a uživatel si chce ověřit, že k selhání dojde i při opětovném provedení zpracování).

4.1.16 Vstupy a výstupy centrály

Vstupy pro centrálu budou zprávy získané prostřednictvím monitoru a přenosového kanálu nebo díky požadavku některého externího uživatele na analýzu na vyžádání (viz část 4.1.11). Centrála by měla poskytovat svým uživatelům následující dva základní typy výstupů:

- GUI jako základní pohled na data uložená v databázi centrály (pomocí statistik a dalších pohledů na data);
- HTML stránku odesílanou externímu uživateli jako výsledek analýzy na vyžádání.

4.2 Funkční požadavky

V časti 4.1 byly neformálně popsány žádané vlastnosti, omezení a požadavky na centrálu systému pro analýzu provozních chyb. Na základě tohoto popisu budou v této podkapitole definovány případy použití reprezentující funkční požadavky, tj. požadavky na služby centrály. Interakce mezi uživateli centrály a případy použití budou zachyceny pomocí diagramu případů použití.

4.2.1 Aktéři

Požadavek na role a oprávnění z části 4.1.5 znemožňuje definování konkrétních typů aktérů. Po nasazení centrály do provozního prostředí se role uživatelů a k nim přiřazená oprávnění mohou dynamicky měnit. Pro modelování případů použití budou proto definováni dva abstraktní aktéři (role) *uživatel* a *administrátor*, kdy uživatel představuje skupinu uživatelů centrály, která má za úkol především využívat její možnosti pro analýzu zpráv, a administrátor skupinu uživatelů centrály, která se stará o správu konfiguračních nastavení a definování procesu zpracování zpráv.

Vazba mezi aktérem a případem použití uvedená v diagramu případů použití značí možnost spuštění případu použití daným aktérem, nebo, vyjádřeno pomocí pojmů role a oprávnění, přítomnost oprávnění k provedení akce (definované případem použití) v dané roli. Některé z případů použití uvedených v následujících částech této podkapitoly zahrnují více různých akcí typu vytvoření, zobrazení, editování, smazání entity (angl. CRUD akce podle slov create, retrieve, update, delete). Vytvoření zvláštních případů použití pro každou z těchto akcí by pouze zvýšilo složitost modelu případů použití a nepřineslo by žádnou vyšší informační hodnotu. V souladu s požadavkem na role a oprávnění však bude možné vykonání každé takové akce podmínit příslušností konkrétního oprávnění k roli uživatele, který chce danou akci provést.

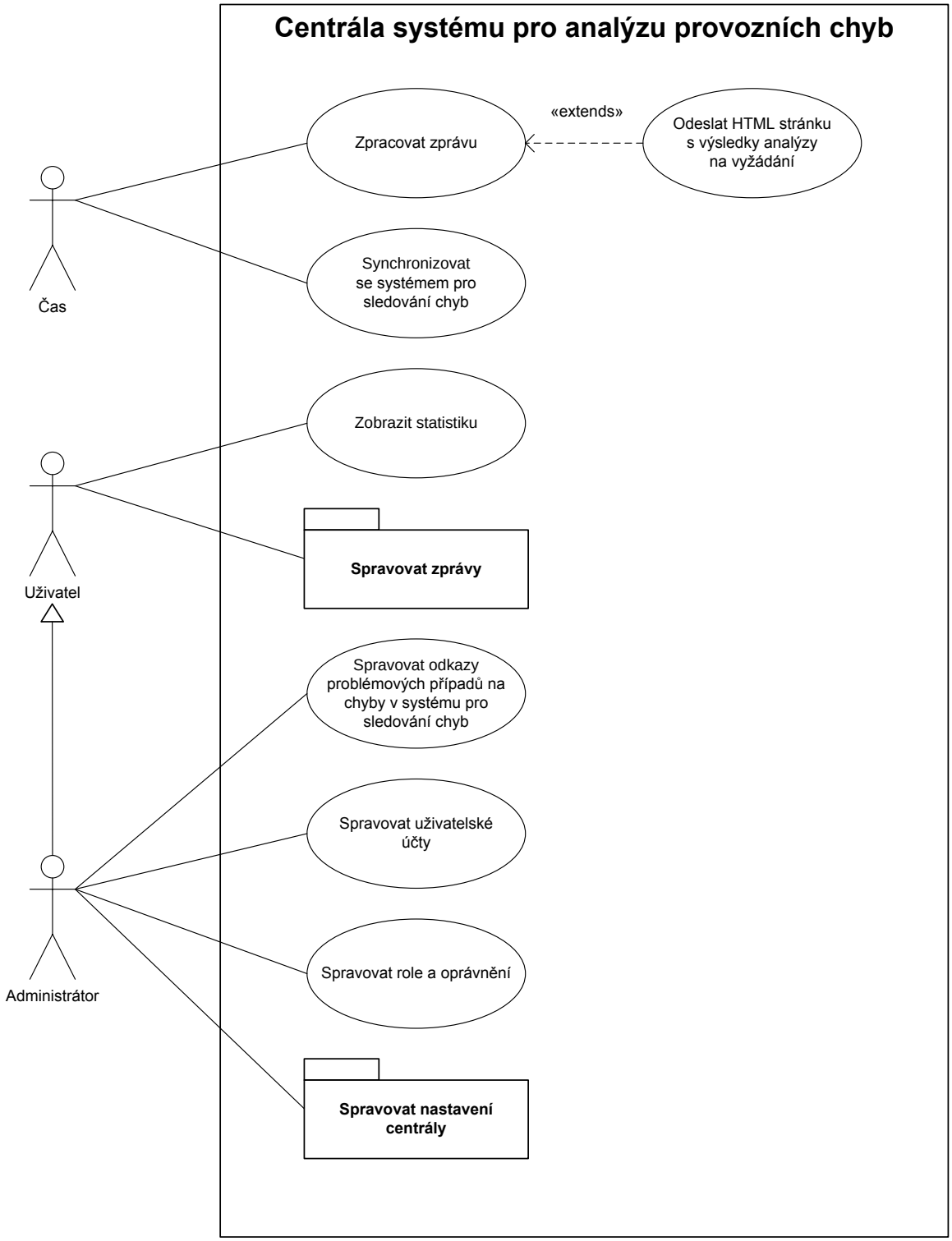
Dalším aktérem, který se v modelu funkčních požadavků bude vyskytovat, je *čas*. Případy použití spouštěné tímto aktérem si lze představit jako asynchronně spouštěné akce reagující na výskyt nějaké externí události (např. příchod zprávy).

4.2.2 Případy použití spouštěné časem

V této části práce budou popsány případy použití, které jsou spouštěny aktérem čas. Tyto případy použití lze nalézt na obrázku 2.

Zpracovat zprávu

Vstupní podmínkou pro spuštění tohoto případu použití je přítomnost zprávy na vstupu centrály. Zpráva je ze vstupu předána části centrály, která zajišťuje automatické zpracování příchozích zpráv. Pro automatické zpracování zprávy je použita aktuální konfigurace procesu zpracování zpráv. Pokud přišla zpráva ze vstupu určeného pro analýzy na vyžádání, je případ použití rozšířen případem použití Odeslat HTML stránku s výsledky analýzy na vyžádání.



Obrázek 2: Hlavní diagram případů použití

Odeslat HTML stránku s výsledky analýzy na vyžádání

Tento případ použití rozšiřuje případ použití Zpracovat zprávu, pokud tímto případem použití zpracovávaná zpráva přišla ze vstupu určeného pro analýzy na vyžádání. Centrála získá z databáze všechny informace o provedeném zpracování (analytické položky, významné soubory, doplňkové

analýzy a problémy), vytvoří z nich HTML stránku a tuto odešle na emailovou adresu externího uživatele centrály uvedenou ve zprávě.

Synchronizovat se systémem pro sledování chyb

Vstupní podmínkou pro spuštění tohoto případu použití je existence alespoň jednoho problému v databázi centrály, který má nastaven odkaz na chybu v systému pro sledování chyb. Centrála se spojí se systémem pro sledování chyb a zkontroluje informace o chybě uvedené v odkazu na chybu u všech problémů, které jej mají nastaven. V případě, že se informace uvedené v odkazu liší od informací v systému pro sledování chyb, jsou informace v odkazu aktualizovány (přepsány informacemi ze systému pro sledování chyb).

4.2.3 Případy použití spouštěné uživatelem

V této části práce budou popsány případy použití, které jsou spouštěny především aktérem uživatel, díky generalizaci aktérů však mohou být spouštěny i aktérem administrátor. Tyto případy použití lze nalézt na obrázcích 2 a 3.

Zobrazit statistiku

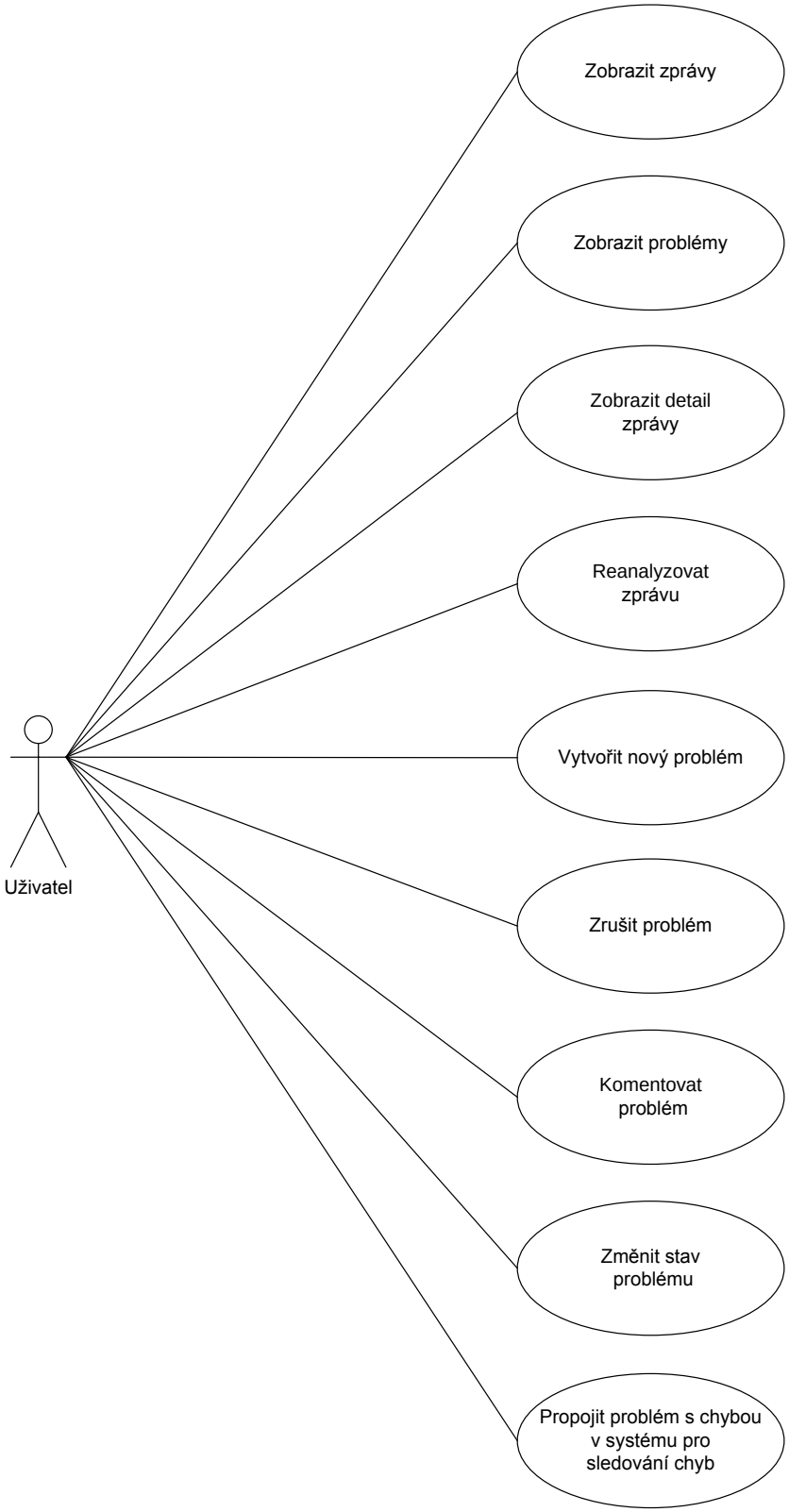
Aktér si vybere jeden ze dvou možných typů statistik, které může centrála zobrazit (viz část 4.1.14). Pro zvolený typ statistiky vybere její parametry (časové období, případně typ analytické položky). Centrála poté zobrazí statistiku ve formě tabulky a grafu.

Zobrazit zprávy

Aktér zadá úvodní parametry pro zobrazení zpráv (časové období, hodnoty analytických položek, přítomnost významných souborů, počet problémů ve zprávě apod.) a vlastnosti zpráv, které si chce nechat zobrazit. Centrála poté zobrazí všechny zprávy z databáze (pouze uživatelem zvolené vlastnosti zpráv) vyhovující zadaným parametrům. Úvodní parametry může aktér dále měnit a zprávy tak filtrovat podle jiných kritérií. Aktér může zobrazené zprávy řadit podle několika jejich vlastností zároveň.

Zobrazit problémy

Aktér zadá úvodní parametry pro zobrazení problémů (časové období, hodnoty analytických položek, přítomnost významných souborů ve zprávách, do nichž problémy patří, hodnotu problému apod.) a vlastnosti problémů, které si chce nechat zobrazit. Centrála poté zobrazí všechny problémy z databáze vyhovující zadaným parametrům. Úvodní parametry může aktér dále měnit a problémy tak filtrovat podle jiných kritérií. Aktér může zobrazené problémy řadit podle několika jejich vlastností zároveň.



Obrázek 3: Diagram případů použití balíčku Spravovat zprávy

Zobrazit detail zprávy

Aktér vybere zprávu, jejíž detail si chce nechat centrálou zobrazit. Centrála zobrazí detail obsahující všechny automaticky i manuálně získané informace ze zprávy (analytické položky, významné soubory, problémy, doplňkové analýzy). Skrze toto detailní zobrazení může aktér získat k manuálnímu zpracování jak vlastní soubor zprávy, tak i její významné soubory, které byly uloženy na úložiště s rychlým přístupem.

Reanalyzovat zprávu

Aktér vybere zprávu, kterou si přeje reanalyzovat. Pro reanalýzu dále zvolí některou z dostupných konfigurací procesu zpracování zpráv. Centrála poté provede reanalýzu zprávy a zobrazí její výsledky, přičemž informace lišící se od výsledku původní analýzy budou zvýrazněny. Aktér potom může výsledkem reanalýzy nahradit výsledek analýzy původní.

Vytvořit nový problém

Aktér vybere zprávu, ke které chce vytvořit nový problém. Aktér zvolí vytvoření nového problému a zadá k tomuto úkonu vyžadované informace (na základě kterých souborů zprávy a analytických položek se má problém vytvořit, povinný komentář a výchozí stav problému). Centrála poté vytvoří k vybrané zprávě nový problém.

Zrušit problém

Aktér vybere problém, jenž si přeje zrušit (může vybrat pouze problémy vytvořené manuálně, tj. pomocí případu použití Vytvořit nový problém). Centrála poté vybraný problém zruší.

Komentovat problém

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s komentáři problémů. Jedná se o akce vytvoření, zobrazení a smazání komentáře problému. Aktér se standardním oprávněním pro smazání komentáře může smazat pouze jím vytvořené komentáře, speciální administrátorské oprávnění pro smazání komentáře umožňuje smazání jakéhokoliv komentáře.

Změnit stav problému

Aktér vybere problém, jehož stav si přeje změnit a vybere z nabídky stavů problémů stav nový. Centrála poté přiřadí problému zvolený stav.

Propojit problém s chybou v systému pro sledování chyb

Aktér vybere problém, jemuž chce nastavit odkaz na chybu v systému pro sledování chyb. Aktér dále poskytne centrále identifikátor chyby v systému pro sledování chyb, na jehož základě bude v databázi centrály vytvořen pro problém odkaz na tuto chybu. Další informace aktér u odkazu na chybu nenastavuje, tyto jsou periodicky aktualizovány spouštěním případu použití Synchronizovat se systémem pro sledování chyb.

4.2.4 Případy použití spouštěné administrátorem

V této části práce budou popsány případy použití, které jsou spouštěny výhradně aktérem administrátor. Tyto případy použití lze nalézt na obrázcích 2 a 4. Díky generalizaci aktérů má administrátor možnost spouštět také případy použití popsané v části 4.2.3.

Spravovat konfiguraci

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s konfigurací procesu zpracování zpráv. Jedná se o akce vytvoření, zobrazení, editování a smazání konfigurace.

Otestovat konfiguraci

Aktér zvolí konfiguraci procesu zpracování zpráv, kterou si přeje otestovat. Dále aktér vybere zprávy, na kterých bude konfigurace testována. Centrála poté provede automatické zpracování vybraných zpráv a zobrazí aktérovi výsledky, přičemž informace lišící se od výsledků původních zpracování testovacích zpráv budou zvýrazněny. Na základě výsledků testování konfigurace může aktér označit konfiguraci za aktuální konfiguraci používanou centrálou při zpracování příchozích zpráv. Konfigurace, která nikdy nebyla testována, nemůže být označena za aktuálně používanou.

