České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Turcaj Patrik

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Detekce anomálního chování distribuovaných webových služeb

Pokyny pro vypracování:

Nastudujte důsledně předchozí diplomovou práci pana Bc. Adama Lysáka. Pro hodnoty z jednotlivých metrik analyzujte možnosti detekce anomálií v chování jednotlivých služeb / počítačů. Navrhněte a implementujte systém detekce anomálií pomocí techniky decision trees. Analyzujte a navrhněte systém pro asistovanou automatizovanou detekci anomálií v chování jednolivých služeb. Pro systém navrhněte webové GUI v technologii ASP.NET, které bude umožňovat správu detekčních mechanizmů systému. Klíčové části systému důsledně otestujte a dokumentujte.

Seznam odborné literatury:

[1] Kaustav Das and Jeff Schneider. 2007. Detecting anomalous records in categorical datasets. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA, 220-229.
[2] George K. Baah, Alexander Gray, and Mary Jean Harrold. 2006. On-line anomaly detection of deployed software: a statistical machine learning approach. In Proceedings of the 3rd international workshop on Software quality assurance (SOQUA '06). ACM, New York, NY, USA, 70-77.
[3] RESTfull .NET, Jon Flanders, ISBN: 978-0-596-51920-9, O"Reilly 2009
[4] Troelsen, Andrew, 2012, Pro C# 5.0 and the .NET 4.5 Framework. Apress, ISBN: 1430242337

Vedoucí: Ing. Martin Mudra

Platnost zadání do konce letního semestru 2017/2018

[redacted]

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 6.2.2017

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práca

Detekcia anomálneho chovania distribuovaných webových služieb

Bc. Patrik Turcaj

Vedúci práce: Ing. Martin Mudra

25. mája 2017

Poďakovanie

Ďakujem svojmu vedúcemu diplomovej práce Ing. Martinovi Mudrovi za všetky poskytnuté rady na konzultáciách, ktoré často prebiehali aj v jeho voľnom čase.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 25. mája 2017 .....

Odkaz na túto prácu

Turcaj, Patrik. *Detekcia anomálneho chovania distribuovaných webových služieb.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017.

Abstrakt

Táto diplomová práca sa zaoberá návrhom a implementáciou systému na detekciu anomálneho chovania distribuovaných služieb. Práca nadväzuje na existujúci systém monitoringu škálovateľných služieb, z ktorého bude využívať namerané dáta o monitorovaných službách. Cieľom je vytvoriť asistovano automatický systém, ktorý bude za použitia strojového učenia upozorňovať užívateľa na anomálie v chovaní monitorovaných služieb.

Klíčová slova detekcia anomálií, rozhodovacie stromy, monitoring služieb, strojové učenie

Abstract

This diploma thesis deals with design and implementation of system for anomally detection in distributed service behavior. This work extends existing system for monitoring scalable services, wich provides data of measured metrics of services. Main goal is to create asisted automatic system, wich will be using machine learning to detect anomalies in monitored services behavior and notify user about such anomalies.

Keywords anomally detection, decision trees, service monitoring, machine learning

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V období posledných rokov sa stalo používanie cloudových služieb veľmi populárne. Rada spoločností upustila od nasadenia svojich produktov na vlastnom hardware a prešla na nasadenie do cloudového prostredia. Motiváciou k nasadeniu svojich služieb môžu byť predovšetkým náklady, ktoré môžu byť výrazne nižšie ako pri nasadení na vlastný hardware.

Cloudové prostredie poskytuje dynamické priradenie prostriedkov na základe aktuálneho vyťaženia nasadenej služby, čo znamená že v čase keď službu používa maximálny počet klientov, sú jej priradené dostatočné prostriedky pre obslúženie tohto počtu klientov. Naopak v čase, keď nie je služba využívaná vôbec, beží služba iba v jednej, niekedy dokonca v žiadnej inštancií až pokým nepríde požiadavok od klienta. Vďaka tomuto riešeniu platí zákazník iba za prostriedky, ktoré naozaj využíva a nemusí platiť za prípadný nevyužitý hardware.

Vzhľadom k narastajúcej popularite cloudových služieb kde sa dynamicky mení počet vytvorených inštancií je zaujímavým problémom detekcia správania nasadených služieb. Pri prípadnom útoku, ktorý by extrémne vyťažoval nasadenú službu, môže cloudové prostredie riešiť situáciu vytváraním nových inštancií, čo by malo negatívny dopad na náklady zákazníka. Riešením by mohla byť aplikácia monitorujúca toto správanie.

Cieľ práce

Táto práca sa zaoberá návrhom a tvorbou systému detekciu anomálií v chovaní v distribuovaných službách, pričom naväzuje na už existujúci projekt Adama Lysáka zameraný na monitorovanie a správu služieb. Pri monitorovaní distribuovaných systémov, aplikácií, webových služieb a iných rôznych systémov môže byť úloha monitorovania správania systému náročná. Motiváciou k detekcií anomálneho chovania v distribuovaných službách je upozorniť užívateľa na neželané správanie služieb a ich inštancií, za ktorými môžu stáť chyby v

implementácií, útoky na služby a iné.

Cieľom vytváraného systému je zefektívniť monitorovanie anomálií v chovaní jednotlivých služieb, pričom za pomoci techniky rozhodovacích stromov bude systém riešiť asistovanú automatizovanú detekciu anomálií. Dáta budú následne vyhodnotené a zobrazené koncovému užívateľovi, pričom užívateľ bude mať možnosť konfigurovať rozhodovací strom, prípadne zobraziť vytvorený rozhodovací strom pre účely zobrazenia medzných hodnôt.

V rámci tejto práce budú spomínané názvy implementovaných tried programovacieho jazyka a tieto názvy nebudú prekladané do slovenského jazyka.

Kapitola 1

Analýza

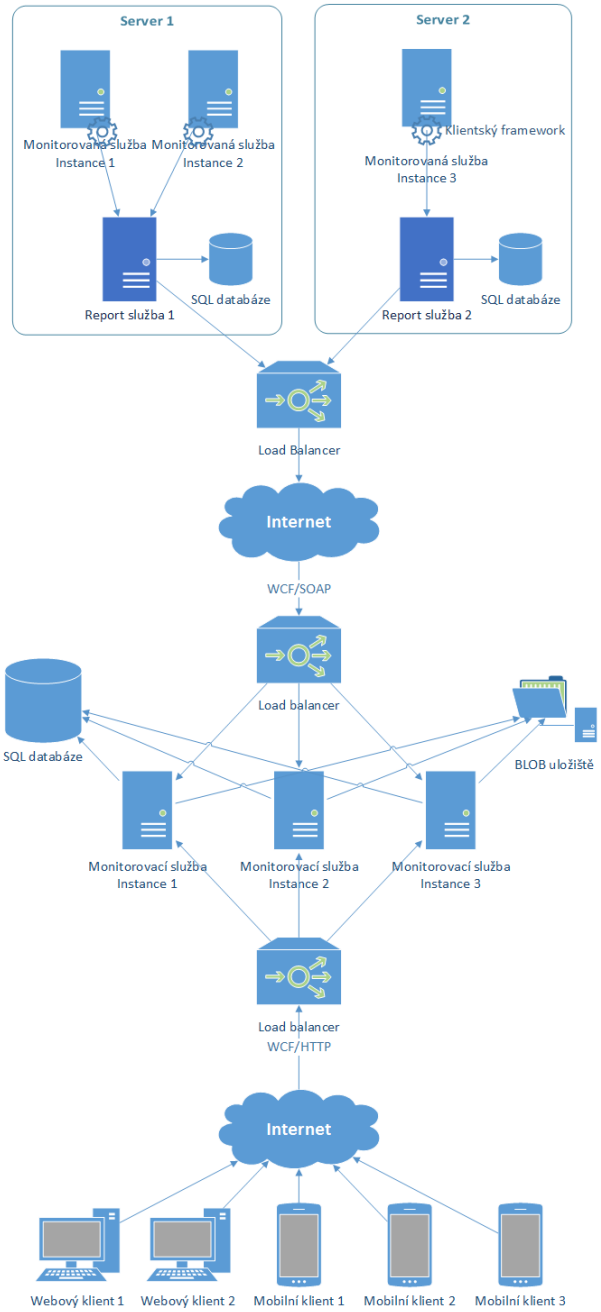
Súčasťou tejto kapitoly je analýza existujúceho projektu pre správu a monitoring škálovateľných služieb, z ktorej bude vytváraný systém využívať dáta. Ďalej bude popísaná analýza problému, ktorý má vytváraný systém riešiť a analýza problematiky rozhodovacích stromov. Nakoniec bude v kapitole obsiahnutá analýza technológií použiteľných pri implementácií vytváraného systému. Záverom budú uvedené funkčné a nefunkčné požiadavky kladené na vytváraný systém.

1.1 Analýza existujúceho riešenia pre monitoring služieb

Pre tvorbu riešenia problému detekcie anomálií budú využívané dáta z už existujúceho systému slúžiaceho na monitoring a konfiguráciu škálovateľných služieb od autora Adama Lysáka[1], ktorý bol implementovaný ako diplomová práca. Je nevyhnutné analyzovať možnosti detekcie anomálií v chovaní jednotlivých služieb pre hodnoty metrík monitorované v tomto projekte.

Cieľom existujúceho projektu bolo navrhnúť a implementovať systém umožňujúci konfiguráciu a monitorovanie horizontálne škálovateľných služieb a ich inštancií. Tento systém umožňuje spravovať a monitorovať metriky služieb, ktoré môžu byť spustené vo viacerých inštanciách a na rôznych fyzických zariadeniach. Systém je implementovaný na platforme Microsoft .NET a je zložený z viacerých projektov, ktorými sú Management service, Report service, Client framework, Mobilní klienti a Webový klient(Obr. 1.1).

1.1.1 Architektúra existujúceho systému pre monitoring služieb



Obr. 1.1: Architektúra existujúceho systému pre monitoring služieb. Horizontálne škálovateľné časti systému sú zobrazené viacnásobným použitím v grafe[1].

1.1.2 Služby v stávajúcom systéme pre monitoring služieb

Za účelom návrhu vytváraného systému pre detekciu anomálií je potreba analyzovať architektúru stávajúceho systému a rovnako aj jeho jednotlivé služby. Vytváraný systém bude využívať dáta vytvárané stávajúcim systémom a preto je potrebné zhodnotiť tieto vytvárané dáta, ich formu, spôsob uloženia a možné prístupy k dátam.

1.1.2.1 Report service

Report service je Windows service monitorujúca metriky jednotlivých inštancií. Služba je nasadená na rovnakom serveri ako inštancie monitorovanej služby a jej úlohou je správa a monitorovanie inštancií, rovnako ako monitorovanie spoločných hardwardových metrík pre všetky inštancie[1]. Služba po prekročení určitého počtu záznamov zasiela balíky nameraných hodnôt metrík do Management service a preto nie je potrebné aby každá inštancia posielala namerané dáta samostatne[1]. Z tohto dôvodu potrebuje Report service vlastné perzistentné úložisko pre namerané dáta. Týmto úložiskom je SQLite[1] databáza, ktorú je možné použiť bez potreby inštalácie akéhokoľvek ďalšieho softwaru[2]. Metriky, ktoré služba zasiela sú:

Vyťaženie CPU Vyťaženie CPU prostredia, na ktorom beží monitorovaná služba

Využitie pamäti Využitie fyzickej/virtuálnej pamäti prostredia, na ktorom beží monitorovaná služba

Počet bežiacich vlákien počet bežiacich vlákien systému, na ktorom beží monitorovaná služba

Počet bežiacich procesov počet bežiacich procesov systému, na ktorom beží monitorovaná služba

Doba odpovedi časový interval od odoslania požiadavku na službu až po prijatie odpovedi

Disk I/O využitie pevných diskov v systéme

Log služba naviac monitoruje aj metriku typu Log, ktorá reprezentuje logy monitorovanej služby. Tieto log záznamy však nemajú numerickú povahu a obsahujú iba informáciu o debug leveli záznamu a správu záznamu. Log záznamy nie sú odosielané Management service periodicky ako numerické metriky, ale iba v prípade ich výskytu.

[1] Lightweight databáza. Domovská stránka: `https://sqlite.org/`

Report service monitoruje všetky metriky pre každú inštanciu služby zvlášť. Služba však zároveň umožňuje pracovať s ucelenými metrikami jednotlivých služieb a to tým spôsobom, že prepočítava všetky namerané dáta metrík inštancií do tzv. agregovaných metrík a tak vznikajú ucelené dáta pre služby[1]. Výpočet ucelených metrík je riešený aritmetickým priemerom nameraných hodnôt metrík inštancií za určitý čas.

1.1.2.2 Management service

Management service je jedna z hlavných serverových častí systému, implementovaná ako webová služba slúžiaca na pripojenie klientskych aplikácií a monitorovaných služieb. Služba vystavuje REST rozhranie, ktoré umožňuje pripojenie klientskych webových a mobilných aplikácií. Ďalej služba vystavuje SOAP službu ktorá poskytuje metódy pre pripojenie Report service. Služba je prepojená s BLOB úložiskom a lokálnou databázou v ktorej sú uchovávané všetky namerané metriky a logy. V relačnej databáze môže v priebehu času vznikať obrovské množstvo dát pochádzajúcich z merateľných metrík. Dáta sú periodicky po určitom čase presúvané z perzistencie do BLOB úložiska. Tento čas je nastaviteľný v konfiguračnom súbore aplikácie. Dáta sú v prípade ich potreby v intervaloch načítané späť do relačnej databázy.

1.1.3 Klienti v stávajúcom systéme pre monitoring služieb

Na interakciu používateľov systému so službami boli v existujúcom systéme vytvorený mobilný a webový klient.

Webový klient Webový klient je implementovaný v technológií ASP.NET MVC. Slúži na monitorovanie metrík a ich inštancií, umožňuje vytvárať nové monitorované služby, meniť stav služieb a inštancií a meniť ich konfiguráciu.

Mobilný klient Mobilní klienti ponúkajú monitorovanie nameraných metrík služieb a ich inštancií. Mobilné aplikácie boli vytvorené pre platformy Android, Windows Phone aj iOS.

1.1.4 Databáza

Stávajúci systém vďaka abstrakcií umožňuje použitie rôznych typov relačných databáz, na vývoj bola použitá databáza Microsoft SQL Server. Manipulácia s databázou prebiehala pomocou technológie Object-relational mapping (ORM), pri ktorej sú databázové záznamy premapované na objekty v programovacom jazyku[3]. ORM bola implementovaná pomocou frameworku NHibernate[4]. Hodnoty metrík (CPU, RAM...) sú ukladané do databázovej tabuľky MetricType.

Databázová tabuľka CustomMetricType ukladá definície všetkých metrík pridaných užívateľom. Vzťah medzi inštanciou služby a monitorovanou metrikou je uložený v tabuľke MeasuredMetric. Namerané hodnoty dát sa nachádzajú v tabuľkách MetricDataSample, NumericMeasuredMetric a LogMeasuredMetric, pričom tabuľky definujú aj dátový typ meranej hodnoty metriky. Databázové tabuľky ReportService a HardwareConfiguration obsahujú dáta o prihlásených Report službách a hardwarových konfiguráciách systému na ktorom je služba spustená. Hodnoty konfigurácií služieb alebo jednotlivých inštancií sú ukladané to databázových tabuliek Configuration, ConfigurationItem, IntegerConfigurationItem, FloatConfiguationItem, DateConfigurationItem a TextConfigurationItem. Poslednou tabuľkou, ktorá ukladá odkazy na zálohované dáta metrík v BLOB úložisku je databázová tabuľka MetricsDataSamplesBackup.

1.1.4.1 Zasielanie nameraných dát

Systém monitoruje horizontálne škálovateľné služby, ktoré spravidla bežia vo viacerých inštanciách[1]. Tu vzniká problém ako poslať požiadavku na konkrétnu inštanciu. V škálovateľných webových službách sa používa technika nazývaná load balancing, kde medzi serverovou časťou a monitorovanými inštanciami existuje zariadenie load balancer, ktoré sa stará o distribúciu prichádzajúcich požiadaviek od serverovej časti medzi jednotlivé inštancie[5].

Existujúci systém využíva techniku load balancing kde je medzi serverovou časťou a ostatnými službami zaradená komponenta load balancer[1]. Hlavným dôvodom je zabránenie preťažovania niektorej z inštancií zatiaľ čo iné inštancie hladujú[5]. Z princípu fungovania load balanceru je samozrejmé, že priama komunikácia serverovej časti s konkrétnou monitorovanou inštanciou nie je možná[5]. Stávajúci systém tento problém rieši tak, že komunikácia bude prebiehať opačným smerom, teda jednotlivé inštancie budú posielať správy serverovej časti[1].