Spravovat typy zpráv

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s typy zpráv. Jedná se o akce vytvoření, zobrazení, editování a smazání typu zprávy.

Spravovat typy významných souborů

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s typy významných souborů. Jedná se o akce vytvoření, zobrazení, editování a smazání typu významného souboru.

Spravovat typy analytických položek

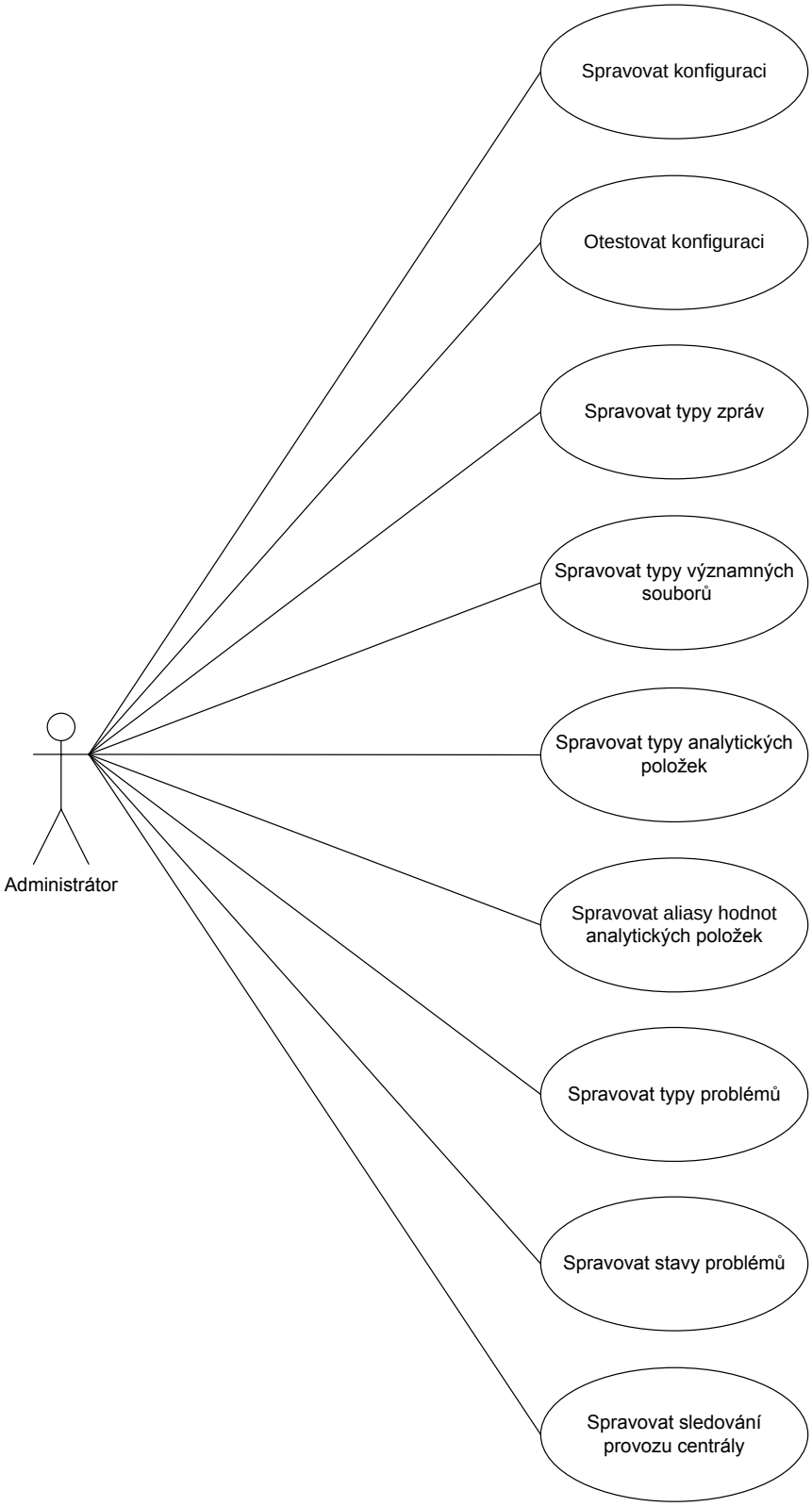
Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s typy analytických položek. Jedná se o akce vytvoření, zobrazení, editování a smazání typu analytické položky.

Spravovat aliasy hodnot analytických položek

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s aliasy hodnot analytických položek. Jedná se o akce vytvoření, zobrazení, editování a smazání aliasu hodnoty analytické položky.

Spravovat typy problémů

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci s typy problémů. Jedná se o akce vytvoření, zobrazení, editování a smazání typu problému.



Obrázek 4: Diagram případů použití balíčku Spravovat nastavení centrály

Spravovat stavy problémů

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro práci se stavy problémů. Jedná se o akce vytvoření, zobrazení, editování a smazání stavu problému.

Spravovat sledování provozu centrály

Tento případ použití zahrnuje všechny akce, které centrála poskytuje pro sledování provozu centrály. Jedná se o akce zobrazení systémových log souborů, zobrazení uživatelských akcí a povolení či zakázání protokolování jednotlivých uživatelských akcí.

4.3 Nefunkční požadavky

Nefunkční požadavky představují omezení kladená na centrálu systému pro analýzu provozních chyb. Specifikace nefunkčních požadavků vychází z neformálního popisu požadavků uvedeného v části 4.1. Všechny nefunkční požadavky přehledně shrnuje tabulka 2.

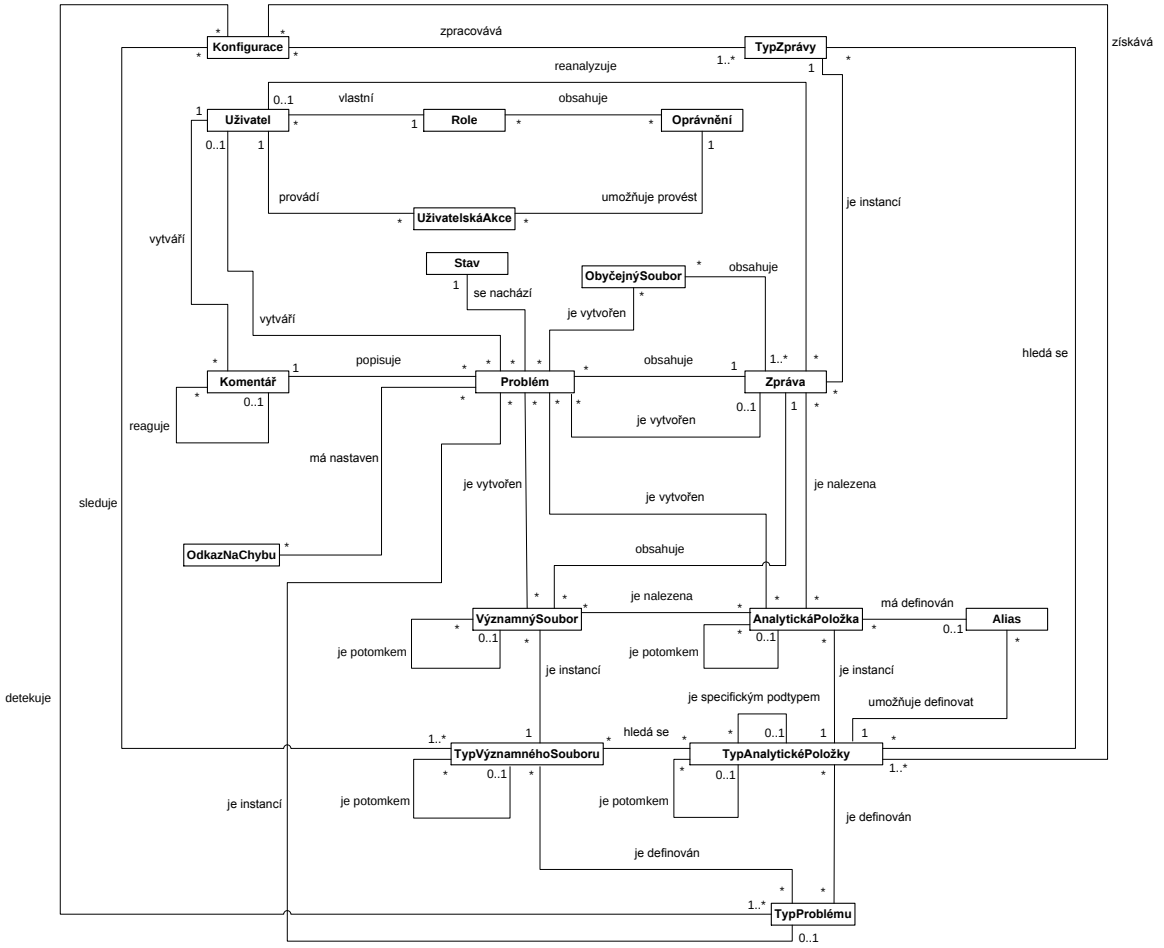
Tabulka 2: Nefunkční požadavky na centrálu systému pro analýzu provozních chyb

Požadavek	Specifikace
Nezávislost na použitém monitoru a přenosovém kanálu	Centrála musí být schopna spolupráce s libovolnou implementací monitoru a přenosového kanálu. Společně s monitorem a přenosovým kanálem musí vytvořit kompletní systém pro analýzu provozních chyb.
Zpracování zprávy libovolného formátu	Zprávy mají vždy podobu jednoho souboru, obvykle v některém používaném kompresním formátu (ZIP, RAR aj.). Pomocí modifikace konfigurace procesu zpracování zpráv musí být možné jednoduše docílit zpracování zpráv v jiném než doposud používaném formátu.
Zpracování velkého množství příchozích zpráv	Centrála musí být schopna zajistit zpracování velkého množství příchozích zpráv současně při zachování akceptovatelné doby zpracování (tj. doby od příjmu zprávy po dokončení jejího zpracování).
Jednoduchá modifikovatelnost procesu zpracování zpráv	Proces zpracování zpráv by měl být modifikovatelný tak, aby jeho změna nevyžadovala zásah do kódu centrály.
Typ aplikace centrály	GUI centrály by mělo být implementováno jako webová aplikace. Části centrály zajišťující automatické zpracování příchozích zpráv by měli být implementovány jako samostatné aplikace běžící odděleně od webové aplikace.
Nasazení centrály v interní síti společnosti	Jelikož bude databáze centrály obsahovat důvěrné informace koncových uživatelů sledovaných produktů, musí být centrála nasazena pouze v interní síti společnosti. Centrála nesmí být přímo přístupná z vnějšího prostředí.
Propojení se systémem pro sledování komunikace s koncovými uživateli	Zprávy v centrále by měly být jednoduchým způsobem (např. pomocí shodného identifikátoru zpráv) propojeny se zprávami v systému pro sledování komunikace s koncovými uživateli.
Absence okamžité zpětné vazby na uživatele produktu	Protože centrála musí být nezávislá na použitém monitoru a přenosovém kanálu, nebude do ní možné implementovat okamžitou zpětnou vazbu na uživatele produktu. Zpětnou vazbu budou muset zajistit uživatelé centrály standardními prostředky elektronické pošty (tj. centrála nebude poskytovat žádné speciální rozhraní pro zpětnou vazbu na uživatele produktu).

4.4 Konceptuální třídy

Na základě požadavků specifikovaných v podkapitolách 4.2 a 4.3 budou nyní definovány konceptuální třídy objektů, se kterými bude centrála pracovat. Konceptuální diagram těchto tříd je uveden

na obrázku 5 a popis tříd se nachází v tabulce 3. Třídy zde uvedeného modelu neobsahují atributy, jedná se pouze o prvotní náhled na třídy získané analýzou specifikovaných požadavků. Zde uvedený model konceptuálních tříd bude dále zpřesňován ve fázi návrhu (viz kapitola 5).



Obrázek 5: Diagram konceptuálních tříd

Tabulka 3: Popis konceptuálních tříd

Třída	Popis
Konfigurace	Reprezentuje konfiguraci procesu zpracování zpráv. Konfigurace zpracovává vždy nejméně jeden typ zpráv, může ale být nastavena tak, aby byla schopna zpracovat několik různých typů zpráv (vztah *zpracovává* se třídou TypZprávy). Konfigurace sleduje ve zprávách přítomnost významných souborů (vztah *sleduje* se třídou TypVýznamnéhoSouboru), z těchto souborů získává analytické položky (vztah *získává* se třídou TypAnalytickéPoložky) a detekuje ve zprávách typy problémů (vztah *detekuje* se třídou TypProblému).
TypZprávy	Reprezentuje typ zprávy, respektive formát jejího souboru (např. ZIP, RAR), který je centrála schopna zpracovat.
Zpráva	Reprezentuje příchozí zprávu. Zpráva je vždy nějakého definovaného typu (vztah *je instancí* se třídou TypZprávy) a obsahuje významné a obyčejné soubory (vztah *obsahuje* se třídami VýznamnýSoubor a ObyčejnýSoubor).
TypVýznamnéhoSouboru	Reprezentuje typ významného souboru, jehož přítomnost ve zprávě je nutné sledovat. Typ významného souboru může být potomkem jiného typu významného souboru (vztah *je potomkem* se třídou TypVýznamnéhoSouboru).

Třída	Popis
VýznamnýSoubor	Reprezentuje významný soubor zprávy. Významný soubor je vždy nějakého definovaného typu (vztah *je instancí* se třídou TypVýznamnéhoSouboru). Významný soubor jistého typu musí být potomkem významného souboru takového typu, který je rodičem typu tohoto významného souboru (vztah *je potomkem* se třídou VýznamnýSoubor). Pokud typ významného souboru není potomkem žádného jiného typu, pak ani významný soubor nemůže být potomkem žádného jiného významného souboru.
TypAnalytickéPoložky	Reprezentuje typ analytické položky, jejíž přítomnost je sledována ve významných souborech zprávy (vztah *hledá se* se třídou TypVýznamnéhoSouboru), nebo přímo v souboru zprávy jistého typu (vztah *hledá se* se třídou TypZprávy). Typ analytické položky může být potomkem jiného typu analytické položky (vztah *je potomkem* se třídou TypAnalytickéPoložky). Typ analytické položky může umožnit definování aliasu pro hodnoty analytických položek svého typu (vztah *umožňuje definovat* se třídou Alias). Typ analytické položky může být specifickým podtypem jiného typu analytické položky (vztah *je specifickým podtypem* se třídou TypAnalytickéPoložky, významné pro detekci problémů, viz část 5.3.4). Nejčastěji jsou specifickými podtypy přímo konkrétní hodnoty analytických položek obecnějšího nadtypu, např. typ analytické položky Operační systém (jeho hodnotami jsou názvy operačních systémů) má své specifické podtypy Windows XP a Windows Vista. Dále musí platit, že nadtyp společně se všemi jeho specifičtějšími podtypy musí být v asociaci *hledá se* se stejným typem významného souboru.
AnalytickáPoložka	Reprezentuje analytickou položku, která je nalezena v některém z významných souborů zprávy (vztah *je nalezena* se třídou VýznamnýSoubor), nebo přímo v souboru zprávy (vztah *je nalezena* se třídou Zpráva). Analytická položka je vždy nějakého definovaného typu (vztah *je instancí* se třídou TypAnalytickéPoložky). Analytická položka jistého typu musí být potomkem analytické položky takového typu, který je rodičem typu této analytické položky (vztah *je potomkem* se třídou AnalytickáPoložka). Pokud typ analytické položky není potomkem žádného jiného typu, pak ani analytická položka nemůže být potomkem žádné jiné analytické položky. Analytická položka může mít pro svou hodnotu definován alias (vztah *má definován* se třídou Alias).
Alias	Reprezentuje alias pro různé hodnoty analytických položek, které mají stejný význam.
TypProblému	Reprezentuje typ problému definovaný typy analytických položek a významných souborů (vztah *je definován* se třídami TypAnalytickéPoložky a TypVýznamnéhoSouboru). Pokud se v definici typu problému nachází více typů analytických položek, které mají hodnotu, pak se musí všechny tyto typy analytických položek hledat ve stejném typu významného souboru (omezení nutné pro detekci problémů, viz část 5.3.4).
ObyčejnýSoubor	Reprezentuje obyčejný soubor zprávy. Obyčejný soubor může být obsažen ve více zprávách (nejsou v něm hledány žádné analytické položky, a proto není vyžadována unikátnost souborů se stejným názvem pro každou zprávu).
Problém	Reprezentuje problém obsažený ve zprávě (vztah *obsahuje* se třídou Zpráva), který je vytvořený na základě analytických položek, významných souborů, obyčejných souborů nebo vlastního souboru zprávy (vztah *je vytvořen* se třídami AnalytickáPoložka, VýznamnýSoubor, ObyčejnýSoubor a Zpráva). Automaticky vytvořený problém je vždy nějakého definovaného typu (vztah *je instancí* se třídou TypProblému). Problém se vždy nachází v nějakém stavu (vztah *nachází se* se třídou Stav) a může mít nastaven odkaz na chybu v systému pro sledování chyb (vztah *má nastaven* se třídou OdkazNaChybu).
Stav	Reprezentuje příznak označující stav problému.
OdkazNaChybu	Reprezentuje odkaz na chybu v systému pro sledování chyb.
Komentář	Reprezentuje komentář blíže popisující problém (vztah *popisuje* se třídou Problém). Komentář může reagovat na jiný již existující komentář problému (vztah *reaguje* se třídou Komentář).
Uživatel	Reprezentuje uživatele centrály. Uživatel vlastní právě jednu roli (vztah *vlastní* se třídou Role). Dále uživatel vytváří problémy a komentáře problémů (vztah *vytváří* se třídami Problém a Komentář), reanalyzuje zprávy (vztah *reanalyzuje* se třídou Zpráva) a provádí akce (vztah *provádí* se třídou UživatelskáAkce).
Role	Reprezentuje roli uživatele centrály. Role obsahuje vždy nejméně jedno oprávnění (vztah *obsahuje* se třídou Oprávnění).
Oprávnění	Reprezentuje oprávnění, které umožňuje uživateli centrály provést konkrétní akci (vztah *umožňuje provést* se třídou UživatelskáAkce).
UživatelskáAkce	Reprezentuje zaprotokolovanou akci provedenou uživatelem centrály.