1.2 Problém detekcie anomálneho chovania

Navrhovaný systém detekcie anomálií bude nadväzovať na existujúci systém pre monitoring škálovateľných služieb(Kapitola 1.1) takým spôsobom, že bude okrem monitorovania a konfigurácie služieb rozšírený o detekciu neobvyklého chovania služieb. Pod neobvyklým chovaním služieb je možné si predstaviť vysokú vyťaženosť niektorej z inštancií, prípadne vysoké namerané hodnoty niektorej z monitorovaných metrík[6]. Za týmito vysokými hodnotami môže byť napríklad chyba v jednej z inštancií, útok na službu ale aj údržba alebo testovanie systému[7].

Vysoké namerané hodnoty je možné rozdeliť do dvoch skupín a to na akceptovateľné, ako napríklad testovanie systému, a na také, o ktorých je

treba informovať užívateľa, do ktorých patrí napríklad útok na službu. Cieľom implementovaného systému bude odhaliť toto neobvyklé správanie a upozorniť na neho užívateľa, pričom sa systém pokúsi naučiť, ktoré z hodnôt sú pre užívateľa akceptovateľné a bude ho upozorňovať len na tie, ktoré akceptovateľné nie sú.

Systém nebude môcť užívateľa upozorniť vždy keď sa v systéme vyskytnú vysoké namerané hodnoty metrík. Je potrebné aby systém po nameraní vysokých hodnôt metrík vedel rozpoznať či sú namerané hodnoty vysoké z dôvodu údržby systému alebo z dôvodu neželanej udalosti akou môže byť útok na službu. Za týmto účelom budú musieť byť vo vytváranom systéme implementované nové dopočítavané metriky, ktoré budú slúžiť na definovanie periodicky sa opakujúcich udalostí naplánovaných užívateľom, pri ktorých sú vysoké hodnoty metrík predpokladané, ako napríklad správa systému.

1.2.1 Dopočítavané metriky

Pri detekovaní anomálneho chovania len na základe nameraných hodnôt metrík by sa vyskytoval problém, že by systém nevedel rozpoznať kedy sú vysoké hodnoty nameraných metrík tolerované a kedy sú nežiadané. Je zrejmé, že keď na monitorovanej službu prebehne útok, budú namerané hodnoty vysoké[7]. Ak by sa systém učil iba na základe hodnôt z týchto metrík tak by dochádzalo k upozorňovaniu užívateľa vždy, keď by sa v systéme vyskytli vysoké hodnoty metrík. Tu nastáva problém, pretože ďalší výskyt vysokých nameraných hodnôt nemusí automaticky znamenať útok na službu, ale môže sa napríklad jednať o pravidelnú údržbu systému[8], kde sú vysoké namerané hodnoty tolerované.

Za účelom čo najviac minimalizovať takéto falošné hlásenie zo systému, bude potreba zaviesť nové metriky, ktoré by užívateľovi umožnili informovať systém o prípadnej plánovanej udalosti pri ktorej sa predpokladajú vysoké hodnoty metrík. Udalosti ako testovanie alebo údržba systému sa vyskytujú mnohokrát periodicky (prvý deň v mesiaci, každú sobotu...) bude vhodné ako nové dopočítavané metriky zaviesť Deň v týždni a Deň v mesiaci. Tieto metriky zabezpečia riešenie problému s periodicky sa opakujúcimi sa udalosťami, ktoré pre užívateľa nemusia byť zaujímavé.

1.2.2 Využitie nameraných dát

Vytváraný systém bude využívať dáta namerané pomocou Report service implementovanej v stávajúcom systéme. Report service monitoruje všetky metriky pre každú inštanciu služby zvlášť[1]. Okrem toho že sú dáta merané pre každú inštanciu samostatne, stávajúci systém tieto dáta využíva na tvorbu agregovaných metrík pre službu ako celok[1]. Report service teda poskytuje dáta ako pre každú z inštancií, tak aj celkovo pre službu počítaním priemeru nameraných dát metrík inštancií v pevnom časovom intervale.

Dáta prichádzajú do Management service v dávkach v nepravidelných intervaloch v rádoch jednotiek sekúnd. Prijímaná dávka dát obsahuje jednu alebo niekoľko objektov reprezentujúcich merané metriky. Objekt reprezentujúci metriku obsahuje okrem nameranej numerickej hodnoty ešte typ meranej metriky, čas kedy bola hodnota nameraná a informácie o inštancií služby pre, ktorú bola hodnota nameraná. Tieto dáta sú následne uložené v perzistencií[1].

Tieto dáta budú ďalej využité pre analýzu a tvorbu rozhodovacieho stromu. Bude potreba zvážiť možnosti, či bude rozhodovací strom vytváraný pre službu ako celok, pre každú z jej inštancií samostatne alebo oboje uvedené. Vzhľadom k povahe nameraných dát, ktoré sú merané pre jednotlivé inštancie a pre službu ako celok sú dopočítavané[1], bude vhodné aplikovať detekciu anomálií na každú z inštancií služby, nie však na službu samotnú. Jedným z dôvodov pre voľbu tohto prístupu je spôsob prípadnej tvorby rozhodovacieho stromu pre službu ako celok.

Jednou možnosťou by bolo použiť agregované dáta, ktoré však vznikajú ako aritmetický priemer hodnôt pre namerané hodnoty metrík z každej z inštancií služby. Veľmi jednoducho môže nastať situácia, kedy jedna inštancia vykazuje kriticky vysoké namerané hodnoty, avšak aritmetický priemer hodnôt cez všetky hodnoty inštancií z tejto hodnoty urobí hodnotu nekritickú a teda užívateľ nepostrehne toto nežiadúce správanie inštancie, čím by sa stratila pointa celého systému. Aplikovanie detekcie anomálií na každú z inštancií tento problém samozrejme rieši, pretože systém na konkrétnej inštancií vykazujúcej kritické hodnoty detekuje a užívateľa na správanie upozorní.

Ďalšou z možností je využiť namerané hodnoty dát zo všetkých inštancií služby. Nevýhodou tejto možnosti je fakt, že tvorba stromu je časovo náročný proces, kde naviac každá nameraná hodnota dát by bola pri výpočte použitá dva razy, a to pri tvorbe stromu pre konkrétnu inštanciu a druhý krát pri tvorbe stromu pre službu ako celok. Vo finále by sa však táto možnosť okrem jej časovej náročnosti potýkala s rovnakým problémom ako prvá uvedená možnosť, pretože vstupom rozhodovacieho stromu je jedna hodnota každej nameranej hodnoty metriky v konkrétnom čase a bolo by teda z všetkých nameraných hodnôt jedného typu v nameraných v rovnakom čase potreba urobiť aritmetický priemer. Z týchto dôvodov je viditeľné, že aplikovanie detekcie anomálií iba na inštancie služieb je postačujúcim a časovo efektívnejším riešením ako aplikovaním detekcie anomálií na službu ako celok alebo kombináciou týchto riešení. Na základe týchto poznatkov je vhodné aby strojové učenie prebiehalo pre každú z inštancií monitorovaných služieb.

1.2.3 Strojové učenie vytváraného systému

Vo vytváranom systéme bude implementované strojové učenie systému na základe nameraných dát prijímaných od Report service. Pre naučenie budú využívané hodnoty metrík meraných stávajúcim systémom a hodnoty dopočíta-

vaných metrík vytvorených v novo vytváranom systéme. Učenie bude prebiehať pre každú z monitorovaných inštancií a bude implementované technikou rozhodovacích stromov. Technika rozhodovacích stromov je ideálnym riešením pre problém detekcie anomálií v distribuovaných službách[9], pretože poskytuje koncovému užívateľovi ľudsky čitateľné pravidlá vo forme stromového grafu pomocou ktorých môže užívateľ jednoducho určiť za akých podmienok sa menia jednotlivé stavy služby[10].

1.3 Rozhodovacie stromy

1.3.1 Definícia rozhodovacích stromov

Rozhodovací strom je pomocný rozhodovací nástroj používajúci stromový graf, ktorý reprezentuje podmienky, ich dôsledky a pravdepodobnosť dosiahnutia udalosti[10]. Umožňuje zobraziť logický vývoj časovo po sebe nasledujúcich rozhodnutí a udalostí. Medzi výhody rozhodovacích stromov oproti iným metódam strojového učenia[11], akou sú napríklad neurónové siete, patrí názornosť, menšia náročnosť na prípravu dát a možnosť získať pravidlá, ktoré sú v pozadí modelu[9]. Vďaka tomu sú vhodným nástrojom na použitie pri vytváranom systéme pre detekciu anomálií, pretože bude pre užívateľa veľmi jednoduché zistiť v akých špecifických prípadoch sa systém dostáva do konkrétnych stavov[9]. Niektoré aplikácie pracujú s predikciou alebo klasifikáciou dát a nie je potrebné vedieť ako model pracuje a ako vyzerá. V iných prípadoch, ako aj v prípade detekcie anomálií, môže aplikácia vyžadovať práve schopnosť zdôvodniť rozhodnutie nad dátami (výzor stromu, konkrétne pravidlo). Rozhodovacie stromy sú vhodné na oba typy týchto problémov.

Rozhodovací strom je vybudovaný na základe trénovacej množiny, ktorý potom predpovedá hodnotu cieľovej premennej na základe vstupných údajov[12]. Strom je väčšinou reprezentovaný graficky v podobe rastúceho stromového grafu zhora nadol[10]. Hlavnými komponentami rozhodovacieho stromu sú uzly a vetvy. Rozhodovací strom počíta s tromi typmi uzlov – koreň, vnútorný uzol a list. Jednotlivé vnútorné uzly stromu reprezentujú podmienku a každý možný výsledok uzlu je reprezentovaný jednou vetvou[13]. Listy stromu, taktiež nazývané koncové uzly, reprezentujú konečné rozhodnutie. Vo vytváranom systéme by mohli byť uzly reprezentované jednotlivými typmi metrík.

Rozhodovací strom sa používa na klasifikovanie objektov do tried na základe série podmienok zameriavajúcich sa na atribúty objektu, prípadne na predikciu doposiaľ neuskutočnených prípadov[14]. Klasifikáciou sa myslí zaradenie objektu podľa jeho atribútov do niektorej zo skupiny existujúcich tried, teda premenná, ktorú predpovedáme je diskrétna. Vo vytváranom systéme bude potrebné definovať koncové triedy, ktoré budú reprezentovať stavy, do ktorých sa môže systém dostať.

Atribúty objektu môžu byť ľubovoľného typu od binárnych, nominálnych, poradových alebo kvantitatívnych[13]. Rozhodovacie stromy, ktorých cieľové triedy môžu nadobúdať konečnú množinu hodnôt sa nazývajú klasifikačné stromy. Stromové modely, kde výsledná hodnota nie je reprezentovaná triedou ale premennou nadobúdajúcou spojité hodnoty nazývame regresné stromy[13]. Metriky existujúceho systému pre monitoring služieb nadobúdajú spojité numerické hodnoty, vo vytváranom systéme teda bude vznikať regresný strom.

1.3.2 Princíp rozhodovacích stromov

Hlavnými požiadavkami na použitie metódy rozhodovacích stromov sú[9]

- Dostatočne veľká tréningová množina dát pre tvorbu rozhodovacieho stromu
- Preddefinované cieľové triedy, do ktorých môžu dáta spadať – diskrétne - záznam buď patrí alebo nepatrí do danej triedy
- Každý záznam dát musí byť možné vyjadriť pomocou pravidiel s množinou atribútov

Veľmi dôležitým krokom pri tvorbe stromu je určiť kritériá (atribúty), podľa ktorých sa rozdeľujú záznamy v jednotlivých uzloch[13]. Algoritmy na tvorbu stromu všeobecne začínajú s celou množinou a snažia sa vybrať atribúty záznamu, ktoré čo najlepšie rozdeľujú záznamy podľa cieľovej triedy[15]. Algoritmus teda musí prejsť všetky vstupné premenné, pre každú určiť spôsob rozdelenia a vyhodnotiť kvalitu tohto rozdelenia. Spôsob rozdelenia závisí aj na tom, či je premenná diskrétna alebo spojitá[16]. V implementovanom systéme budú tieto atribúty reprezentované jednotlivými typmi metrík meraných v stávajúcom systéme.

V prípade, že premenná je diskrétna, môže algoritmus vytvoriť vetvu pre každú nadobúdanú hodnotu, z čoho však môže vzniknúť zbytočne široký rozhodovací strom[17]. Tieto vetvy je však možné zlúčiť, a to práve na základe toho či majú záznamy podobné rozdelenie vzhľadom na cieľovú triedu[9].

Ak je premenná spojitá, algoritmus musí určiť hranicu, ktorá rozdelí záznamy na dve skupiny. To je možné dosiahnuť napríklad vyskúšaním všetkých nadobudnutých hodnôt premennej a následne určenie najlepšieho rozdelenia.

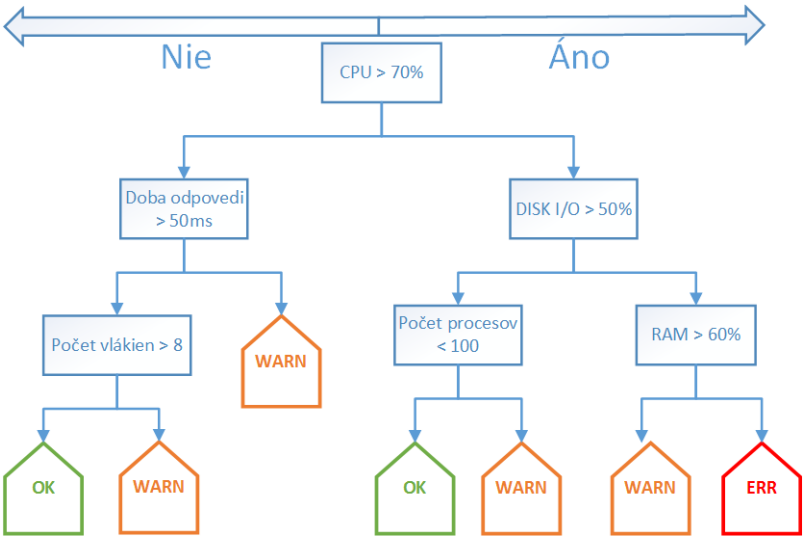
Jedným z prístupov k rozhodovacím stromom je rozdeliť dáta a základe rovnorodosti ich atribútov[17]. Predpokladom pre tvorbu rozhodovacieho stromu je existujúca množina dát, ktorej všetky záznamy boli klasifikované[13]. Táto

tréningová množina dát je využitá na tvorbu klasifikačného modelu – rozhodovacieho stromu. Tento model je následne použitý na klasifikáciu dát, ktoré zatiaľ neboli priradené do tried.

Stromy, ktoré vzniknú automaticky pomocou algoritmu, nemusia mať vždy optimálny tvar vhodný pre efektívnu klasifikáciu[18]. Strom je možné orezávať priamo pri konštrukcii stromu, alebo až po vytvorení celého stromu[16]. Motiváciou k orezávaniu vetví stromu môže byť napríklad dostatočne veľká pravdepodobnosť, že dáta v danej vetve patria do jednej z výsledných tried.

Klasifikácia pomocou rozhodovacieho stromu prebieha cestou záznamu od koreňa stromu až k jeho listu. V každom kroku je záznam otestovaný na podmienku aktuálneho uzlu stromu[9]. Keďže každá možná výsledná hodnota podmienky má k sebe patriacu cestu do ďalšieho uzlu, záznam pokračuje po danej ceste do nasledujúcich uzlov až dôjde do listového uzlu, kde je klasifikovaný triedou. Každá cesta od koreňa stromu až po list môže byť vyjadrená ako pravidlo kde podmienka pravidla je logický súčin všetkých podmienok jednotlivých uzlov po ceste z koreňa do listu[10].

Na obrázku Obr. (Kapitola 1.2) je znázornený jednoduchý príklad rozhodovacieho stromu. Nech existuje množina nemeraných hodnôt metrík systému, kde jeden záznam obsahuje informácie o metrikách ako využitie CPU, RAM, Disk I/O, počet procesov a počet vlákien. Rozhodovací strom takýto záznam vyhodnotí tak, že vyhodnocuje záznam na podmienku v každom z navštívených uzlov, pričom začína v koreňovom uzle a musí prejsť až do jedného z rozhodovacích uzlov. Tu je záznam klasifikovaný jednou z preddefinovaných koncových tried.



Obr. 1.2: Ukážka modelu vytváraného rozhodovacieho stromu meraných metrík pre monitorovanú službu

Každý binárny rozhodovací strom je možné vyjadriť ako výraz v Disjunktívnej Normálnej Forme (DNF)[10]. Takisto akákoľvek funkcia vo výrokovej logike sa dá vyjadriť ako rozhodovací strom[10]. Preto je zo vzniknutého rozhodovacieho stromu veľmi jednoduché určiť, za akých podmienok nastane jedna z koncových udalostí.

Hlavný dôvod pre aplikáciu tejto techniky je jej prehľadnosť a jednoduchá interpretácia, ktorá umožňuje užívateľom ľahko a rýchlo vyhodnocovať získané výsledky, identifikovať kľúčové položky a vyhľadávať zaujímavé segmenty prípadov[11]. Vedia pracovať s kontinuálnymi aj kategorickými premennými a vďaka grafickej reprezentácií stromu umožňujú užívateľovi veľmi prehľadne zobraziť vzťahy medzi premennými pri klasifikácií anomálneho chovania služby. Rozhodovacie stromy nevyžadujú žiadnu špeciálnu prípravu dát. Rozhodovacie stromy poskytujú jasnú indikáciu, ktoré vlastnosti sú dôležité pre predikciu alebo klasifikáciu[9].