5 Návrh

Tato kapitola se zabývá návrhem centrály systému pro analýzu provozních chyb. Nejprve bude rozšířen konceptuální model z části 4.4. Dále bude v kapitole představen návrh fyzického schématu databáze a detailně bude rozpracován návrh zpracování zpráv. V závislosti na zvolených vývojových nástrojích a technologiích pak bude diskutován návrh architektury systému a návrhové třídy.

5.1 Rozšíření konceptuálního modelu

Při tvorbě konceptuálního modelu v části 4.4 se kladl důraz na to, aby třídy modelu pokrývaly všechny funkční a nefunkční požadavky. Nyní bude tento model rozšířen tak, aby použitím tříd v něm obsažených šlo požadované chování centrály skutečně realizovat.

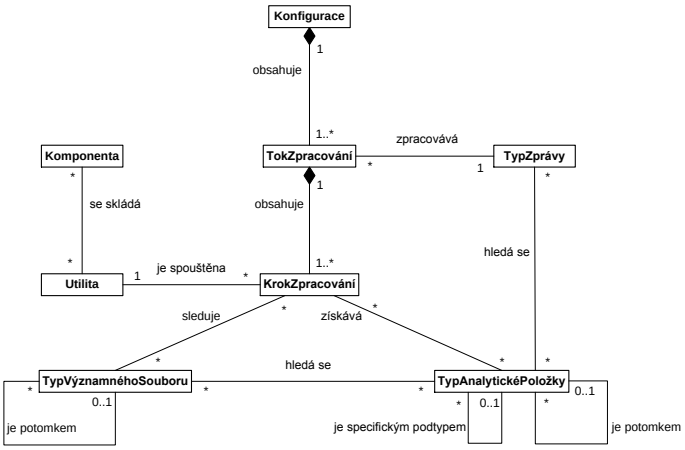
5.1.1 Rozšíření třídy Konfigurace

Jedním z nefunkčních požadavků (viz část 4.3) je požadavek na jednoduchou modifikovatelnost procesu zpracování zpráv. Třídou, která reprezentuje proces zpracování zpráv, je *Konfigurace*. Obsahuje vazby na další třídy, které jsou pro proces zpracování zpráv významné (TypVýznamnéhoSouboru, TypAnalytickéPoložky aj.), ale v podobě, v jaké byla představena v části 4.4, tato třída neříká nic o vlastním provedení tohoto procesu. V této podkapitole budou představeny třídy, které rozšíří třídu Konfigurace tak, aby proces zpracování zpráv vyhověl požadavku na svou jednoduchou modifikovatelnost.

Třída *TokZpracování* reprezentuje postup, pomocí něhož jsou zpracovávány zprávy určitého typu. Roli, kterou měla třída Konfigurace v asociaci *zpracovává* se třídou TypZprávy, nyní přebírá třída TokZpracování.

Každý tok zpracování se skládá z jednoho či více kroků, jež jsou v rozšířeném modelu třídy Konfigurace reprezentovány třídou *KrokZpracování*. Krok zpracování spouští právě jednu utilitu (reprezentována třídou *Utilita*), která ke svému běhu může vyžadovat přítomnost některých dalších souborů či knihoven (tyto části utility jsou reprezentovány třídou *Komponenta*). Utility budou nejčastěji sledovat přítomnost významných souborů ve zprávách a získávat z nich analytické položky (vztahy *sleduje* a *získává* jsou přeneseny z třídy Konfigurace na třídu KrokZpracování). Prvním krokem zpracování v každém toku bude nejčastěji převedení souboru zprávy (např. rozbalení archivu) do podoby, s níž budou moci další kroky pracovat. Pomocí třídy KrokZpracování bude zajištěno splnění požadavku na zpracování zpráv libovolného formátu. Každá doplňková analýza bude realizována pomocí nějaké utility jednoho z kroků zpracování.

Modifikovat proces analýzy zpráv bude znamenat přidání, změnu, či smazání kroku zpracování nějakého toku zpracování uvnitř konfigurace. Jelikož je chování kroků zpracování realizováno pomocí externích utilit, nebude muset být měněn kód centrály, pouze kód utilit. Požadavek na modifikovatelnost procesu zpracování zpráv bude tedy splněn. Způsob, jakým bude v centrále zajištěno samotné zpracování zpráv (tj. spouštění kroků zpracování), bude diskutován v částech 5.1.2 a 5.3. Třídy představené v této podkapitole a jejich vztahy se třídami původního konceptuálního modelu jsou znázorněny na obrázku 6.



Obrázek 6: Rozšíření konceptuální třídy Konfigurace

5.1.2 Třídy zajišťující zpracování zpráv

Základní jednotku zpracování zpráv představuje třída *Služba* reprezentující část centrály nasazenou na stroji dedikovaném výhradně pro zpracování zpráv. V rámci centrály bude moci být spuštěno více služeb zároveň. Zvýšením počtu běžících služeb se zajistí zvýšení počtu zpráv, které budou moci být v daném okamžiku zpracovány (splnění nefunkčního požadavku na zpracování velkého množství příchozích zpráv). Hlavním úkolem služby je přijímat příchozí zprávy a analyzovat je, tj. spustit pro každou zprávu tok zpracování (viz část 5.1.1) odpovídající jejímu typu.

Soubory zpráv daného typu přicházejí do centrály na některé z definovaných vstupních úložišť (reprezentována jsou třídou *VstupníÚložiště*). Z těchto úložišť jsou soubory odebírány ke zpracování jednotlivými službami. Určit úložiště, ze kterých bude služba soubory zpráv odebírat, lze prostřednictvím typu služby (reprezentuje jej třída *TypSlužby*). Nemá-li služba nastaven typ, je spuštěna, ale neodebírá soubory ze žádného úložiště a čeká, dokud jí typ nebude přidělen.

Fakt, že má služba nastaven typ a tedy má obsluhovat nějaká úložiště, ještě neznamená, že služba bude soubory zpráv z těchto úložišť skutečně odebírat. Služba vždy obsluhuje pouze ta úložiště, na která přicházejí soubory zpráv typů, pro něž existuje v aktuální konfiguraci tok zpracování.

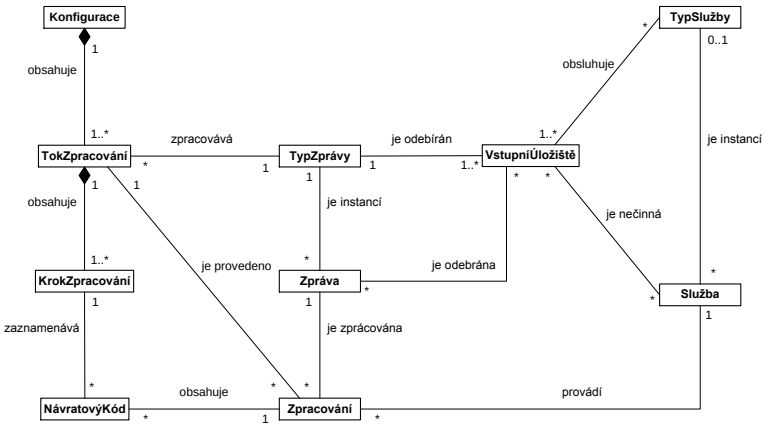
Odebírání příchozích zpráv ze vstupních úložišť splňuje požadavek nezávislosti centrály na monitoru a přenosovém kanálu. K zajištění funkčnosti plnohodnotného systému pro analýzu provozních chyb je ale nutné, aby k úložištím měly přístup všechny části systému (v závislosti na implementaci monitoru a přenosového kanálu je z tohoto pravidla možné některou z těchto dvou částí vyjmout, vždy však musí být zaručeno doručování zpráv na úložiště podle jejich příslušnosti k typům zpráv definovaných centrálou).

Zvláštním typem služby, která se nestará přímo o zpracování zpráv, je tzv. *hashovací služba*. Jejím úkolem je ke každé příchozí zprávě spočítat hash jejího souboru a do databáze centrály uložit záznam o výskytu zprávy na konkrétním vstupním úložišti. Hashovací služba zajišťuje, že se nebudou nikdy zpracovávat dva stejné soubory. Soubory, jejichž hash se již v databázi centrály vyskytuje, hashovací služba ze vstupního úložiště odmazává. Hashovací služba běží nad jedním či více vstupními úložišti, která jsou podobně jako u běžné služby definována jejím typem.

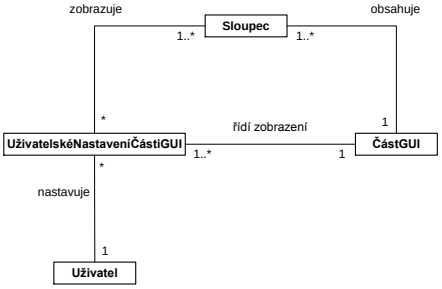
Průběh zpracování zpráv je nutné sledovat (nutné např. pro testování konfigurace, kdy je třeba uživateli zprostředkovat nejen výsledky testovací analýzy, ale zejména informace o možných

selháních některých kroků zpracování). Protokol o průběhu zpracování zprávy reprezentuje třída *Zpracování*. Tento protokol obsahuje informace o službě, která zprávu zpracovala, a návratové kódy (reprezentuje je třída *NávratovýKód*) získané provedením kroků zpracování (tj. spuštěním externích utilit).

Na obrázku 7 je uvedena část diagramu konceptuálních tříd rozšířeného o třídy představené v této podkapitole.



Obrázek 7: Třídy zajišťující zpracování zpráv

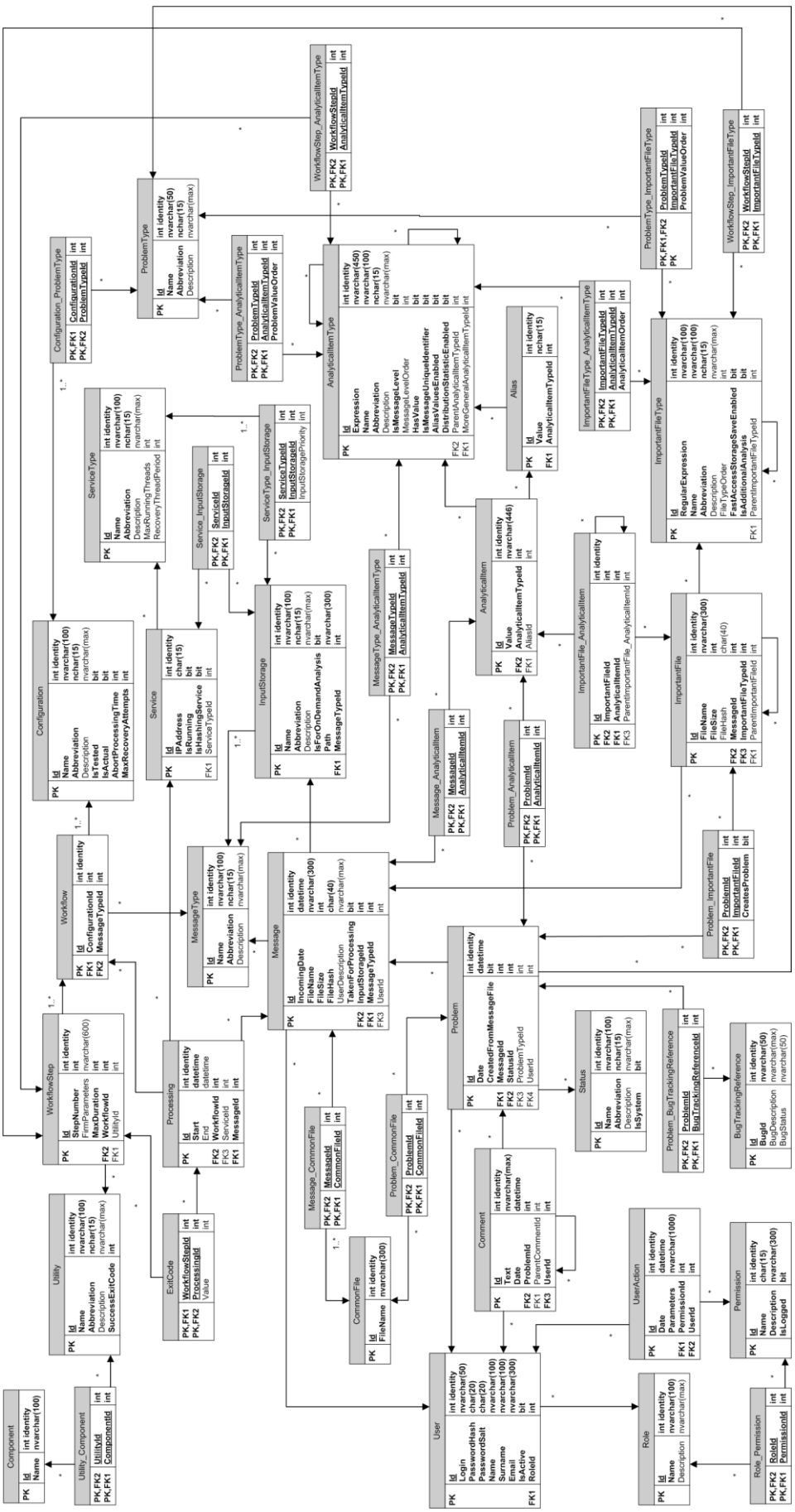


Obrázek 8: Třídy zajišťující uživatelskou konfigurovatelnost GUI

5.1.3 Třídy zajišťující uživatelskou konfigurovatelnost GUI

V případech použití Zobrazit zprávy a Zobrazit problém bylo uvedeno, že si aktér může v GUI nechat zobrazit pouze některé vlastnosti zpráv nebo problémů. Taková vlastnost GUI je označována jako jeho konfigurovatelnost.

Základní třídou zajišťující konfigurovatelnost je třída *ČástGUI* reprezentující konfigurovatelnou část GUI. Část GUI poskytuje k zobrazení sadu vlastností (sloupců) reprezentovaných třídou *Sloupec*. Z těchto vlastností si uživatel může sestavit vlastní nastavení části GUI (reprezentované třídou *UživatelskéNastaveníČástiGUI*), které si bude moci uložit do databáze a při dalším zobrazení konfigurovatelné části GUI jej bude moci znovu použít. Všechny výše uvedené třídy jsou znázorněny na obrázku 8.