1.3.3 Veľkosť tréningovej množiny

Relevancia a počet záznamov v testovacej množine výrazne ovplyvňuje budúcu výkonnosť vytvoreného modelu[13]. Jedným prípadom je preučenie stromu, kedy algoritmus na tvorbu rozhodovacích stromov môže vytvoriť strom, ktorý sám obsahuje viacej štruktúr ako tréningová množina, z čoho vznikne veľmi komplexný strom[18]. Prvou stratégiou, ako predísť tomuto správaniu je orezávanie stromu, ktoré sa deje až po jeho vytvorení[16]. Algoritmus orezávania stromu znovu prejde vytvorený strom a oreže podstromy, ktoré sú považované

sa irelevantné vzhľadom k danému odhadu chyby[13]. Druhou stratégiou je zredukovanie tréningovej množiny ešte pred vytvorením samotného stromu. Jednou z možností ako to dosiahnuť je vymazať menej dôležité záznamy z tréningovej množiny. Jedným z prípadov, kedy sú dáta považované za menej významné je záznam s chýbajúcimi atribútmi. Táto technika vedie k vytvoreniu jednoduchších a presnejších stromov[19].

1.3.4 Konštrukcia rozhodovacieho stromu

Mnoho algoritmov na tvorbu rozhodovacích stromov je variáciou základného hladného algoritmu, ktorý pracuje v priestore možných rozhodovacích stromov[9]. Algoritmy konštruujú rozhodovací strom na základe poskytnutej tréningovej množiny[13]. Algoritmus konštrukcie rozhodovacieho stromu T pozostáva z piatich krokov:

```
Create node T;
T = getTrainingSet();
while(true) {
        if ( whole T set is positive ) {
                create child T;
                end;
        }
        if ( whole T set is negative ) {
                create child T;
                end;
        }
        Choose X attribute and split T to subsets by X;
        for each subset create child T;
}
```

Algoritmus 1.1: Pseudokód algoritmu pre tvorbu rozhodovacieho stromu

Z tejto myšlienky vychádza niekoľko známych algoritmov na tvorbu rozhodovacích stromov. Jednými z prvých a zároveň veľmi často používaných algoritmov pre vytváranie rozhodovacích stromov sú algoritmy ID3 a C4.5 od tvorcu Rossa Quinlana[9]. Pre tvorbu rozhodovacích stromov existuje mnoho algoritmov (CART, C5.0, CHAID a iné), z ktorých niektoré vychádzajú priamo z ID3 alebo C4.5 (C5.0)[9] a preto budú tieto dva algoritmy bližšie analyzované:

Algoritmus ID3

Jedná sa o jeden z prvých známych algoritmov zameraných na tvorbu rozhodovacích stromov. Jeho tvorcom je John Ross Quinlan, ktorý ho publikoval v roku 1975 v knihe Machine Learning[11]. Algoritmus funguje tak, že rozhodovací strom začína ako samostatný koreňový uzol, ktorý obsahuje celú tréningovú množinu. Algoritmus najprv skontroluje, či je možné zaradiť všetky záznamy

zo vstupnej množiny do jednej cieľovej triedy[17]. V prípade že to nie je možné, je potreba zvoliť atribút záznamu, na základe ktorého sa záznamy rozdelia. Pre každú skupinu vzniknutú týmto rozdelením je vytvorený uzol stromu ako potomok aktuálneho uzlu, pričom každá skupina reprezentuje jednu hodnotu testovaného atribútu záznamu[17].

Takto algoritmus skontroluje každý z novo vytvorených uzlov. Algoritmus končí delenie uzlu v prípade, že všetky záznamy prichádzajúce do uzlu je možné zaradiť do jednej triedy. Vtedy je uzol označený ako koncový a je označený príslušnou triedou. Algoritmus končí vtedy, keď sú uzavreté všetky cesty vytvoreného rozhodovacieho stromu[9].

Dôležitým prvkom algoritmu je spôsob, akým je volený testovací atribút na základe ktorého sa uzol rozdelí. V prípade, že by bol testovací atribút vyberaný náhodne, algoritmus by produkoval stromy rôznej zložitosti[9]. Jednoduchšie stromy by podporovali generalizáciu dát, u zložitých stromov by hrozilo riziko preučenia[20]. Takýto strom poskytuje výborné výsledky na testovacej množine, avšak pri aplikovaní na reálne dáta má vysokú mieru chybovosti[10]. V algoritme ID3 je na voľbu testovacieho atribútu použitá štatistická metóda nazývaná informačný zisk, ktorá určuje, ako dobre konkrétny atribút rozdeľuje dáta vzhľadom na konečnú klasifikáciu[17]. Medzi nevýhody algoritmu ID3 patrí predovšetkým obmedzenie na prácu iba s diskrétnymi hodnotami[17], nepodporuje orezávanie stromu[17] a teda oproti C45 častejšie dochádza k preučeniu stromu a nevie sa vysporiadať s chýbajúcimi hodnotami atribútov[9].

Algoritmus C4.5

Algoritmus s názvom C4.5 je rozšírením algoritmu ID3[16]. Publikovaný bol v roku 1993 Rossom Quinlanom[16]. Stal sa pomerne obľúbeným algoritmom pre tvorbu rozhodovacích stromov po tom, ako bol v roku 2008 ocenený prvým miestom v práci Top 10 Algorithms in Data Mining publikovanou vydavateľstvom Springer LNCS[21]. Oproti algoritmu ID3 má algoritmus C4.5 niekoľko vylepšení:

- Spracovanie diskrétnych aj spojitých premenných – aby algoritmus vedel pracovať aj so spojitými hodnotami, zvolí si bod z intervalu a následne podľa neho tieto spojité hodnoty rozdeľuje podľa toho, či sú vyššie alebo nižšie ako tento bod[9]

- Schopnosť spracovať záznamy s chýbajúcimi hodnotami atribútov – algoritmu chýbajúcu hodnotu vynechá pri počítaní entropie a informačného zisku

- Orezávanie vytvoreného stromu – po vytvorení stromu algoritmus spätne prejde strom a snaží sa nahradiť vetvy, ktoré pri výpočte nepomáhajú za listové uzly

Pretože dáta merané v stávajúcom systéme v Report service nadobúdajú spojité hodnoty,je pre vytváraný systém ideálne použiť algoritmus C4.5, ktorý vie s týmito hodnotami pracovať. Pri meraní hodnôt metrík vzniká v systéme veľké množstvo dát[1] a preto je možné predpokladať, že aj vytváraná tréningová množina bude obsahovať stovky hodnôt, z čoho by potencionálne mohol vznikať veľmi komplexný strom. Z toho dôvodu je lepšie použiť algoritmus C4.5, ktorý je schopný orezávať vzniknutý strom a zabraňuje tak preučeniu algoritmu[16].

1.3.5 Nevýhody rozhodovacích stromov

Problém tvorby optimálneho rozhodovacieho stromu je známy ako NP-úplný[13]. Používané algoritmy na tvorbu rozhodovacích stromov sú založené na heuristike ako napríklad spomenutý hladný algoritmus, ktorý robí optimálne riešenia na každej vetve, avšak nevie garantovať že riešenie je optimálne aj globálne pre celý strom[9].

Algoritmus na tvorbu rozhodovacieho stromu môže na základe vstupných dát vytvoriť veľmi komplexný strom, ktorý perfektne rozhoduje vstupné dáta, avšak pri aplikovaní stromu na ostatné dáta vracia nesprávne výsledky[13]. Tento jav je nazývaný preučenie rozhodovacieho stromu. Je možné mu predchádzať použitím metódy prerezávania stromu[16].

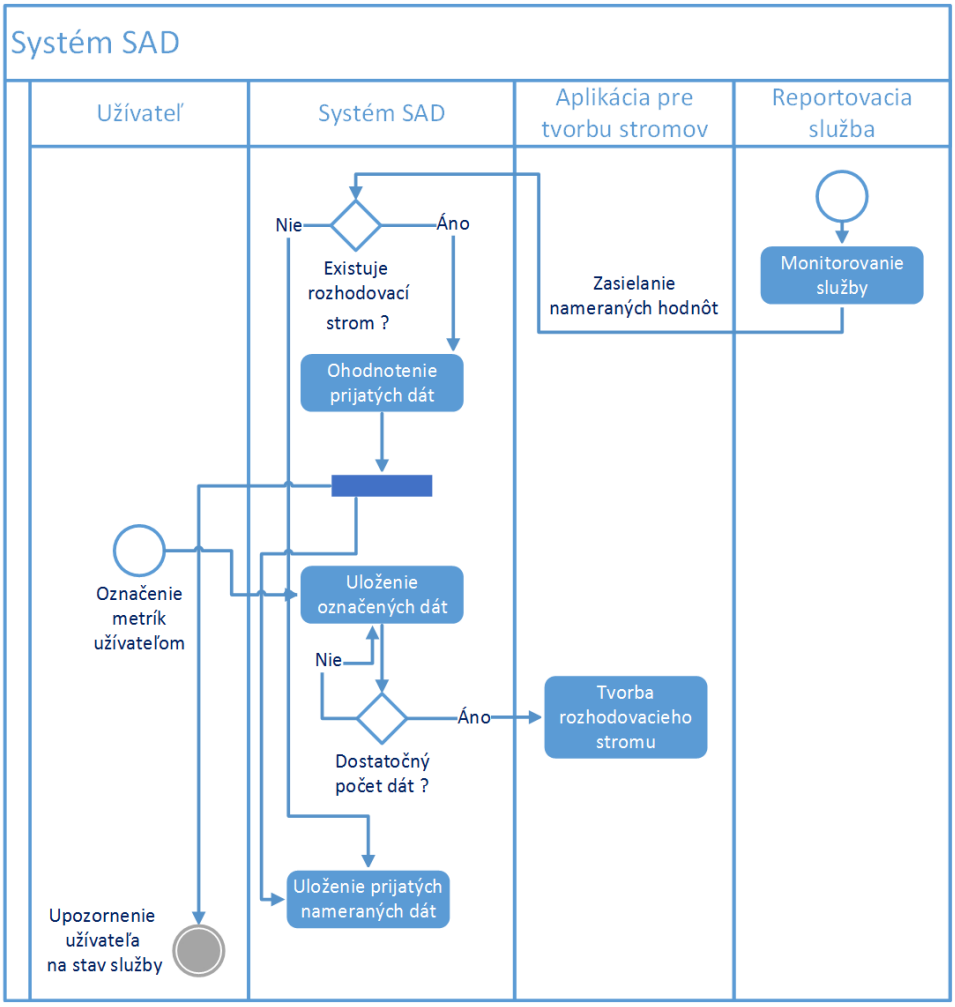
Použitie rozhodovacích stromov nie je robustné. To znamená že aj malá zmena dát môže vyvolať veľkú zmenu vo vytvorenom rozhodovacom strome[10].

1.4 Príprava dát pre tvorbu stromu

Prerekvizitou k tvorbe rozhodovacieho stromu je tréningová množina dát[13]. Množina je vytvorená z hodnôt stromových atribútov, ktoré vo vytváranom systéme budú tvorené meranými a dopočítavanými metrikami. Vstupné dáta musia byť vo forme zoznamu, kde jedna položka zoznamu obsahuje jednu hodnotu požadovaných stromových atribútov a koncovú klasifikačnú triedu. Systém bude implementovaný ako asistovaná automatická detekcia anomálií v chovaní služieb, preto pre každý vstupný záznam bude rozhodnutie o koncovej klasifikačnej triede priraďované užívateľom. V systéme bude potrebné definovať koncové klasifikačné triedy, zobraziť užívateľovi možné vstupné záznamy a povoliť mu klasifikovať tieto záznamy do koncovej klasifikačnej triedy.

Do systému okrem aktuálnej funkcionality zobrazenia hodnôt metrík bude naviac pridaná funkcionalita, ktorá umožní užívateľovi tieto hodnoty rozdeliť podľa času do preddefinovaných tried. Predpokladá sa totiž, že užívateľ ako

správca týchto služieb je oboznámený s prípadnou údržbou systému alebo priemerných vyťažovaním hardwaru a teda vie rozhodnúť, v akom čase je v poriadku keď sú namerané hodnoty metrík vysoké a kedy to už v poriadku nie je. Po tom, čo užívateľ označí dostatočný počet hodnôt, systém tieto dáta spracuje a naučí sa toto rozdelenie, na čo mu slúži technika rozhodovacích stromov(Obr. 1.3). Od určitého bodu bude systém vďaka rozhodovaciemu stromu vedieť namerané hodnoty metrík klasifikovať bez užívateľa a v prípade potreby ho na ne upozorniť.



Obr. 1.3: Sekvenčný diagram zjednodušene popisuje funkcie jednotlivých častí systému pri detekcií anomálií služby

1.4.1 Zobrazenie dát užívateľovi

Vstupná tréningová množina dát bude vo vytváranom systéme vytváraná podľa dát vytvorených užívateľom pomocou webového rozhrania. Vo webovom

rozhraní bude potreba zobraziť užívateľovi namerané dáta metrík pomocou grafových komponent. Grafové komponenty by mali byť vytvorené tak, aby definičným oborom grafu bol čas a obor hodnôt boli možné hodnoty metriky. Keďže jednotlivé metriky majú rôzne obory hodnôt, ktoré sú z pravidla merané v rozličných jednotkách, bude potreba zobraziť každú metriku na samostatnom grafe a všetky tieto grafy bude potreba synchronizovať podľa času. Tým že budú zobrazené dáta synchronizované podľa času, bude užívateľ môcť jednoducho zhodnotiť povahu dát za jeden časový úsek.

Grafové komponenty budú musieť okrem zobrazenia dát umožniť jednoducho priradiť im preddefinovanú koncovú triedu. Tieto komponenty bude potrebné zobraziť jednu pod druhou, pričom užívateľ bude môcť vytvoriť interval v čase cez všetky zobrazené grafy a priradiť tomuto intervalu koncovú triedu(Obr. 2.2). Týmto spôsobom bude postupným vytváraním intervalov vytvorená tréningová množina pre rozhodovací strom.

Namerané hodnoty bude musieť byť možné zobraziť na webovom rozhraní v reálnom čase, rovnako ako aj spätne zadaním časového intervalu. Pretože v systéme bude vytvorená grafová komponenta pre každú zobrazenú metriku, prichádzajúce dáta do grafových komponent budú musieť mať iný formát ako pri príchode z Report service do Management service. Prijaté dáta budú musieť byť vo forme listu hodnôt jedného typu, preto nebudú dáta prijímané priamo z Report service, ktorá dáta posiela vo forma balíku, ktorý obsahuje hodnoty každej z meraných metrík[1], ale budú získavané z perzistencie pre každú meranú metriku samostatne.

Po tom, ako bude vytvorená tréningová množina, bude pri tvorbe stromu potrebné určiť, čo budú jednotlivé stromové atribúty, ktoré budú tvoriť stromové uzly. Jednou možnosťou je pre každý typ metriky prijatý od Report service vytvoriť jeden stromový atribút. To by však znamenalo, že pre každú službu bude vytvorený strom vždy z rovnakých atribútov.

Vhodnejším spôsobom bude zobraziť užívateľovi zoznam možných stromových atribútov. Tieto zoznam atribútov bude vytvorený tak, že pre každý typ metriky prijatý od Report service bude vytvorená jedna položka zoznamu možných atribútov, a užívateľ zvolí pre tvorbu stromu len tie atribúty, ktoré ho zaujímajú. Užívateľovi bude ďalej zobrazený zoznam dopočítavaných metrík, z ktorých si bude môcť takisto zvoliť požadované metriky. Rovnako bude konfigurácia obsahovať, či sa má strom vytvoriť iba na základe novo označených hodnôt metrík alebo má systém pri tvorbe stromu použiť aj dávnejšie namerané hodnoty, z ktorých už bol vytvorený predchádzajúci rozhodovací strom.

1.5 Frameworky pre tvorbu grafových komponent

V systéme na webovom rozhraní nebudú namerané hodnoty dát užívateľovi len jednoducho zobrazené, ale bude potreba vytvoriť graf pre každú zobrazovanú metriku, tieto grafy zobraziť jeden pod druhým a synchronizovať ich podľa času

kedy bola zobrazovaná hodnota nameraná. Graf musí podporovať zobrazenie dát v reálnom čase ako aj zobrazenie dát z minulosti. Ďalej bude musieť byť možné zobrazené dáta označiť, a to nie na každom grafe samostatne, ale cez všetky zobrazené grafy súčasne vytvorením časového intervalu, ideálne ťahom kurzoru po jednom z grafov. Po vytvorení intervalu bude potrebné tomuto intervalu priradiť hodnotu, ktorá reprezentuje preddefinovanú koncovú triedu a túto hodnotu v intervale na grafe zobraziť. V súčasnej dobe existuje rada frameworkov, ktoré umožňujú prácu s grafovými komponentami na grafických rozhraniach.