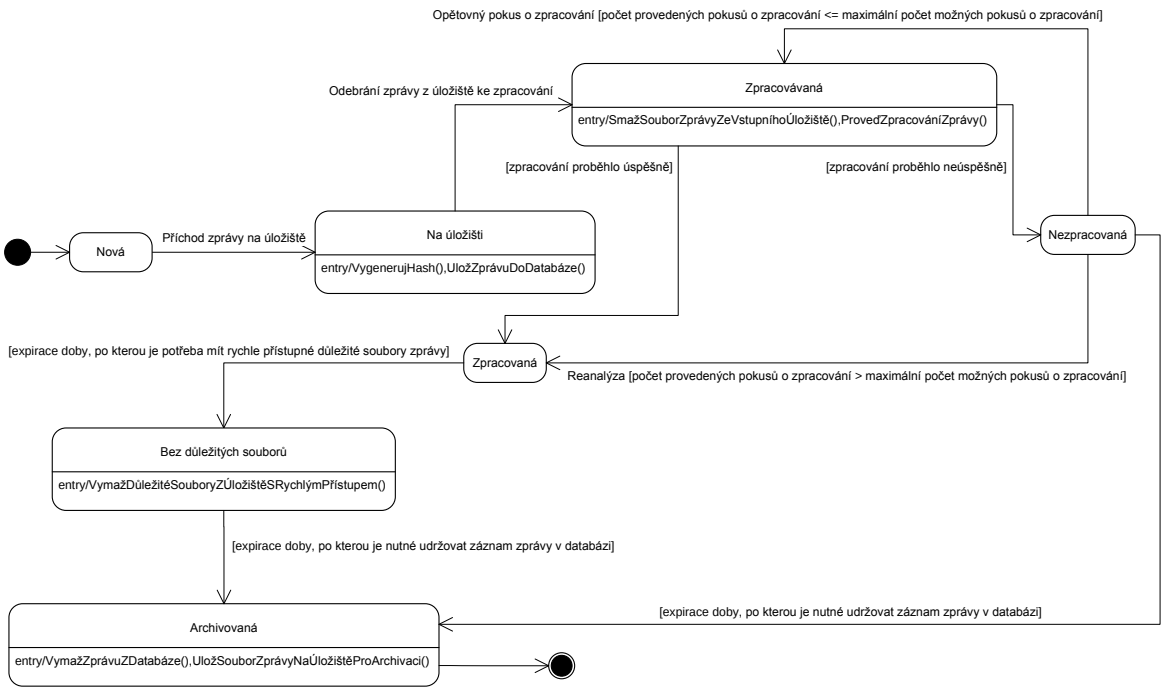
Obrázek 9: Diagram fyzického schématu databáze

5.2 Návrh fyzického schématu databáze

V přecházejících částech této kapitoly byl konceptuální model tříd uvedený v části 4.4 rozšířen o třídy zajišťující některé specifické funkčnosti centrály. Použitím tříd konceptuálního modelu tak bude možné plně realizovat požadované chování centrály. Tato podkapitola představuje výsledný návrh fyzického schématu databáze získaný transformací konceptuálního modelu doplněného o atributy tříd. Identifikace všech potřebných atributů tříd je proces náročný, některé atributy byly identifikovány až při implementaci centrály. Fyzické schéma databáze tak procházelo během realizace centrály mnoha změnami. Diagram fyzického schématu databáze centrály na obrázku 9 znázorňuje schéma databáze v současné době implementované verzi centrály (viz kapitola 6). V této verzi zcela chybí realizace tříd, které zajišťují uživatelskou konfigurovatelnost GUI. Datové typy v diagramu již reflektují prostředí zvoleného databázového serveru (viz část 5.4). V diagramu jsou tučně vyznačeny atributy, které nepovolují prázdné hodnoty. Podrobný popis fyzického schématu databáze je uveden v příloze A.

5.3 Zpracování zpráv

Následující podkapitola se bude detailně zabývat problematikou zpracování zpráv. Představen zde bude stavový diagram třídy Zpráva, na kterém bude ukázán životní cyklus zprávy od jejího přijetí centrálou až po její vyřazení z databáze centrály. Další text podkapitoly se bude věnovat návrhu způsobu, jakým budou zprávy zpracovávány službou pro zpracování zpráv (viz část 5.1.2). Předposlední část této podkapitoly shrne požadavky a omezení, jež musí splňovat externí utilita, aby ji mohlo být možné jednoduše začlenit do toku zpracování (viz část 5.1.1). Závěr podkapitoly vysvětlí postup, jakým služba pro zpracování zpráv detekuje a vytváří problémy.



Obrázek 10: Stavový diagram třídy Zpráva

5.3.1 Stavový diagram třídy Zpráva

Stavový diagram třídy Zpráva je zachycen na obrázku 10.

První kontakt nové zprávy (stav *Nová*) s centrálou probíhá na vstupním úložišti, kde je pro soubor zprávy vygenerován unikátní hash a záznam o zprávě (společně s informací, na kterém ze vstupních úložišť se zpráva nachází) je vložen do databáze centrály (stav *Na úložišti*). Následně je zpráva z úložiště odebrána některou ze služeb pro zpracování zpráv, které dané úložiště obsluhují. Soubor zprávy je poté službou pro zpracování zpráv z úložiště odmazán, jeho kopii si služba zkopíruje do lokálního adresáře na stroji služby a v tomto adresáři bude poté provedeno zpracování zprávy příslušným tokem zpracování. Zpráva se po tuto dobu nachází ve stavu *Zpracovávaná*.

V závislosti na úspěšnosti zpracování se pak zpráva dostane do jednoho ze dvou stavů *Zpracovaná* nebo *Nezpracovaná*, přičemž nezpracované zprávy jsou službami pro zpracování v pravidelných intervalech odebírány z databáze a opětovně zpracovávány, dokud jejich zpracování neproběhne úspěšně, nebo se nedosáhne limitu stanovujícího maximální počet pokusů o opětovné zpracování nezpracovaných zpráv (tento limit lze nastavit pro každou konfiguraci procesu zpracování odlišně, služby ale vždy pracují s limitem stanoveným aktuální konfigurací). Zprávy, které dosáhly limitu stanovujícího maximální počet pokusů o opětovné zpracování, zůstávají nezpracovány, dokud nebudou reanalyzovány manuálně nebo nevyprší doba, po kterou je nutné udržovat záznam zprávy v databázi.

O zpracované zprávě jsou v centrále dostupné všechny informace (provedené analýzy, nalezené problémy, významné soubory ke stažení). Rychlá dostupnost významných souborů zprávy je potřebná pouze během několika prvních týdnů existence zprávy v databázi centrály. Proto zpráva po čase přechází do stavu *Bez důležitých souborů*, kdy jsou významné soubory zprávy z úložiště s rychlým přístupem odmazány.

Doba života zprávy v centrále končí přenesením souboru zprávy na úložiště pro archivaci (např. páska) a jejím vymazáním z databáze (stav *Archivovaná*).

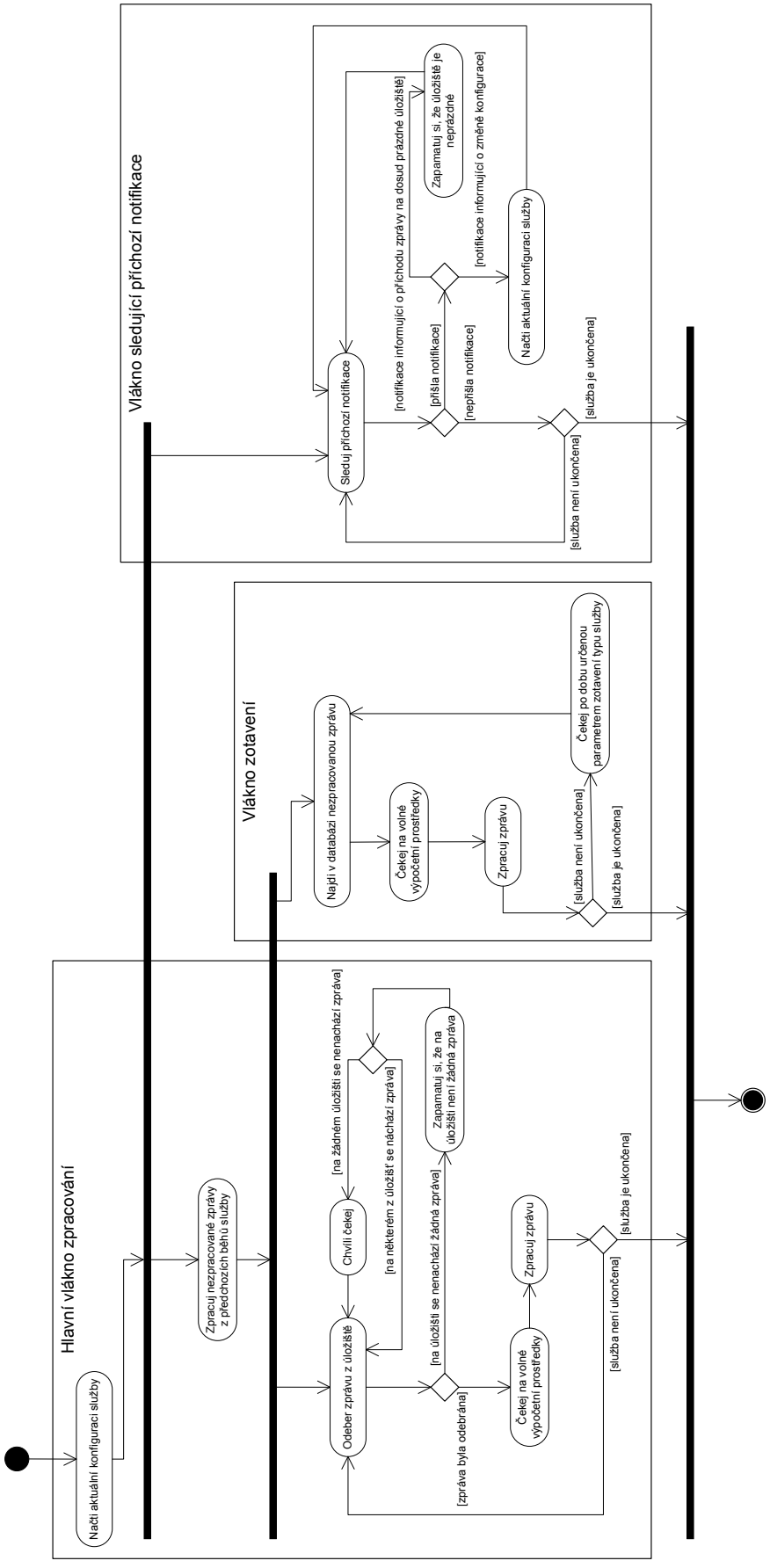
5.3.2 Služba pro zpracování zpráv

Jak bylo uvedeno v části 5.1.2, o zpracování příchozích zpráv, tj. o provedení příslušného toku zpracování, se stará služba pro zpracování zpráv běžící na stroji vyhrazeném pro zpracování zpráv. Těchto služeb může běžet v centrále několik, přičemž každá může zpracovávat zprávy z jiného vstupního úložiště. Běh služby je pomocí diagramu aktivit znázorněn na obrázku 11. Z diagramu je patrné, že služba realizuje své chování pomocí tří vláken, jejichž činnost bude v následujících odstavcích podrobně popsána.

Hlavní vlákno zpracování

Po svém spuštění si služba z databáze centrály načte svou aktuální konfiguraci, tj. aktuální konfiguraci procesu zpracování zpráv a typ služby, který určí vstupní úložiště obsluhovaná službou a další parametry služby. Než bude služba pokračovat dalším krokem, spustí vlákno pro sledování příchozích notifikací.

Následně jsou v databázi centrály identifikovány zprávy, které služba při svých minulých bězích nezpracovala. Tyto zprávy jsou poté službou postupně zpracovávány (pouze ale, pokud ještě nedosáhly limitu stanovujícího maximální možný počet pokusů o opětovné zpracování). Po



Obrázek 11: Diagram aktivit znázorňující běh služby pro zpracování zpráv

dokončení zpracování těchto zpráv služba spustí vlákno zotavení a pokračuje ve zpracování zpráv, nyní již zpráv nově příchozích.

Zprávy jsou službou postupně odebírány ze vstupních úložišť, které služba obsluhuje. Služba si je schopna pamatovat, že na některém ze vstupních úložišť nejsou žádné zprávy. Pokud služba prošla všechna vstupní úložiště, a ze žádného se jí nepodařilo odebrat zprávu ke zpracování, služba chvíli čeká (dotazy pro odběr zpráv z úložišť by v takovém případě pouze zbytečně zatěžovaly databázový server), než se opět pokusí odebrat zprávu ze vstupního úložiště.

Služba by měla výkonu stroje, na němž běží, maximálně využít. Proto je navržena tak, aby zpracování každé zprávy spouštěla ve zvláštním vlákně. Počet těchto vláken lze omezit jedním z parametrů typu služby. Službu lze tedy optimalizovat tak, aby využila výkonu jak jednoprocesorových strojů, tak i strojů s vícejádrovými procesory.

Počet současně spuštěných vláken zpracování zpráv nesmí překročit maximální počet stanovený typem služby – tento fakt je v diagramu aktivit vyjádřen aktivitou *Čekej na volné výpočetní prostředky*, kdy po odběru zprávy ze vstupního úložiště musí služba počkat, dokud počet spuštěných vláken zpracování nebude menší než maximálně stanovený počet. Teprve potom může spustit aktivitou *Zpracuj zprávu* nové vlákno zpracování. Za vlákno zpracování je považováno jakékoliv vlákno zpracování běžící v rámci služby, tj. do celkového počtu vláken zpracování se počítají vlákna zpracování spuštěná jak hlavním vláknem zpracování, tak vláknem zotavení.

Vlákno pro sledování příchozích notifikací

Toto vlákno má za úkol sledovat notifikace, které mohou být službě zaslány databázovým serverem, na němž běží databáze centrály. Tyto notifikace mají službu upozornit na změnu v její konfiguraci (např. změna aktuální konfigurace procesu zpracování zpráv nebo změna některého z parametrů typu služby) a příchod nové zprávy na dosud prázdné úložiště. V obou případech služba musí příslušně zareagovat, tedy buď načíst z databáze aktuální verzi konfigurace, nebo pro příslušné úložiště nastavit příznak jeho neprázdnosti.

Vlákno zotavení

Vlákno má za úkol v pravidelném intervalu zotavení (určen jedním z parametrů typu služby) získávat z databáze informace o nezpracovaných zprávách. Případné nezpracované zprávy se toto vlákno pokouší zpracovat, během jednoho intervalu zotavení vlákno provede zpracování nejstarší zprávy z detekovaných nezpracovaných zpráv.

5.3.3 Požadavky na externí utility toku zpracování

Vlastní zpracování zprávy nezajišťuje přímo služba pro zpracování zpráv, jejímuž návrhu byla věnována předchozí podkapitola, ale tok zpracování, který tato služba pro každou zprávu spouští. Kroky toku zpracování jsou externí utility, a aby bylo možné do toku zpracování jednoduše přidat novou utilitu, musí utility splňovat dva základní požadavky, které budou podrobně vysvětleny v následujícím textu.

Rozhraní utilit

Utility musí být realizovány jako programy spustitelné v prostředí příkazové řádky. Při spuštění utility jí předává služba parametry příkazové řádky v pořadí odpovídajícím výskytu parametrů v následujícím seznamu:

- `-p <pevné parametry>`
  - definuje-li krok zpracování pro utilitu nějaké pevné parametry, jsou jí tyto předány prostřednictvím tohoto parametru
- `-d <pracovní adresář>`
  - tímto parametrem je utilitě předána cesta k adresáři, ve kterém probíhá zpracování zprávy
- `-f <id typu souboru> <id typu souboru otce> <regulární výraz typu souboru>`
  - tento parametr je předán, je-li s krokem zpracování spouštějícím utilitu asociován nějaký typ významného souboru
  - za přepínačem tohoto parametru jsou předány tři uvedené vlastnosti každého typu významného souboru, který je s krokem zpracování spouštějícím utilitu asociován
  - pokud typ významného souboru nemá žádný nadřazený typ významného souboru, je vlastnost `<id typu souboru otce>` rovna nule
- `-i <id typu souboru> <id typu souboru otce> <regulární výraz typu souboru> <id typu analytické položky> <id typu analytické položky otce> <typ analytické položky má hodnotu> <výraz typu analytické položky>`
  - tento parametr je předán, je-li s krokem zpracování spouštějícím utilitu asociován nějaký typ analytické položky
  - tento parametr je předán pro každý typ souboru, ve kterém se hledá některý z typů analytických položek asociovaných s krokem zpracování spouštějícím utilitu
  - za přepínačem tohoto parametru se nacházejí tři uvedené vlastnosti typu významného souboru a za nimi pak čtyři uvedené vlastnosti pro každý typ analytické položky, který se hledá v tomto typu významného souboru
  - pokud typ analytické položky má hodnotu, je hodnota `<typ analytické položky má hodnotu>` rovna jedné, jinak nule
  - pokud typ významného souboru nemá žádný nadřazený typ významného souboru, je hodnota `<id typu souboru otce>` rovna nule
  - pokud typ analytické položky nemá žádný nadřazený typ analytické položky, je hodnota `<id typu analytické položky otce>` rovna nule

Formát výstupního XML souboru toku zpracování

Utility vykonávají nejrůznější činnosti, od rozbalení archivu až po detekci významných souborů a analytických položek. Informace o detekovaných významných souborech a v nich nalezených ana-

lytických položkách musí být uloženy do databáze. Utility by neměly pracovat s databází centrály přímo, výsledek zpracování by do ní měl být vložen jedinou transakcí s co možná nejkratší dobou trvání. Za tímto účelem centrála ještě před spuštěním toku zpracování vytváří ve svém lokálním adresáři se souborem zprávy výstupní XML soubor následujícího tvaru:

```
<?xml version="1.0" encoding="utf-8" ?>
<Output>
  <UserDescription/>
</Output>
```

Do tohoto souboru pak musí utility během toku zpracování zapsat informace o všech významných souborech detekovaných ve zprávě a analytických položkách, které byly ve významných souborech nalezeny. Nezáleží na tom, jak bude zápis do tohoto souboru realizován a jestli bude do souboru zapisovat pouze jedna utilita, nebo do něj bude postupně zapisovat více utilit. Po skončení toku zpracování si služba tento XML soubor načte, detekuje problémy (viz část 5.3.4) a provede uložení získaných informací do databáze pomocí jedné transakce.