1.5.1 Anycharts

Je komplexný balík knižníc pre tvorbu grafov pre webové rozhrania pomocou JavaScript[22]. Hlavnými produktami sú:

AnyChart Knižnica so základnou sadou grafových riešení

AnyChart Stock and Financial Charts Grafová vizualizácia finančných riešení. Podpora zobrazenia časovo orientovaných dát, zobrazenie dát v reálnom čase. Vhodné aj pre aplikácie monitorujúce výkon a správu klientov

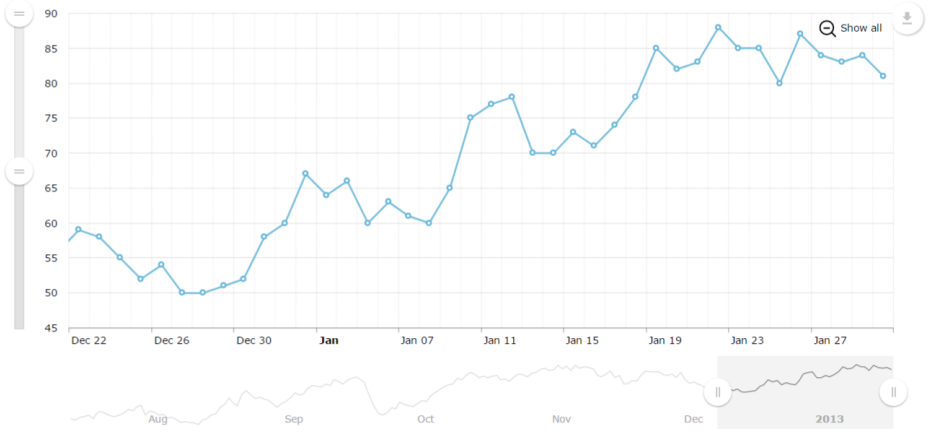
Knižnice sú kompatibilné s jazykmi ako ASP.NET, PHP, Perl a iné a podporujú tvorbu dát z XML, JSON a CSV formátov. Riešenie si zakladá na rýchlosti a efektivite, je schopné vyrendrovať okolo 250000 dátových bodov za sekundu a v časovo zameraných dátach je možné pripojiť 10000 dátových bodov každých 100 milisekúnd[22]. Knižnice vystavujú API kde umožňujú konfiguráciu vytváraných grafov. Riešenie pracuje out of the box, kde vstupom je množina dát a výstupom je vytvorený graf. Knižnice podporujú pridávanie viac ako jednej osy do grafu, možnosť rozšírenia o rôzne grafické témy, exportovanie a tlač grafov a iné. Riešenie AnyCharts je dostupné bezplatne pre nekomerčné a študijné účely, inak je spoplatnené v závislosti na zvolenej verzií odlišujúcej sa v spôsobe využitia a licencii.

1.5.2 AmCharts

Je JavaScriptové riešenie na vizualizáciu dát pomocou grafov. AmCharts [23] ponúka niekoľko rôznych knižníc kde každá poskytuje podporu pre určité typy grafov ako napríklad:

JavaScript Charts Knižnica pre vizualizáciu dát pomocou základných interaktívnych grafov

JavaScript Stock Charts Knižnica pre vizualizáciu a analýzu finančných alebo inak časovo orientovaných dát



Obr. 1.4: Graf pre časové dáta zobrazený pomocou frameworku AmCharts

Grafy je možné vytvárať a konfigurovať dvoma spôsobmi, kde jedna možnosť je pomocou JavaScript objektu a druhá pomocou JSON objektu[23]. Knižnice podporujú vlastnosti ako porovnanie viacerých množín dát, upravenie granularity na základe časového pásma, zobrazenie viacerých grafov v jednej komponente alebo označenie dát podľa časového intervalu. Čo sa týka tvorby grafov, knižnice pracujú podobným spôsobom ako AnyCharts, kde vstupom je množina dát a výstupom je vytvorený graf.

Knižnica Stock Charts je zameraná na časovo orientované dáta, čo je ideálnym riešením pre zobrazenie dát na webovom rozhraní pre vytváraný systém. Vo vytváranom systéme je potrebné synchronizovať dáta pre všetky zobrazené metriky podľa času za účelom rýchleho zhodnotenia stavu systému užívateľom na základe dát v jednom časovom úseku.

Výhodou je možnosť využívať knižnicu bezplatne za podmienky že na každej z vytvorených grafových komponent bude odkaz na webovú stránku AmCharts. Ďalej AmCharts ponúka aj viac levelov platených komerčných verzií, kde sú grafové komponenty zobrazené bez odkazu a odlišujú sa vo veciach ako podpora produktu, vylepšenia verzií, počet zariadení na ktorých je možné zobraziť jeden konkrétny graf, poskytnutie zdrojového kódu a iné.

1.5.3 D3.js

Knižnica D3 [24] pre vizualizáciu dát pomocou grafov nepracuje out of the box ako AmCharts alebo AnyChart, ale prenecháva plnú kontrolu nad tvorbou grafu pre vývojára. Ponúka väčšie možnosti konfigurácie ako AmCharts alebo AnyCharts, avšak vývojár musí vytvoriť každý graf sám pomocou API poskytovaného D3. Výhodou tohto riešenia je veľká možnosť konfigurácie grafov a prenechanie kontroly nad tvorbou grafu na vývojára. Nevýhodou tohto riešenia

môže byť väčšia časová náročnosť pri vytváraní jednotlivých grafov, nepodporovanie IE8 a rendrovanie dát iba v SVG (Scalable Vector Graphics), ktorá v niektorých prehliadačoch nemusí byť podporovaná. D3.js je licencované pod BSD licenciou [25].

1.6 Tvorba rozhodovacieho stromu

Po tom, ako užívateľ označil namerané hodnoty metrík, bude mať systém všetky potrebné náležitosti k vytvoreniu tréningovej množiny a rozhodovacieho stromu. Z dôvodu časovej náročnosti tvorby rozhodovacích stromov musí byť tento proces navrhnutý a implementovaný tak, aby bol jednoducho horizontálne škálovatelný. Prerekvizitou k tvorbe rozhodovacieho stromu je tvorba vstupná množina dát[13]. Pretože tvorba tréningovej množiny je takisto časovo náročný proces, bude tento proces rovnako horizontálne škálovateľný.

Systém spracuje prijaté intervaly od užívateľa a vytvorí tréningovú množinu pre vytváraný rozhodovací strom podľa dát z poslednej uloženej stromovej konfigurácie. Táto vstupná množina bude slúžiť ako jeden zo vstupov pre vytváraný rozhodovací strom. Po vytvorení rozhodovacieho stromu bude tento strom uložený do perzistencie. Tvorba rozhodovacieho stromu bude prebiehať za pomoci frameworku strojového učenia a to z niekoľkých dôvodov:

Náročnosť implementácie Implementovať algoritmus pre tvorbu rozhodovacích stromov, ktorý by podporoval spojité hodnoty, ktoré je potrebné vo vytváranom systéme spracovávať, a implementoval metódy, ktoré znižujú riziko tvorby komplexného stromu by bolo časovo náročné.

Správnosť algoritmu Algoritmus pre tvorbu rozhodovacieho stromu, ktorý podporuje spojité hodnoty je komplexným algoritmom, ktorý by bolo treba dostatočne otestovať čo by bolo časovo náročné. Algoritmy obsiahnuté vo frameworkoch strojového učenia sú testované celým tímom testerov a teda vzniká menšie riziko chýb ako pri ručne implementovanom algoritme.

Miera abstrakcie Frameworky strojového učenia poskytujú rôzne implementácie algoritmov (ID3, C4.5) pre tvorbu stromov. Implementácia viacerých druhov algoritmov v réžií jednotlivca by zabrala veľké množstvo času.

Za účelom voľby vhodného frameworku pre vytváraný systém detekcie anomálií nasleduje analýza možných frameworkov strojového učenia.

1.7 Frameworky strojového učenia

Vytváraný systém bude implementovaný ako systém pre asistovanú automatickú detekciu anomálií. Po tom, ako užívateľ pomocou webového rozhrania rozdelí

záznamy obsahujúce namerané hodnoty metrík do preddefinovaných koncových tried, musí sa systém toto rozdelenie naučiť aby vedel identifikovať budúce namerané záznamy a klasifikovať ich do koncových tried podľa toho, v ktorej z koncových tried sú čo najviac podobné hodnoty záznamov. K dosiahnutiu tohto naučenia bude využitý niektorý z frameworkov, ktorý tento prístup k vývoju umožňujú.