Příklad výstupního XML souboru lze nalézt v příloze B, jeho formát popisují následující pravidla DTD (Document Type Definition, viz např. [11]):

```
<!ELEMENT Output (UserDescription, ImportantFile*)>
<!ELEMENT UserDescription (#PCDATA)>
<!ELEMENT ImportantFile (Name, Size, Hash, TypeId, AnalyticalItem*, ImportantFile*)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Size (#PCDATA)>
<!ELEMENT Hash (#PCDATA)>
<!ELEMENT TypeId (#PCDATA)>
<!ELEMENT AnalyticalItem (Value, TypeId, AnalyticalItem*)>
<!ELEMENT Value (#PCDATA)>
```

Význam jednotlivých elementů je následující:

- *Output*. Kořenový element.
- *UserDescription*. Popis problému od koncového uživatele.
- *ImportantFile*. Záznam o výskytu významného souboru ve zprávě.
- *Name*. Název významného souboru.
- *Size*. Velikost významného souboru v bajtech.
- *Hash*. Hash významného souboru spočítaný hashovací funkcí SHA-1.
- *TypeId*. Identifikátor typu významného souboru nebo analytické položky (utilita jej získá prostřednictvím parametrů příkazové řádky).
- *AnalyticalItem*. Záznam o výskytu analytické položky ve významném souboru.
- *Value*. Hodnota analytické položky (v případě, že typ analytické položky není hodnotový, pak hodnota musí být rovna výrazu typu analytické položky).

Nadřazenost významných souborů a analytických položek musí být ve výstupním XML souboru vyjádřena hierarchickým zanořováním elementů *ImportantFile*, respektive *AnalyticalItem*.

5.3.4 Detekce problémů

Poté, co služba získá data z výstupního XML souboru toku zpracování, musí ve zprávě detekovat problémy. Služba má vždy načtenou aktuální konfiguraci procesu zpracování, s níž jsou spojeny typy problémů, které lze ve zprávě detekovat. Typy problémů služba rozděluje na dvě skupiny:

- *Obecné typy problémů.* V definici typu problému nefiguruje žádný typ analytické položky, který by byl specifickým podtypem nějakého jiného typu analytické položky.
- *Specifické typy problémů.* V definici typu problému se nachází alespoň jeden typ analytické položky, který je specifickým podtypem nějakého jiného typu analytické položky.

Proces detekce problémů

Proces detekce problémů lze neformálně popsat následující sekvencí kroků:

1. Detekuj problémy specifických typů.
    1.1. Pro každý specifický typ problému, který má být detekován aktuální konfigurací procesu zpracování zpráv:
        1.1.1. Najdi ve zprávě analytické položky typů, které se nacházejí v definici typu problému.
        1.1.2. Najdi ve zprávě významné soubory typů, které se nacházejí v definici typu problému.
        1.1.3. Pokud nalezený počet analytických položek a významných souborů nepostačuje k vytvoření problému (např. není nalezena žádná analytická položka ani významný soubor nebo je nalezena pouze část analytických položek a významných souborů), ukonči detekci problémů daného typu problému a přejdi na další specifický typ problému.
        1.1.4. Jinak byl ve zprávě detekován problém. Z nalezených analytických položek a významných typů souborů vytvoř ke zprávě problém daného typu problému pro každou možnou unikátní hodnotu problému.
2. Detekuj problémy obecných typů.
    2.1. Pro každý obecný typ problému, který má být detekován aktuální konfigurací procesu zpracování zpráv:
        2.1.1. Najdi ve zprávě analytické položky typů, které se nacházejí v definici typu problému.
        2.1.2. Najdi ve zprávě významné soubory typů, které se nacházejí v definici typu problému.
        2.1.3. Pokud nalezený počet analytických položek a významných souborů nepostačuje k vytvoření problému (např. není nalezena žádná analytická položka ani významný soubor nebo je nalezena pouze část analytických položek a významných souborů), ukonči detekci problémů daného typu problému a přejdi na další obecný typ problému.

2.1.4. Jinak byl ve zprávě detekován problém. Z nalezených analytických položek a významných typů souborů vytvoř ke zprávě problém daného typu problému pro každou možnou unikátní hodnotu problému. V tomto kroku však nevytvářej problémy, pokud je typ problému definován pouze jednou analytickou položkou (dále označovaná jako analytická položka A), která má hodnotu, a zároveň jsou splněna všechna níže uvedená kritéria:

- byl již vytvořen problém na základě nějaké analytické položky B, která je specifickým podtypem analytické položky A,
- typ analytické položky A má hodnotu, typ analytické položky B hodnotu nemá,
- hodnota analytické položky A je shodná s výrazem typu analytické položky B.

Příklad detekce problémů

Aktuální konfigurací procesu zpracování zpráv jsou detekovány dva typy problémů, typ problému *pA* a typ problému *pB*. Typ problému *pA* je definován dvěma typy analytických položek *taA1* a *taA2*. Typ problému *pB* je definován dvěma typy analytických položek *taB1* a *taB2* a výskytem významného souboru *tsB*. Typy analytických položek *taA1*, *taB1*, *taB2* mají hodnotu, typ analytické položky *taA2* je specifickým podtypem typu *taB2* a nemá hodnotu (zde je relevantní uvést i výraz typu *tA2* – ten nechť je roven např. řetězci XXX). Typ problému *pA* je tedy specifickým typem problému, typ *pB* je obecným typem problému.

Při zpracování zprávy se nalezly následující analytické položky a významné soubory:

- analytická položka typu *taA1* s hodnotou AAA,
- analytická položka typu *taA1* s hodnotou BBB,
- analytická položka typu *taA2*,
- analytická položka typu *taB1* s hodnotou CCC,
- analytická položka typu *taB2* s hodnotou XXX,
- významný soubor typu *tsB* s názvem SOUBOR.

Služba se bude snažit nejprve detekovat problémy specifického typu *pA*. Pro vytvoření problému typu *pA* se ve zprávě našlo dostatek analytických položek. Dokonce budou vytvořeny dva problémy tohoto typu, protože lze vytvořit dvě unikátní hodnoty problému, „AAA XXX“ je hodnota jednoho vytvořeného problému a „BBB XXX“ je hodnota druhého problému, který bude vytvořen. Pro účely určení hodnot problému se u typu analytické položky, která nemá hodnotu, bere jako její hodnota výraz typu analytické položky (zde řetězec XXX).

Dále služba přistoupí k detekci problémů obecného typu *pB*. Pro vytvoření problému se nalezlo dostatek analytických položek a významných souborů. U obecných typů problémů je však potřeba zkontrolovat, zda nejsou splněny podmínky, které by vytvoření problému zabránily. Tyto podmínky výše uvedený případ nesplňuje a problém typu *pB* může být vytvořen (hodnota problému bude „CCC XXX SOUBOR“). Pokud by byl z definice typu problému vyňat typ analytické položky *taB1*, pak by podmínky zabraňující vytvoření obecného problému splněny byly a problém by vytvořen být nemohl (typ problému *pB* by byl definován pouze jedním typem analytické polož-

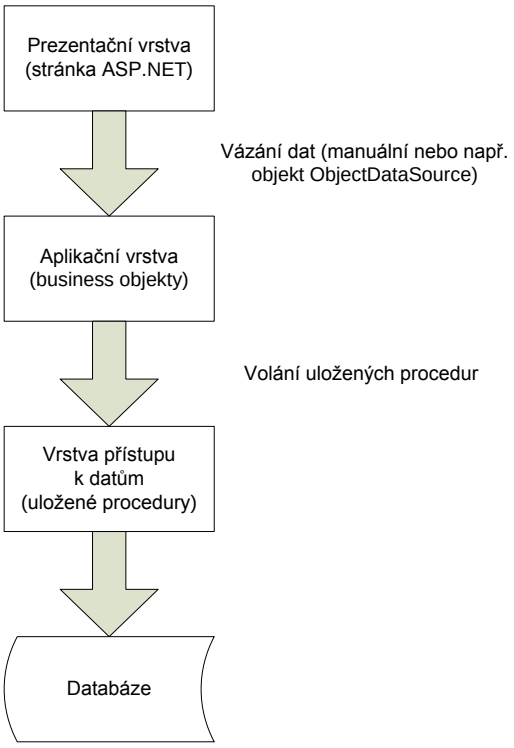
ky, který má hodnotu, typem *taB2*, a ve výše uvedeném případě již byly vytvořeny dva problémy na základě analytické položky typu *taA2*, který je specifickým podtypem typu *tB2*, nemá hodnotu, a jeho výraz je shodný s hodnotou nalezené analytické položky typu *taB2*).

5.4 Zvolená platforma a vývojové nástroje

Pro realizaci centrály je zvolena platforma Microsoft Windows. Všechny části systému centrály běží pod operačním systémem Microsoft Windows Server 2003. Pro realizaci centrály je použito běhového prostředí a knihoven Microsoft .NET Framework 3.5. Zvoleným programovacím jazykem je jazyk C# verze 3.0. K implementaci GUI webové aplikace je využito technologie ASP.NET 3.5. Databáze centrály je implementována v prostředí Microsoft SQL Server 2005.

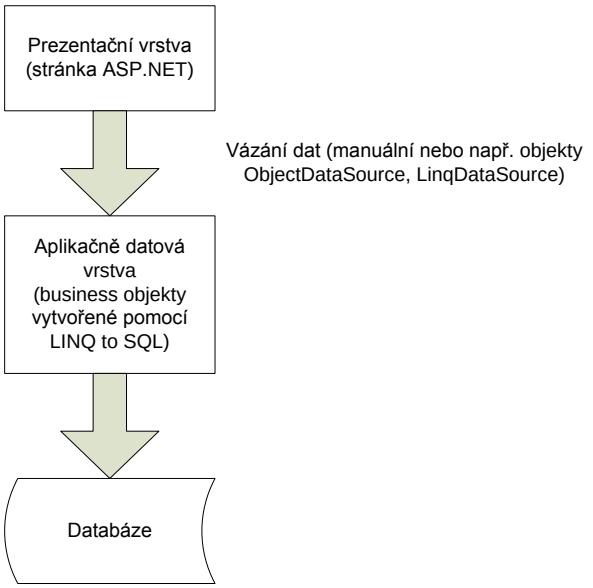
5.5 Návrh architektury

Při návrhu architektury je třeba brát v potaz zvolené vývojové nástroje. GUI centrály je realizováno pomocí technologie ASP.NET 3.5. V předchozích verzích ASP.NET byly webové aplikace vyvíjeny jako klasické třívrstvé aplikace. Vrstva přístupu k datům byla tvořena nejčastěji uloženými procedurami. Objekty aplikační vrstvy, tzv. business objekty, pak byly vytvářeny pomocí volání uložených procedur. Business objekty realizovaly aplikační logiku. Prezentační vrstva pak byla tvořena třídami stránek ASP.NET, které k business objektům přistupovaly pomocí *vázání dat* (angl. *data-binding*). Technologie ASP.NET umožňuje automatizovat proces vázání dat pomocí tzv. data source objektů (nejčastěji se používá *ObjectDataSource*). Obrázek 12 ukazuje nejčastěji používanou architekturu webových aplikací realizovaných pomocí technologie ASP.NET 2.0.



Obrázek 12: Architektura webové aplikace v ASP.NET 2.0

Ve verzi ASP.NET 3.5, která je použita pro vývoj centrály, je přístup k datům ještě více zjednodušen pomocí technologie *LINQ to SQL*. LINQ je do jazyků C# a Visual Basic .NET integrovaný dotazovací jazyk nad libovolnými objekty implementujícím rozhraní *IEnumerable*. LINQ to SQL je technologie objektově-relačního mapování objektů prostředí .NET na databázové objekty, se kterými lze operovat pomocí LINQ výrazů. První dvě vrstvy předchozího modelu architektury pak lze nahradit vrstvou jednou tvořenou business objekty vytvořenými pomocí LINQ to SQL. Tyto objekty lze doplnit o aplikační logiku pomocí tzv. *partial classes* (rozdělené třídy), kdy stav (vlastnosti třídy) je získán pomocí objektově-relačního mapování a aplikační logiku lze přidat pomocí vlastních metod, jejichž kód je umístěn do souborů mimo kód vygenerovaný nástrojem automatizujícím proces objektově-relačního mapování. Přístup prezentační vrstvy k business objektům může navíc využívat nového prostředku vázání dat speciálně vyvinutého pro jazyk LINQ – objektu *LinqDataSource*. Architektura webových aplikací v ASP.NET 3.5 za použití technologie LINQ to SQL je uvedena na obrázku 13.

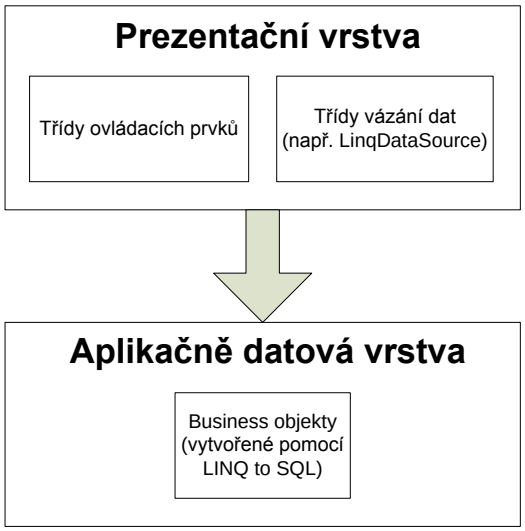


Obrázek 13: Architektura webové aplikace v ASP.NET 3.5

Architektura GUI centrály vychází z nového modelu architektury webové aplikace, který přinesla verze ASP.NET 3.5. Služby centrály jsou implementované jako služby operačního systému Windows, nemají tedy žádné GUI, pracují přímo s business objekty vytvořenými pomocí LINQ to SQL.

5.6 Návrhové třídy

Roli návrhových tříd hrají třídy vytvořené pomocí objektově-relačního mapování technologie LINQ to SQL. Pro každou tabulku v databázi (viz návrh fyzického schématu databáze v části 5.2) je vytvořena jí odpovídající třída a doplněny případné metody realizující aplikační logiku. Tyto třídy spadají do aplikačně datové vrstvy. V prezentační vrstvě hrají nejvýznamnější roli třídy reprezentující ovládací prvky a třídy vázání dat (např. LinqDataSource či ObjectDataSource). Příslušnost výše popsaných tříd do vrstev ukazuje obrázek 14.



Obrázek 14: Příslušnost návrhových tříd do vrstev

6 Implementace

Tato kapitola se zaměřuje na definování rozsahu návrhu centrály, který byl ve fázi implementace realizován. Na diagramu nasazení je ukázáno nasazení implementovaných částí centrály na výpočetní zdroje v prostředí interní sítě společnosti AVG Technologies.

6.1 Definování rozsahu implementace

Z kapitol 4 a 5 lze usoudit, že implementace centrály jako celku by byla značně časově náročná. Jedná se o rozsáhlý systém. Pro posouzení správnosti návrhu a přínosu tohoto systému zaměstnancům společnosti AVG Technologies ale nebude nezbytně nutné, aby musely být implementovány všechny části centrály. Po diskuzi se zainteresovanými zaměstnanci společnosti AVG Technologies byly pro implementaci vybrány následující části návrhu centrály:

- databáze centrály,
- služby pro zpracování zpráv,
- utility realizující základní tok zpracování pro současný typ vstupu generovaný monitorem AVG Diag,
- webová aplikace realizující GUI centrály, kde bude implementován detailní pohled na zpracovanou zprávu,
- statistika počtu vytvořených problému se shodnou hodnotou za dané časové období (viz část 4.1.14).

Realizace uvedených částí návrhu centrály pokrývá všechny nefunkční požadavky (viz část 4.3) a funkční požadavky definované případy užití Zpracovat zprávu, Zobrazit detail zprávy a Zobrazit statistiku. Po implementaci těchto částí návrhu centrála pomáhá:

- vývojářům produktů, kteří mají díky statistikám přehled o aktuálně se vyskytujících problémech,
- zaměstnancům oddělení technické podpory, kteří primárně pracují se systémem Genesis (systém pro sledování komunikace s koncovými uživateli, viz část 4.1.10) a díky existenci propojení záznamu zprávy v databázi centrály se záznamem zprávy v systému Genesis mohou jednoduše přejít ze systému Genesis do centrály, kde vidí všechny automaticky provedené analýzy a nemusejí je vykonávat manuálně (propojení je realizováno pomocí unikátního identifikátoru přítomného v každé zprávě nazvaného AVGDIAG-GUID).

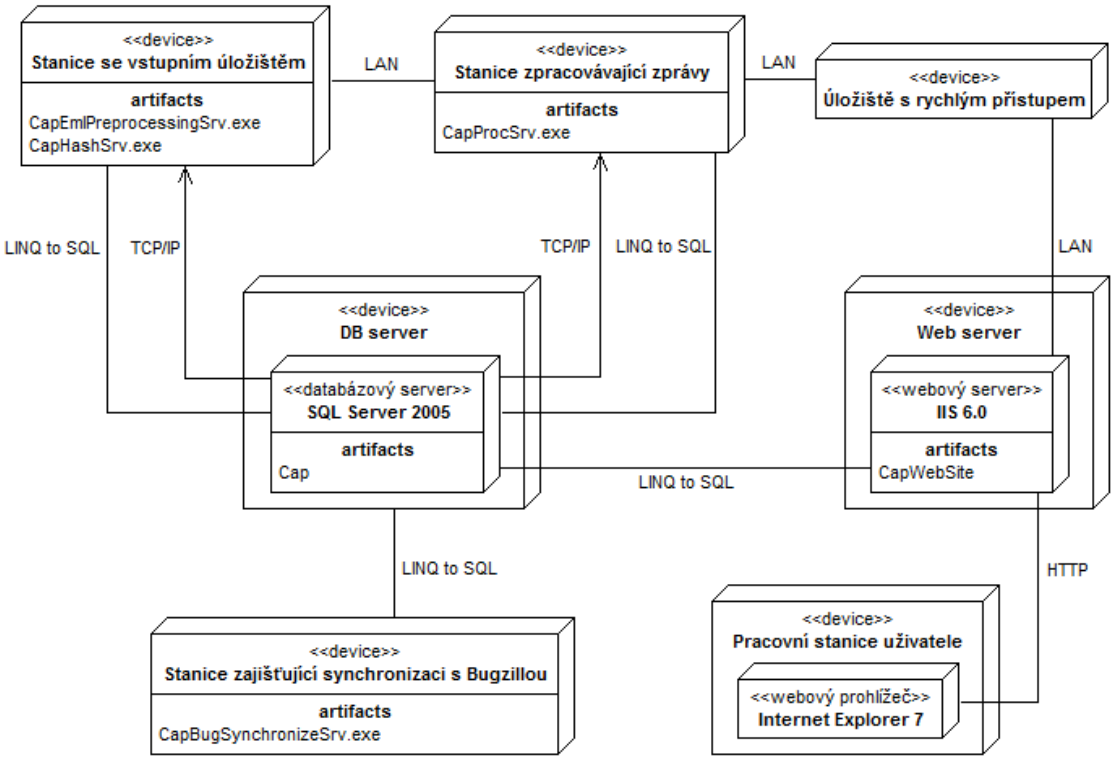
Dále byla implementována jedna komponenta, která adaptuje obecný návrh centrály do prostředí společnosti AVG Technologies. Touto komponentou je služba, která má za úkol transformovat zprávy z formátu souborů EML do formátu ZIP. Je tomu tak proto, že v současné době mohou být některé zprávy z důvodu větší velikosti rozděleny na několik menších EML souborů a pak je třeba tyto soubory před vlastním zpracováním zprávy centrálou sloučit do jednoho souboru zprávy

(tak jak to předpokládá návrh centrály). Toto předzpracování zprávy má také další výhodu a to, že se uživatelům centrály bude se soubory formátu ZIP pracovat lépe než s formátem EML.

Poslední implementovanou funkční jednotkou centrály je služba realizující případ použití Synchronizovat se systémem pro sledování chyb. Realizace tohoto případu použití nebyla diskutována v návrhu centrály (viz kapitola 5), protože implementace této funkčnosti závisí na systému pro sledování chyb, který se v prostředí společnosti, kde bude centrála nasazena, používá (ve společnosti AVG Technologies to je open-source řešení Bugzilla).

6.2 Diagram nasazení

Na obrázku 15 se nachází diagram nasazení implementovaných částí centrály.



Obrázek 15: Diagram nasazení

Zpráva přichází na vstupní úložiště (výpočetní zdroj *Stanice se vstupním úložištěm*), kde je soubor zprávy službou *CapEmlPreprocessingSrv.exe* převeden z formátu EML do formátu ZIP. Následně je pro zprávu službou *CapHashSrv.exe* vygenerován hash a záznam o příchodu zprávy na úložiště je vložen do databáze *Cap* (běží v prostředí databázového serveru *SQL Server 2005* na výpočetním zdroji *DB server*).

Ze vstupního úložiště jsou zprávy odebírány službou *CapProcSrv.exe* (výpočetní zdroj *Stanice zpracovávající zprávy*), která zprávu zpracuje (viz část 5.3.2), výsledky zpracování uloží do databáze a významné soubory na úložiště s rychlým přístupem (výpočetní zdroj *Úložiště s rychlým přístupem*).

Webová aplikace realizující GUI centrály *CapWebSite* běží v prostředí webového serveru *IIS 6.0* na výpočetním zdroji *Web server*. S GUI centrály pracují uživatelé, kteří k němu přistupují

ze svých stanic (výpočetní zdroj *Pracovní stanice uživatele*) za pomoci webového prohlížeče (podporovány jsou Internet Explorer 7 a nejnovější verze prohlížečů Firefox a Opera).

Služba zajišťující synchronizaci se systémem pro sledování chyb *CapBugSynchronize-Srv.exe* běží na výpočetním zdroji *Stanice zajišťující synchronizaci s Bugzillou*.

Části centrály si mezi sebou předávají soubory (soubor zprávy, významné soubory) prostřednictvím sdílení adresářů v rámci lokální sítě (komunikační cesty *LAN*). Komunikační cesty *LINQ to SQL* reprezentují komunikaci služeb a webové aplikace s databází. Databáze posílá službám notifikace prostřednictvím TCP/IP socketů (komunikační cesta *TCP/IP*). Webový prohlížeč přistupuje k webové aplikaci pomocí protokolu HTTP (komunikační cesta *HTTP*).

7 Zhodnocení výsledků a rozšíření

Již v průběhu implementace byly realizované části centrály průběžně testovány, zejména byl kladen důraz na otestování správného chování procesu zpracování zpráv, správného provádění toku zpracování a vytváření problémů. Při testování byla používána ostrá data získávaná monitorem AVG Diag.

Souběžně s vývojem centrály probíhal také vývoj nového produktu společnosti AVG Technologies, AVG 8.0. Program AVG Diag používaný v tomto produktu generoval zprávy, které obsahovaly odlišně strukturované informace než v předchozích verzích. S vydáním AVG 8.0 se tak mohla otestovat schopnost centrály rychle reagovat na změny ve vstupních datech. Návrh centrály počítal se snadnou modifikovatelností procesu zpracování zpráv. Této vlastnosti centrály bylo využito již při beta-testování AVG 8.0 a bylo ověřeno, že modifikace procesu zpracování zpráv skutečně nevyžaduje žádnou změnu v kódu služeb zpracování, pouze změnu kódu externích utilit.

V současné době jsou implementovány všechny části centrály, které byly v části 6.1 vybrány k realizaci. Předpoklad, že centrála bude pomáhat vývojářům, byl naplněn již při beta-testování AVG 8.0, kdy tento produkt generoval poměrně velké množství chyb, a vývojáři ocenili dostupnost aktuálního přehledu o provozních chybách ve formě statistik poskytovaných centrálou. Měli možnost vidět, které problémy se vyskytovaly nejčastěji a řešení problémů tak mohli vhodně prioritizovat. Využitelnost centrály pro zaměstnance technické podpory do značné míry závisí na vhodně zvolených doplňkových analýzách a činnostech, které bude tok zpracování realizovat automatizovaně. V prvních verzích proto zaměstnanci technické podpory centrálu nevyužívali příliš často, to se ale s přibývajícím množstvím vhodných utilit mění a také synchronizace problémů s chybami v systému pro sledování chyb Bugzilla může centrálu zaměstnancům technické podpory zatraktivnit (jestliže je ve zprávě nalezen známý problém, který má nastaven správný odkaz do Bugzilly, může si zaměstnanec technické podpory rychle najít další informace o problému v systému Bugzilla a poskytnout tak koncovému uživateli řešení jeho problému rychleji).

Lze tedy konstatovat, že implementovaná část centrály splnila očekávání a ve společnosti AVG Technologies pomáhá s detekcí a řešením provozních chyb. V současné době je ale implementována pouze část návrhu centrály, a proto případné další práce na projektu budou směřovat k dokončení realizace stávajícího návrhu centrály. Po dokončení této fáze lze pak systém dále rozvíjet třemi hlavními směry:

- upravit monitor, přenosový kanál a centrálu tak, aby bylo možné zprovoznit zpětnou vazbu na uživatele produktů při detekování známých problémů (tj. centrála bude úzce svázána s monitorem a přenosovým kanálem – odklon od současného obecného návrhu),
- využít technik dolování asociačních pravidel pro detekování často se vyskytujících kombinací analytických položek ve vybrané množině zpráv (usnadní hledání nových typů problémů),
- dále rozvíjet možnosti, které centrála poskytuje pro automatizované zpracování (tj. odklon od automatizace zpracování zpráv k automatizaci jiných typů procesů).

8 Závěr

Cílem práce bylo analyzovat požadavky na centrálu systému pro analýzu provozních chyb, na jejich základě vytvořit návrh centrály a realizovat vhodnou část návrhu tak, aby mohla být vyhodnocena jeho správnost a praktická využitelnost centrály.

Požadavky byly získány především prostřednictvím konzultací se zaměstnanci společnosti AVG Technologies, která bude prvním uživatelem centrály po její budoucí implementaci. Sběr požadavků a jejich analýzu lze považovat za úspěšně provedenou, budoucí uživatelé centrály (zaměstnanci společnosti AVG Technologies) odsouhlasili všechny uvedené požadavky jako realizovatelné.

Na základě získaných požadavků byl vytvořen model případů použití a byly specifikovány nefunkční požadavky. Analýza požadavků byla ukončena definováním konceptuálních tříd, které byly ve fázi návrhu rozšířeny o některé další třídy tak, aby použitím všech tříd konceptuálního modelu bylo možné požadované chování centrály realizovat. Návrh centrály dále pokračoval definováním fyzického schématu databáze, návrhem procesu zpracování zpráv a návrhem architektury.

Po diskuzi se zainteresovanými zaměstnanci společnosti AVG Technologies byla pro implementaci zvolena část návrhu centrály, která by měla demonstrovat možnosti centrály a její praktickou využitelnost pro vývojáře a zaměstnance technické podpory. Při vydání nového produktu AVG 8.0 byla centrála zapojena do procesu detekce provozních chyb a bylo konstatováno, že centrála naplnila očekávané představy. V realizaci centrály by tak mělo být pokračováno, systém se může stát jedním z důležitých nástrojů pro automatizované zpracování hlášení o problémech koncových uživatelů produktů společnosti AVG Technologies.

Přínos práce spočívá především v návrhu centrály systému pro analýzu provozních chyb. Realizací návrhu lze implementovat centrálu v prostředí libovolné společnosti zabývající se vývojem softwarových produktů. Při řešení práce a studiu literatury se autor nesetkal s podobně obecným a detailním řešením problematiky. Práce tedy může mít kromě realizovaného prakticky použitelného výstupu (v podobě implementované části centrály v prostředí společnosti AVG Technologies) také význam pro další studium problematiky automatizovaného zpracování hlášení o chybách softwaru v provozním prostředí.

Autor při řešení práce získal:

- zkušenosti se sběrem požadavků, kdy měl na rozdíl od školních projektů možnost přímého kontaktu s budoucími uživateli vyvíjeného systému,
- zkušenosti s implementací prakticky používaného systému (nutnost reakce na připomínky uživatelů k dosud realizovanému výstupu),
- osvojil si nejnovější technologie vývoje informačních systémů na platformě Microsoft Windows (Microsoft .NET Framework 3.5).

Literatura

[1] AVG Technologies. *Podporované platformy* [online]. [cit. 2008-20-04]. Dostupné na URL: <http://www.avg.cz/doc/24/cz/crp/0>

[2] *Core dump* [online]. Poslední modifikace: 20. dubna 2008. [cit. 2008-20-04]. Dostupné na URL: <http://en.wikipedia.org/wiki/Core_dump>.

[3] Microsoft. *Microsoft Online Crash Analysis* [online]. [cit. 2008-23-04]. Dostupné na URL: <http://oca.microsoft.com/en/Welcome.aspx>

[4] Northrup, T. *Using Microsoft Online Crash Analysis* [online]. Poslední modifikace: 6. října 2003 [cit. 2008-23-04]. Dostupné na URL: <http://www.microsoft.com/windowsxp/using/setup/expert/northrup_03october06.mspx>

[5] Apple. *Technical Note TN2123: CrashReporter* [online]. Poslední modifikace: 1. dubna 2008 [cit. 2008-24-04]. Dostupné na URL: <http://developer.apple.com/technotes/tn2004/tn2123.html>

[6] Google. *google-breakpad: Crash reporting* [online]. [cit. 2008-25-04]. Dostupné na URL: <http://code.google.com/p/google-breakpad>

[7] *Crash Reporter* [online]. Poslední modifikace: 17. dubna 2008 [cit. 2008-25-04]. Dostupné na URL: <http://en.wikipedia.org/wiki/Crash_reporter>

[8] CZilla Tým. *Talkback - pomocník při pádu Mozilly: Jak na pády aplikací Mozilla.org* [online]. Poslední modifikace: 16. listopadu 2004 [cit. 2008-26-04]. Dostupné na URL: <http://www.czilla.cz/podpora/talkback.html>

[9] Clause, J., Orso, A. *A Technique for Enabling and Supporting Debugging of Field Failures* [online]. [cit. 2008-27-04]. Dostupné na URL: <http://groups.csail.mit.edu/pag/reading-group/clause07field.ps>

[10] AVG Technologies. *Profil společnosti* [online]. [cit. 2008-27-04]. Dostupné na URL: <http://www.avg.cz/cz.71>

[11] Refsnes Data. *DTD Tutorial* [online]. [cit. 2008-02-05]. Dostupné na URL: <http://www.w3schools.com/dtd/default.asp>

[12] MacDonald, M., Szpuszta, M. *ASP.NET 2.0: Tvorba dynamických stránek profesionálně*. 1. vydání. Zoner Press, Brno, 2006. ISBN 80-86815-38-2

[13] Microsoft. *The LINQ Project* [online]. [cit. 2008-03-05]. Dostupné na URL: <http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>

Příloha A

Popis fyzického schématu databáze

Tabulka	Configuration (Konfigurace)
Popis	Konfigurace procesu zpracování zpráv.
Sloupec	**Popis**
Id	Identifikátor konfigurace. Primární klíč.
Name	Název konfigurace.
Abbreviation	Zkratka konfigurace.
Description	Popis konfigurace. Prázdná hodnota je povolena.
IsTested	Příznak určující zda je konfigurace otestována.
IsActual	Příznak určující zde se jedná o aktuální konfiguraci. V systému se nachází vždy právě jedna aktuální konfigurace.
AbortProcessingTime	Doba (v minutách), za kterou bude započaté zpracování prohlášeno za neplatné a zpráva bude moci být službami opětovně zpracovávána.
MaxRecoveryAttempts	Maximální možný počet pokusů o opětovné zpracování nezpracované zprávy.

Tabulka	Workflow (Tok zpracování)
Popis	Tok zpracování definující kroky zpracování pro konkrétní typ zprávy.
Sloupec	**Popis**
Id	Identifikátor toku zpracování. Primární klíč.
ConfigurationId	Cizí klíč do tabulky Configuration. Realizuje vztah konfigurace *obsahuje* tok zpracování.
MessageTypeId	Cizí klíč do tabulky MessageType. Realizuje vztah tok zpracování *zpracovává* typ zprávy.

Tabulka	WorkflowStep (Krok zpracování)
Popis	Krok zpracování realizující pomocí utility některou z činností toku zpracování.
Sloupec	**Popis**
Id	Identifikátor kroku zpracování. Primární klíč.
StepNumber	Pořadí kroku zpracování v toku zpracování.
FirmParameters	Pevné parametry, které budou předány utilitě při jejím spuštění. Prázdná hodnota je povolena a znamená, že utilitě nebudou předány žádné pevné parametry.
MaxDuration	Maximální možná doba (v sekundách), po kterou bude služba čekat na dokončení běhu utility. Pokud utilita neskončí do uplynutí této doby, služba proces utility ukončí pomocí standardních prostředků operačního systému.
WorkflowId	Cizí klíč do tabulky Workflow. Realizuje vztah tok zpracování *obsahuje* krok zpracování.
UtilityId	Cizí klíč do tabulky Utility. Realizuje vztah utilita *je spouštěna* krokem zpracování. Prázdná hodnota je povolena a znamená, že se jedná o poslední systémový krok zpracování (není spouštěna žádná utilita, ale provádí se přímo kód služby). Systém vždy zajistí, aby každý tok zpracování končil systémovým krokem zpracování.

Tabulka	Utility (Utilita)
Popis	Utilita realizující některou z činností toku zpracování.
Sloupec	**Popis**
Id	Identifikátor utility. Primární klíč.
Name	Název utility.
Abbreviation	Zkratka utility.
Description	Popis utility. Prázdná hodnota je povolena.
SuccessExitCode	Návratový kód očekávaný v případě úspěšného ukončení běhu utility.

Tabulka	Component (Komponenta)
Popis	Komponenta potřebná k běhu utility.
Sloupec	**Popis**
Id	Identifikátor komponenty. Primární klíč.
Name	Název komponenty.

Tabulka	Utility_Component
Popis	Vazební tabulka realizující vztah utilita *se skládá* z komponenty.
Sloupec	**Popis**
UtilityId	Cizí klíč do tabulky Utility. Společně se sloupcem ComponentId tvoří primární klíč.
ComponentId	Cizí klíč do tabulky Component. Společně se sloupcem UtilityId tvoří primární klíč.

Tabulka	MessageType (Typ zprávy)
Popis	Typ zprávy, který lze zpracovat pomocí některého kroku zpracování.
Sloupec	**Popis**
Id	Identifikátor typu zprávy. Primární klíč.
Name	Název typu zprávy.
Abbreviation	Zkratka typu zprávy.
Description	Popis typu zprávy. Prázdná hodnota je povolena.

Tabulka	ProblemType (Typ problému)
Popis	Na základě typu problému jsou detekovány a vytvářeny konkrétní problémy. Typ problému je definován typy analytických položek a typy významných souborů (viz příslušné vazební tabulky).
Sloupec	**Popis**
Id	Identifikátor typu problému. Primární klíč.
Name	Název typu problému.
Abbreviation	Zkratka typu problému.
Description	Popis typu problému. Prázdná hodnota je povolena.

Tabulka	Configuration_ProblemType
Popis	Vazební tabulka realizující vztah konfigurace *detekuje* typ problému.
Sloupec	**Popis**
ConfigurationId	Cizí klíč do tabulky Configuration. Společně se sloupcem ProblemTypeId tvoří primární klíč.
ProblemTypeId	Cizí klíč do tabulky ProblemType. Společně se sloupcem ConfigurationId tvoří primární klíč.

Tabulka	ImportantFileType (Typ významného souboru)
Popis	Typ významného souboru zprávy, ve kterém lze hledat analytické položky.
Sloupec	**Popis**
Id	Identifikátor typu významného souboru. Primární klíč.
RegularExpression	Regulární výraz specifikující významné soubory typu (např. „*.dmp", „*.log").
Name	Název typu významného souboru.
Abbreviation	Zkratka typu významného souboru.
Description	Popis typu významného souboru. Prázdná hodnota je povolena.
FileTypeOrder	Pořadí zobrazení typu významného souboru v detailním pohledu na zprávu. Prázdná hodnota je povolena, pokud typ významného souboru představuje doplňkovou analýzu nebo je použit pouze pro předání parametrů externí utilitě (tj. nikdy nebude vytvořen žádný významný soubor typu).
FastAccessStorageSaveEnabled	Příznak určující zda budou významné soubory typu ukládány na úložiště s rychlým přístupem.
IsAdditionalAnalysis	Příznak určující zda typ významného souboru reprezentuje doplňkovou analýzu.
ParentImportantFileTypeId	Cizí klíč do tabulky ImportantFileType. Realizuje vztah typ významného souboru *je potomkem* jiného typu významného souboru. Prázdná hodnota je povolena a znamená, že typ významného souboru není potomkem žádného jiného typu významného souboru.

Tabulka	WorkflowStep_ImportantFileType
Popis	Vazební tabulka realizující vztah tok zpracování *sleduje* významný soubor určitého typu.
Sloupec	**Popis**
WorkflowStepId	Cizí klíč do tabulky WorkflowStep. Společně se sloupcem ImportantFileTypeId tvoří primární klíč.
ImportantFileTypeId	Cizí klíč do tabulky ImportantFileType. Společně se sloupcem WorkflowStepId tvoří primární klíč.

Tabulka	ProblemType_ImportantFileType
Popis	Vazební tabulka realizující vztah typ problému *je definován* typem významného souboru.
Sloupec	**Popis**
ProblemTypeId	Cizí klíč do tabulky ProblemType. Společně se sloupcem ImportantFileTypeId tvoří primární klíč.
ImportantFileTypeId	Cizí klíč do tabulky ImportantFileType. Společně se sloupcem ProblemTypeId tvoří primární klíč.
ProblemValueOrder	Pořadí názvu významného souboru v hodnotě problému.

Tabulka	AnalyticalItemType (Typ analytické položky)
Popis	Typ analytické položky, kterou lze hledat ve významných souborech zprávy.
Sloupec	**Popis**
Id	Identifikátor typu analytické položky. Primární klíč.
Expression	Výraz, podle kterého lze v souboru analytickou položku typu detekovat.
Name	Název typu analytické položky.
Abbreviation	Zkratka typu analytické položky.
Description	Popis typu analytické položky. Prázdná hodnota je povolena.
IsMessageLevel	Příznak určující zda se analytická položka typu vyskytuje ve zprávě pouze jednou (případně vícekrát, ve více souborech, ale vždy se stejnou hodnotou).
MessageLevelOrder	Určuje pořadí zobrazení hodnoty analytické položky typu. Prázdná hodnota je povolena, pokud typ analytické hodnoty nemá nastaven příznak IsMessageLevel.
HasValue	Příznak určující zda typ analytické položky má hodnotu. Pokud ano, tak v tabulce AnalyticalItem existuje záznam pro každou nalezenou hodnotu analytické položky. Pokud typ analytické položky nemá hodnotu, pak v tabulce AnalyticalItem existuje pouze jeden záznam s hodnotou rovnou výrazu typu analytické položky.
IsMessageUniqueIdentifier	Příznak určující zda hodnota analytické položky typu jednoznačně identifikuje zprávu v databázi.
AliasValuesEnabled	Příznak určující zda lze pro hodnotu analytické položky typu definovat alias.
DistributionStatisticEnabled	Příznak určující zda lze pro typ analytické položky vytvořit statistiku distribuce počtu nalezených problémů přes hodnoty analytických položek za dané časové období.
ParentAnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Realizuje vztah typ analytické položky *je potomkem* jiného typu analytické položky. Prázdná hodnota je povolena a znamená, že typ analytické položky není potomkem žádného jiného typu analytické položky.
MoreGeneralAnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Realizuje vztah typ analytické položky *je specifickým podtypem* jiného typu analytické položky. Prázdná hodnota je povolena a znamená, že typ analytické položky není specifickým podtypem žádného jiného typu analytické položky.

Tabulka	WorkflowStep_AnalyticalItemType
Popis	Vazební tabulka realizující vztah krok zpracování *získává* analytické položky určitého typu.
Sloupec	**Popis**
WorkflowStepId	Cizí klíč do tabulky WorkflowStep. Společně se sloupcem AnalyticalItemTypeId tvoří primární klíč.
AnalyticalItemTypeId	Cizí klíč do tabulky ImportantFileType. Společně se sloupcem WorkflowStepId tvoří primární klíč.

Tabulka	ProblemType_AnalyticalItemType
Popis	Vazební tabulka realizující vztah typ problému *je definován* typem analytické položky.
Sloupec	**Popis**
ProblemTypeId	Cizí klíč do tabulky ProblemType. Společně se sloupcem AnalyticalItemTypeId tvoří primární klíč.
AnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Společně se sloupcem ProblemTypeId tvoří primární klíč.
ProblemValueOrder	Pořadí hodnoty (resp. výrazu typu) analytické položky v hodnotě problému.

Tabulka	ImportantFileType_AnalyticalItemType
Popis	Vazební tabulka realizující vztah analytická položka určitého typu *se hledá* ve významném souboru určitého typu.
Sloupec	**Popis**
ImportantFileTypeId	Cizí klíč do tabulky ImportantFileType. Společně se sloupcem AnalyticalItemTypeId tvoří primární klíč.
AnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Společně se sloupcem ImportantFileTypeId tvoří primární klíč.
AnalyticalItemOrder	Pořadí hodnoty (resp. výrazu typu) analytické položky typu při zobrazení analytických položek nalezených ve významném souboru typu.

Tabulka	MessageType_AnalyticalItemType
Popis	Vazební tabulka realizující vztah analytická položka určitého typu *se hledá* v souboru zprávy určitého typu. V případě, že je s krokem zpracování spojen typ analytické položky, který lze hledat v souboru zprávy, pak hodnoty `<id typu souboru>`, `<id typu souboru otce>` a `<regulární výraz typu souboru>` parametru `-i` jsou rovny `0`, `0` a `_M_` (viz část 5.3.3).
Sloupec	**Popis**
MessageTypeId	Cizí klíč do tabulky MessageType. Společně se sloupcem AnalyticalItemTypeId tvoří primární klíč.
AnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Společně se sloupcem MessageTypeId tvoří primární klíč.

Tabulka	Processing (Zpracování)
Popis	Záznam o zpracování zprávy.
Sloupec	**Popis**
Id	Identifikátor zpracování. Primární klíč.
Start	Začátek zpracování.
End	Konec zpracování. Prázdná hodnota je povolena a znamená, že zpracování nebylo dokončeno.
WorkflowId	Cizí klíč do tabulky Workflow. Realizuje vztah zpracování *je provedeno* tokem zpracování.
ServiceId	Cizí klíč do tabulky Service. Realizuje vztah služba *provádí* zpracování.
MessageId	Cizí klíč do tabulky Message. Realizuje vztah zpráva *je zpracována* zpracováním.

Tabulka	ExitCode (Návratový kód)
Popis	Návratový kód utility kroku zpracování.
Sloupec	**Popis**
WorkflowStepId	Cizí klíč do tabulky WorkflowStep. Realizuje vztah krok zpracování *zaznamenává* návratový kód. Společně se sloupcem ProcessingId tvoří primární klíč.
ProcessingId	Cizí klíč do tabulky Processing. Realizuje vztah zpracování *obsahuje* návratový kód. Společně se sloupcem WorkflowStepId tvoří primární klíč.
Value	Návratový kód utility kroku zpracování. Prázdná hodnota je povolena a znamená, že provádění utility bylo službou ukončeno v důsledku překročení maximální doby trvání běhu utility definované krokem zpracování.

Tabulka	InputStorage (Vstupní úložiště)
Popis	Vstupní úložiště, na které přicházejí do systému zprávy.
Sloupec	**Popis**
Id	Identifikátor vstupního úložiště. Primární klíč.
Name	Název vstupního úložiště.
Abbreviation	Zkratka vstupního úložiště.
Description	Popis vstupního úložiště. Prázdná hodnota je povolena.
IsForOnDemandAnalysis	Příznak určující zda zprávy, které přicházejí na vstupní úložiště, mají být postoupeny k analýze na vyžádání.
Path	UNC cesta identifikující vstupní úložiště v síti.
MessageTypeId	Cizí klíč do tabulky MessageType. Realizuje vztah zprávy určitého typu *jsou odebírány* ze vstupního úložiště.

Tabulka	ServiceType (Typ služby)
Popis	Typ služby definuje pro službu základní parametry a vstupní úložiště (viz příslušná vazební tabulka), která jsou službou obsluhována.
Sloupec	**Popis**
Id	Identifikátor typu služby. Primární klíč.
Name	Název typu služby.
Abbreviation	Zkratka typu služby.
Description	Popis typu služby. Prázdná hodnota je povolena.
MaxRunningThreads	Maximální počet současně běžících vláken zpracování. Prázdná hodnota je povolena. Typ služby s prázdnou hodnotou v tomto sloupci nemůže být přiřazen službě pro zpracování zpráv.
RecoveryThreadPeriod	Určuje dobu (v sekundách) mezi dvěma po sobě následujícími pokusy o opětovné zpracování zpráv (tj. interval zotavení, viz část 5.3.2). Prázdná hodnota je povolena. Typ služby s prázdnou hodnotou v tomto sloupci nemůže být přiřazen službě pro zpracování zpráv.

Tabulka	ServiceType_InputStorage
Popis	Vazební tabulka realizující vztah služba určitého typu *obsluhuje* vstupní úložiště.
Sloupec	**Popis**
ServiceTypeId	Cizí klíč do tabulky ServiceType. Společně se sloupcem InputStorageId tvoří primární klíč.
InputStorageId	Cizí klíč do tabulky InputStorage. Společně se sloupcem ServiceTypeId tvoří primární klíč.
InputStoragePriority	Priorita vstupního úložiště. U služby pro zpracování zpráv určuje počet zpráv, který je z úložiště službou odebrán, než se přesune na další úložiště s nižší prioritou (zde vyšší hodnota znamená větší prioritu). U služby hashovací určuje pořadí, ve kterém bude služba vstupní úložiště obsluhovat (zde menší hodnota znamená větší prioritu).

Tabulka	Service (Služba)
Popis	Služba pro zpracování zpráv, nebo služba hashovací.
Sloupec	**Popis**
Id	Identifikátor služby. Primární klíč.
IPAddress	IP adresa stroje, na kterém služba běží (formát IPv4).
IsRunning	Příznak určující zda služba běží.
IsHashingService	Příznak určující zda se jedná o hashovací službu, nebo službu pro zpracování zpráv.
ServiceTypeId	Cizí klíč do tabulky ServiceType. Realizuje vztah služba *je instancí* typu služby. Prázdná hodnota je povolena a znamená, že službě nebyl dosud přidělen žádný typ. Služba bez typu neprovádí žádnou činnost a čeká na notifikaci, která jí sdělí, že jí byl typ přidělen.

Tabulka	Service_InputStorage
Popis	Vazební tabulka realizující vztah služba *je nečinná* na vstupním úložišti. Pokud existuje pro službu a vstupní úložiště v této tabulce záznam, znamená to, že služba neodebírá z úložiště zprávy (zjistila, že na úložišti není žádná zpráva). Takové službě se po příchodu nové zprávy na vstupní úložiště musí poslat notifikace a záznam z této tabulky se musí odmazat (tj. služba zprávy začne odebírat a záznam do této tabulky vloží, až z daného vstupního úložiště opět odebere všechny zprávy).
Sloupec	**Popis**
ServiceId	Cizí klíč do tabulky Service. Společně se sloupcem InputStorageId tvoří primární klíč.
InputStorageId	Cizí klíč do tabulky InputStorage. Společně se sloupcem ServiceId tvoří primární klíč.

Tabulka	Message (Zpráva)
Popis	Zpráva o provozní chybě.
Sloupec	**Popis**
Id	Identifikátor zprávy. Primární klíč.
IncomingDate	Datum příchodu zprávy na vstupní úložiště.
FileName	Název souboru zprávy.
FileSize	Velikost souboru zprávy v bajtech.
FileHash	Hash souboru zprávy spočítaný hashovací funkcí SHA-1.
UserDescription	Popis problému od koncového uživatele.
TakenForProcessing	Příznak určující zda již zpráva byla odebrána ze vstupního úložiště ke zpracování.
InputStorageId	Cizí klíč do tabulky InputStorage. Realizuje vztah zpráva *je odebrána* ze vstupního úložiště.
MessageTypeId	Cizí klíč do tabulky MessageType. Realizuje vztah zpráva *je instancí* typu zprávy.
UserId	Cizí klíč do tabulky User. Realizuje vztah uživatel *reanalyzuje* zprávu. Prázdná hodnota je povolena a znamená, že zpráva dosud nebyla reanalyzována.

Tabulka	CommonFile (Běžný soubor)
Popis	Běžný soubor detekovaný ve zprávě.
Sloupec	**Popis**
Id	Identifikátor běžného souboru. Primární klíč.
FileName	Název běžného souboru.

Tabulka	Message_CommonFile
Popis	Vazební tabulka realizující vztah zpráva *obsahuje* obyčejný soubor.
Sloupec	**Popis**
MessageId	Cizí klíč do tabulky Message. Společně se sloupcem CommonFileId tvoří primární klíč.
CommonFileId	Cizí klíč do tabulky CommonFile. Společně se sloupcem MessageId tvoří primární klíč.

Tabulka	ImportantFile (Významný soubor)
Popis	Významný soubor detekovaný ve zprávě.
Sloupec	**Popis**
Id	Identifikátor významného souboru. Primární klíč.
FileName	Název významného souboru.
FileSize	Velikost významného souboru v bajtech.
FileHash	Hash významného souboru spočítaný hashovací funkcí SHA-1.
MessageId	Cizí klíč do tabulky Message. Realizuje vztah zpráva *obsahuje* významný soubor.
ImportantFileTypeId	Cizí klíč do tabulky ImportantFileType. Realizuje vztah zpráva *je instancí* typu zprávy.
ParentImportantFileId	Cizí klíč do tabulky ImportantFile. Realizuje vztah významný soubor *je potomkem* jiného významného souboru. Prázdná hodnota je povolena a znamená, že významný soubor není potomkem žádného jiného významného souboru.

Tabulka	AnalyticalItem (Analytická položka)
Popis	Analytická položka nalezená v některém z významných souborů zprávy.
Sloupec	**Popis**
Id	Identifikátor analytické položky. Primární klíč.
Value	Hodnota analytické položky.
AnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Realizuje vztah analytická položka *je instancí* typu analytické položky.
AliasId	Cizí klíč do tabulky Alias. Realizuje vztah analytická položka *má definován* alias. Prázdná hodnota je povolena a znamená, že hodnota analytické položky nemá definován žádný alias.

Tabulka	Alias (Alias)
Popis	Alias hodnoty analytické položky.
Sloupec	**Popis**
Id	Identifikátor aliasu. Primární klíč.
Value	Hodnota aliasu.
AnalyticalItemTypeId	Cizí klíč do tabulky AnalyticalItemType. Realizuje vztah typ analytické položky *umožňuje definovat* alias.

Tabulka	Message_AnalyticalItem
Popis	Vazební tabulka realizující vztah analytická položka *je nalezena* v souboru zprávy.
Sloupec	**Popis**
MessageId	Cizí klíč do tabulky Message. Společně se sloupcem AnalyticalItemId tvoří primární klíč.
AnalyticalItemId	Cizí klíč do tabulky AnalyticalItem. Společně se sloupcem MessageId tvoří primární klíč.

Tabulka	ImportantFile_AnalyticalItem
Popis	Vazební tabulka realizující vztah analytická položka *je nalezena* ve významném souboru zprávy.
Sloupec	**Popis**
Id	Identifikátor záznamu. Primární klíč.
ImportantFileId	Cizí klíč do tabulky ImportantFile.
AnalyticalItemId	Cizí klíč do tabulky AnalyticalItem.
ParentImportantFile_AnalyticalItemId	Cizí klíč do tabulky ImportantFile_AnalyticalItem. Realizuje vztah analytická položka nalezená ve významném souboru *je potomkem* jiné analytické položky nalezené ve významném souboru. Prázdná hodnota je povolena a znamená, že analytická položka nalezená ve významném souboru není potomkem žádné jiné analytické položky nalezené ve významném souboru.

Tabulka	Problem (Problém)
Popis	Problém vytvořený na základě detekce kombinace analytických položek a významných souborů ve zprávě (tuto kombinaci definuje typ problému). Při manuálním zakládání problémů lze problémy vytvářet také na základě běžných souborů a souboru zprávy.
Sloupec	**Popis**
Id	Identifikátor problému. Primární klíč.
Date	Datum vytvoření problému.
CreatedFromMessageFile	Příznak určující zda byl problém vytvořen na základě souboru zprávy. Realizuje vztah problém *je vytvořen* na základě souboru zprávy.
MessageId	Cizí klíč do tabulky Message. Realizuje vztah zpráva *obsahuje* problém.
StatusId	Cizí klíč do tabulky Status. Realizuje vztah problém *se nachází* ve stavu.
ProblemTypeId	Cizí klíč do tabulky ProblemType. Realizuje vztah problém *je instancí* typu problému. Prázdná hodnota je povolena a znamená, že problém není instancí žádného typu problému (možné pouze u manuálně vytvořených problémů).
UserId	Cizí klíč do tabulky User. Realizuje vztah uživatel *vytváří* problém. Prázdná hodnota je povolena a znamená, že problém byl vytvořen automaticky.

Tabulka	Problem_CommonFile
Popis	Vazební tabulka realizující vztah problém *je vytvořen* na základě obyčejného souboru.
Sloupec	**Popis**
ProblemId	Cizí klíč do tabulky Problem. Společně se sloupcem CommonFileId tvoří primární klíč.
CommonFileId	Cizí klíč do tabulky CommonFile. Společně se sloupcem ProblemId tvoří primární klíč.

Tabulka	Problem_AnalyticalItem
Popis	Vazební tabulka realizující vztah problém *je vytvořen* na základě analytické položky.
Sloupec	**Popis**
ProblemId	Cizí klíč do tabulky Problem. Společně se sloupcem AnalyticalItemId tvoří primární klíč.
AnalyticalItemId	Cizí klíč do tabulky AnalyticalItem. Společně se sloupcem ProblemId tvoří primární klíč.

Tabulka	Problem_ImportantFile
Popis	Vazební tabulka realizující vztah problém *je vytvořen* na základě významného souboru.
Sloupec	**Popis**
ProblemId	Cizí klíč do tabulky Problem. Společně se sloupcem ImportantFileId tvoří primární klíč.
ImportantFileId	Cizí klíč do tabulky CommonFileId. Společně se sloupcem MessageId tvoří primární klíč.
CreatesProblem	Příznak určující zda soubor identifikovaný hodnotou ImportantFileId skutečně participoval na vytvoření problému (pak má příznak hodnotu 1), nebo byl do této tabulky vložen, protože se v něm našla analytická položka, na jejímž základě byl problém vytvořen (pak má příznak hodnotu 0).

Tabulka	Status (Stav)
Popis	Stav problému.
Sloupec	**Popis**
Id	Identifikátor stavu problému. Primární klíč.
Name	Název stavu problému.
Abbreviation	Zkratka stavu problému.
Description	Popis stavu problému. Prázdná hodnota je povolena.
IsSystem	Příznak určující zda se jedná o systémový stav problému, který nelze z databáze odstranit.

Tabulka	BugTrackingReference (Odkaz na chybu)
Popis	Stav problému.
Sloupec	**Popis**
Id	Identifikátor odkazu na chybu. Primární klíč.
BugId	Identifikátor chyby v systému pro sledování chyb.
BugDescription	Popis chyby v systému pro sledování chyb. Prázdná hodnota je povolena.
BugStatus	Stav chyby v systému pro sledování chyb. Prázdná hodnota je povolena.

Tabulka	Problem_BugTrackingReference
Popis	Vazební tabulka realizující vztah problém *má nastaven* odkaz na chybu v systému pro sledování chyb.
Sloupec	**Popis**
ProblemId	Cizí klíč do tabulky Problem. Společně se sloupcem BugTrackingReferenceId tvoří primární klíč.
BugTrackingReferenceId	Cizí klíč do tabulky BugTrackingReference. Společně se sloupcem ProblemId tvoří primární klíč.

Tabulka	Comment (Komentář)
Popis	Komentář blíže popisující stav řešení problému.
Sloupec	**Popis**
Id	Identifikátor komentáře. Primární klíč.
Text	Text komentáře.
Date	Datum vytvoření komentáře.
ProblemId	Cizí klíč do tabulky Problem. Realizuje vztah komentář *popisuje* problém.
ParentCommentId	Cizí klíč do tabulky Comment. Realizuje vztah komentář *reaguje* na jiný komentář. Prázdná hodnota je povolena a znamená, že komentář nereaguje na žádný jiný komentář.
UserId	Cizí klíč do tabulky User. Realizuje vztah uživatel *vytváří* komentář.

Tabulka	User (Uživatel)
Popis	Uživatel centrály.
Sloupec	**Popis**
Id	Identifikátor uživatele. Primární klíč.
Login	Přihlašovací jméno uživatele.
PasswordHash	Hash hesla spočítaný hashovací funkcí SHA-1.
PasswordSalt	Sůl hesla.
Name	Jméno uživatele.
Surname	Příjmení uživatele.
Email	Email uživatele.
IsActive	Příznak určující zda je uživatel aktivním uživatelem pracujícím s centrálou (neaktivní uživatelé již nemají do systému povolen přístup).
RoleId	Cizí klíč do tabulky Role. Realizuje vztah uživatel *vlastní* roli.

Tabulka	Role (Role)
Popis	Role uživatele.
Sloupec	**Popis**
Id	Identifikátor role. Primární klíč.
Name	Název role.
Description	Popis role. Prázdná hodnota je povolena

Tabulka	Permission (Oprávnění)
Popis	Oprávnění k provedení uživatelské akce.
Sloupec	**Popis**
Id	Identifikátor oprávnění. Primární klíč.
Name	Název oprávnění.
Description	Popis oprávnění.
IsLogged	Příznak určující zda má být uživatelská akce oprávnění protokolována do tabulky UserAction.

Tabulka	UserAction (Uživatelská akce)
Popis	Záznam o provedení uživatelské akce.
Sloupec	**Popis**
Id	Identifikátor záznamu o provedení uživatelské akce. Primární klíč.
Date	Datum provedení uživatelské akce.
Parameters	Parametry potřebné k uvedení systému do stavu před provedením uživatelské akce.
PermissionId	Cizí klíč do tabulky Permission. Realizuje vztah oprávnění *umožňuje provést* uživatelskou akci.
UserId	Cizí klíč do tabulky User. Realizuje vztah uživatel *provádí* uživatelskou akci.

Příloha B

Výstupní XML soubor toku zpracování

```
<?xml version="1.0" encoding="utf-8" ?>
<Output>
  <UserDescription>
  This is automatically generated text. Please, do not change it.
  ----------------------------------------------------------------
  AVGDIAG-GUID=A6D0E6B0-67A1-4b74-A9F8-F26409A07688.01C8A8A05608EE5E.avgdiag
  AVG Version: 8.0.93
  Virus DB Version: 269.23.0/1381
  AVG License Number: 75A-TH2RX1-P23-CL1-S2ELWV-XF2
  E-mail plugins: eMail-Scanner
  OS Version: Microsoft Windows XP Professional Service Pack 3, v.3264
  Full memory: 991.5 MB
  Free memory: 684.7 MB
  System date: Sun, 27 Apr 2008 21:53:17 +02:00GMT
  Components in error/warning state:
     AntiSpy: Datenbank ist veraltet
     AntiVirus: Datenbank ist veraltet
     UpdateMgr: Datenbankaktualisierung ist deaktiviert.
  ----------------------------------------------------------------
  </UserDescription>
  <ImportantFile>
    <Name>285250-mail.emd</Name>
    <Size>1725</Size>
    <Hash>4FDFD82D8531FD5735A9C50D8AC7B7A0FFA8596B</Hash>
    <TypeId>5</TypeId>
    <AnalyticalItem>
      <Value>crashdiag@avg.com</Value>
      <TypeId>1</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>A6D0E6B0-67A1-4b74-A9F8-F26409A07688.01C8A8A05608EE5E.avgdiag</Value>
      <TypeId>2</TypeId>
    </AnalyticalItem>
  </ImportantFile>
  <ImportantFile>
  <Name>env_info.xml</Name>
    <Size>786108</Size>
    <Hash>6D6B9F4D3A9D57527713E79A2953877310F3AE4B</Hash>
    <TypeId>4</TypeId>
 </ImportantFile>
 <ImportantFile>
   <Name>wd_info.xml</Name>
   <Size>57022</Size>
   <Hash>CD3AA39944FF7A79CFB7DE89B84FDF2D2203FCC1</Hash>
   <TypeId>8</TypeId>
 </ImportantFile>
 <ImportantFile>
   <Name>decoded_logs.zip</Name>
   <Size>38663</Size>
   <Hash>9BC4BD08777E450547EE212CB4B914A46766B3D8</Hash>
   <TypeId>12</TypeId>
 </ImportantFile>
 <ImportantFile>
   <Name>history.txt</Name>
   <Size>2607</Size>
   <Hash>73787FF3E299087920D853E754CDF0262DB01A73</Hash>
   <TypeId>15</TypeId>
 </ImportantFile>
 <ImportantFile>
   <Name>AVG_devices.txt</Name>
   <Size>731</Size>
   <Hash>ACD87097E7499D5D47D4A92422E3AC0B4EA1F799</Hash>
    <TypeId>16</TypeId>
 </ImportantFile>
```

```
<ImportantFile>
  <Name>AVG_services.txt</Name>
  <Size>1770</Size>
  <Hash>A72EA92E322EC4312CF376C9FECE715D7BD674F8</Hash>
  <TypeId>17</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>Uninstall.txt</Name>
  <Size>2109</Size>
  <Hash>351D17514B1930E10CD2C78F084685819F4FCF01</Hash>
  <TypeId>19</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>emc_last1000.txt</Name>
  <Size>5574</Size>
  <Hash>40A1D891F8963004C23DF2C10EB9B9FF5BB640F0</Hash>
  <TypeId>20</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>Processes.txt</Name>
  <Size>2684</Size>
  <Hash>3AE84652E7F89B8635C64884EBAC16508E909A79</Hash>
  <TypeId>22</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>Processes.ada</Name>
  <Size>35294</Size>
  <Hash>FC40256E162F5824E708C17750EF7B72589758CF</Hash>
  <TypeId>23</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>avg8inst.log</Name>
  <Size>148726</Size>
  <Hash>6649DF68CA394CAA3D26C374F1634B9EC8B8C330</Hash>
  <TypeId>24</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>Services.ada</Name>
  <Size>25094</Size>
  <Hash>6548B31469D3BE6CFD537572CD9482139F9F9B62</Hash>
  <TypeId>25</TypeId>
</ImportantFile>
<ImportantFile>
  <Name>avg_info.xml</Name>
  <Size>8786</Size>
  <Hash>43AA74A3C053C92766A5A10ED6CA321C53EB701C</Hash>
  <TypeId>1</TypeId>
  <AnalyticalItem>
    <Value>8.0.93</Value>
    <TypeId>3</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>269.23.0/1381</Value>
    <TypeId>4</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>75P-TH1RM1-P03-C01-S2FLWI-XFC</Value>
    <TypeId>5</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>AVG Anti-Virus</Value>
    <TypeId>6</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>Voll</Value>
    <TypeId>7</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>8.0.0.81</Value>
    <TypeId>8</TypeId>
  </AnalyticalItem>
  <AnalyticalItem>
    <Value>16.04.2008, 09:34</Value>
    <TypeId>27</TypeId>
  </AnalyticalItem>
```

```
    <AnalyticalItem>
      <Value>Microsoft Outlook</Value>
      <TypeId>28</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>C:\Programme\AVG\AVG8</Value>
      <TypeId>29</TypeId>
    </AnalyticalItem>
  </ImportantFile>
  <ImportantFile>
    <Name>sys_info.xml</Name>
    <Size>7822</Size>
    <Hash>86F4556ACACDA500287DD50DEF03A5C19595C46E</Hash>
    <TypeId>3</TypeId>
    <AnalyticalItem>
      <Value> Microsoft Windows XP Professional Service Pack 3, v.3264</Value>
      <TypeId>9</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>2008.04.27</Value>
      <TypeId>23</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>19:53:11</Value>
      <TypeId>24</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>991.5 MB</Value>
      <TypeId>25</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>700.6 MB</Value>
      <TypeId>26</TypeId>
    </AnalyticalItem>
  </ImportantFile>
  <ImportantFile>
    <Name>avgupd.exe_128537919562656250.dmp</Name>
    <Size>201716</Size>
    <Hash>6B54B1CE12CE0F6E810C46FD8D653BBD0FF3A1FE</Hash>
    <TypeId>6</TypeId>
    <ImportantFile>
      <Name>avgupd.exe_128537919562656250.dmplog</Name>
      <Size>13175</Size>
      <Hash>AAD6271CDD6C5ED018F4AB78A070FB0FADD80D4B</Hash>
      <TypeId>7</TypeId>
      <AnalyticalItem>
        <Value>User Mini Dump File</Value>
        <TypeId>10</TypeId>
      </AnalyticalItem>
     <AnalyticalItem>
       <Value>0 days 0:03:48.140</Value>
       <TypeId>12</TypeId>
     </AnalyticalItem>
     <AnalyticalItem>
       <Value>0 days 0:00:17.000</Value>
       <TypeId>13</TypeId>
     </AnalyticalItem>
     <AnalyticalItem>
       <Value>avgupd.exe</Value>
       <TypeId>14</TypeId>
     </AnalyticalItem>
     <AnalyticalItem>
       <Value>avginet!Curl_readwrite+13f0</Value>
       <TypeId>18</TypeId>
     </AnalyticalItem>
    <AnalyticalItem>
      <Value>Update (Michal Konig)</Value>
      <TypeId>15</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>avginet</Value>
      <TypeId>16</TypeId>
    </AnalyticalItem>
    <AnalyticalItem>
      <Value>avginet.dll</Value>
```

```
        <TypeId>17</TypeId>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>APPLICATION_FAULT_INVALID_POINTER_READ_avginet!Curl_readwrite+13f0</Value>
        <TypeId>19</TypeId>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>avgupd.exe</Value>
        <TypeId>20</TypeId>
        <AnalyticalItem>
          <Value>Mon Feb 25 12:51:17 2008 (47C2ABB5)</Value>
          <TypeId>21</TypeId>
        </AnalyticalItem>
        <AnalyticalItem>
          <Value>8.0.0.80</Value>
          <TypeId>22</TypeId>
        </AnalyticalItem>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>avgcfgx.dll</Value>
        <TypeId>20</TypeId>
        <AnalyticalItem>
          <Value>Mon Mar 10 20:20:10 2008 (47D589EA)</Value>
          <TypeId>21</TypeId>
        </AnalyticalItem>
        <AnalyticalItem>
          <Value>8.0.0.86</Value>
          <TypeId>22</TypeId>
        </AnalyticalItem>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>avgupd.dll</Value>
        <TypeId>20</TypeId>
        <AnalyticalItem>
          <Value>Tue Mar 25 11:54:16 2008 (47E8D9D8)</Value>
          <TypeId>21</TypeId>
        </AnalyticalItem>
        <AnalyticalItem>
          <Value>8.0.0.89</Value>
          <TypeId>22</TypeId>
        </AnalyticalItem>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>avginet.dll</Value>
        <TypeId>20</TypeId>
        <AnalyticalItem>
          <Value>Mon Feb 25 15:04:54 2008 (47C2CB06)</Value>
          <TypeId>21</TypeId>
        </AnalyticalItem>
        <AnalyticalItem>
          <Value>8.0.0.80</Value>
          <TypeId>22</TypeId>
        </AnalyticalItem>
      </AnalyticalItem>
      <AnalyticalItem>
        <Value>avglogx.dll</Value>
        <TypeId>20</TypeId>
        <AnalyticalItem>
          <Value>Fri Feb 22 18:59:30 2008 (47BF0D82)</Value>
          <TypeId>21</TypeId>
        </AnalyticalItem>
        <AnalyticalItem>
          <Value>8.0.0.80</Value>
          <TypeId>22</TypeId>
        </AnalyticalItem>
      </AnalyticalItem>
    </ImportantFile>
  </ImportantFile>
</Output>
```