1.7.1 Weka 3

Weka[26] je kolekciou algoritmov strojového učenia pre dolovanie dát. Vo frameworku sú obsiahnuté nástroje pre vizualizáciu, regresiu, klasifikáciu, preprocesing a clustering. Weka je open source software vydaný pod GNU GPL licenciou[27]. Weka je primárne vytvorená pre programovací jazyk Java, avšak existuje riešenie IKVM.NET[28] , ktoré povolí volať Javovské triedy priamo zo zdrojového kódu v jazyku .NET. IKVM je rovnako vydávaný pod licenciou GNU GPL[27].

Výhody:

- Licencia GNU GPL[27]
- Obsahuje potrebné algoritmy pre klasifikáciu

Nevýhody:

- Určený pre Javu - potreba využitia IKVM.NET, čím vzniká v projekte viac závislostí na iných knižniciach, čo by mohlo spôsobovať problémy pri prípadných budúcich rozšíreniach systému
- Neprehľadná dokumentácia projektu - použitie formátov pdf, chýbajúce prehľadné rozčlenenie do celkov

1.7.2 Accord.NET Framework

Accord.NET je framework využívaný pre vedecké výpočty v jazyku .NET[29]. Je postavený na AForge.NET[2], ktorý je populárnym frameworkom pre spracovanie obrázkov. Accord.NET rozširuje tento framework o nové knižnice, ktoré obsahujú široký záber riešení pre štatistické spracovanie dát, strojové učenie, rozpoznávanie vzorov a mnohé iné. Je vydávaný pod LGPL[30] licenciou, avšak niektoré jeho časti sú vydávané pod GNU GPL[27].

[2]Domovská stránka: `http://www.aforgenet.com/`

Výhody:

- Prehľadná dokumentácia a veľké množstvo príkladov
- Obsahuje požadované algoritmy pre klasifikáciu
- Vytvorený priamo pre využitie v .NET
- Licencia GNU GPL[27]

1.7.3 Encog Machine Learning Framework

Encog[31] je framework strojového učenia, ktorý podporuje množstvo pokročilých algoritmov, rovnako ako obsahuje podporujúce triedy pre normalizáciu a prípravu dát. Obsahuje knižnice pre algoritmy pre neurónové siete, Bayesové siete, Hidden Markov model a genetické algoritmy. Väčšina algoritmov z frameworku beží paralelne a je škálovateľná. Encog poskytuje vlastný workbench pre jednoduchšiu modeláciu a trénovanie algoritmov. Vydávaný pod licenciou Apache 2.0 License[32].

Výhody:

- Vlastný workbench uľahčujúci testovanie aplikácie
- Podporuje jazyky Java a .NET

Nevýhody:

- Neposkytuje algoritmy pre tvorbu rozhodovacích stromov
- Menej prehľadná dokumentácia vo formáte pdf oproti riešeniu Accord

1.8 Požiadavky

Výsledkom analýzy sú funkčné a nefunkčné požiadavky kladené na systém:

1.8.1 Funkčné požiadavky

Prehľadné zobrazenie nameraných hodnôt metrík
Systém musí umožniť v užívateľskom rozhraní užívateľovi zobraziť namerané hodnoty metrík. Metriky budú zobrazované pomocou grafových prvkov a bude možné ich zobraziť v reálnom čase, prípade zobraziť metriky z minulosti zadaním časového intervalu. Hodnoty nameraných metrík budú užívateľovi zobrazené takým spôsobom, aby boli namerané hodnoty jedného časového úseku v rovnakej vertikálnej rovine. Toto umožní užívateľovi subjektívne zhodnotiť stav systému v daný čas.

Efektívne označenie nameraných hodnôt metrík
Keďže sa jedná o systém s asistovanou detekciou anomálií, bude od užívateľa vyžadovaná určitá súčinnosť pri detekcii anomálií a to konkrétne pri vytváraní prvotnej množiny pre tvorbu rozhodovacieho stromu. Užívateľ bude musieť subjektívne rozhodnúť, do akej cieľovej triedy patria hodnoty nameraných metrík v určitom čase. Systém musí byť navrhnutý tak, aby táto súčinnosť bola pre užívateľa čo najjednoduchšia.

Vyhodnocovanie nameraných hodnôt metrík
Systém bude vyhodnocovať namerané hodnoty metrík pomocou vzniknutého rozhodovacieho stromu. Systém bude užívateľa informovať o výslednom stave tohto procesu.

Zobrazenie vzniknutého rozhodovacieho stromu
Systém užívateľovi musí umožniť zobraziť vzniknutý rozhodovací strom. V grafickej reprezentácií stromu musia byť zobrazené medzné hodnoty a matematické logické operátory aby užívateľ v prípade potreby mohol jednoznačne identifikovať cestu k výslednému uzlu stromu, ktorý hodnotí namerané hodnoty metrík.

Tvorba stromu musí byť horizontálne škálovateľná
Systém bude schopný monitorovať viac než jednu službu, kde každá služba môže bežať vo viacerých inštanciách. Pre každú takúto inštanciu bude vytvorený vlastný rozhodovací strom. Keďže tvorba stromu môže byť časovo náročná, bude proces tvorby stromu horizontálne škálovateľný.

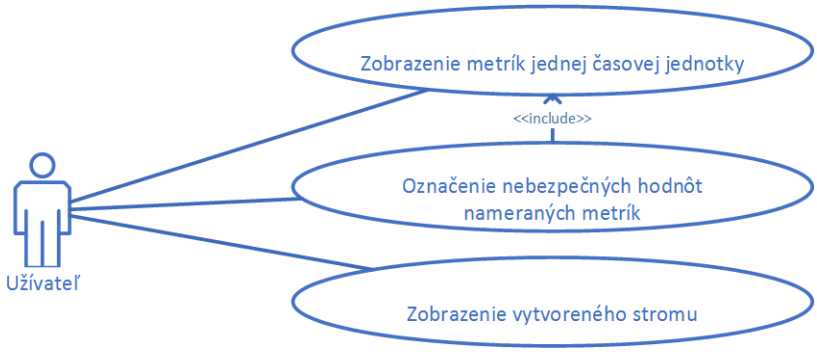
1.8.2 Nefunkčné požiadavky

- Využitie dát z existujúceho systému pre monitoring služieb[1]
- Grafické užívateľské rozhranie implementované v technológií `ASP.NET`
- Využitie techniky rozhodovacích stromov

1.9 Prípady použitia systému

Vytváraný systém je navrhovaný ako asistovano-automatický systém, preto sa možné interakcie užívateľa so systémom týkajú iba súčinnosti užívateľa pri vytváraní tréningovej množiny a zobrazení vytvoreného rozhodovacieho stromu(Obr. 1.5). Užívateľ spustí webového klienta a zobrazí si monitorované služby. Následne z inštancií zvolí tú, pre ktorú chce vytvoriť rozhodovací strom. Užívateľ zvolí záložku pre zobrazenie hodnôt nameraných metrík pre danú inštanciu služby. Od užívateľa bude vyžadovaná súčinnosť pri vytváraní rozhodovacieho stromu. Užívateľ ohodnotí namerané hodnoty metrík do koncových tried. Systém musí užívateľovi umožniť zobraziť namerané hodnoty, označiť

užívateľom zvolenú skupinu týchto hodnôt a priradiť im koncovú klasifikačnú triedu. Systém z označených dát vytvorí rozhodovací strom a začne vyhodnocovať aktuálny stav služby podľa tohto stromu a zobrazovať aktuálny stav na webovom rozhraní. Užívateľ si bude môcť zobraziť vytvorený rozhodovací strom.



Obr. 1.5: Diagram prípadov použitia pre najčastejšie interakcie užívateľa so systémom.

Kapitola 2

Návrh

V kapitole Návrh je popísaná integrácia navrhovanej služby do už existujúceho riešenia pre monitoring škálovateľných služieb. Ďalej je popísaný návrh tvorby tréningovej množiny pre rozhodovací strom, návrh tvorby rozhodovacieho stromu, práca s existujúcim rozhodovacím stromom a klasifikácia nameraných hodnôt po tvorbe rozhodovacieho stromu.

2.1 Tvorba tréningovej množiny

Stávajúci systém okrem numerických metrík odosielaných v periodickom intervale implementuje aj metriky typu Log[1], ktoré reprezentujú logy monitorovanej aplikácie(Kapitola 1.1.2.1). Tento druh metrík nie je odosielaný do Management service periodicky, ale iba pri jeho prípadnom výskyte(Kapitola 1.1.2.1). Metrika typu Log bude preto vo vytváranom systéme spracovávaná osobitne, a to tak, že pri výskyte Log metriky bude užívateľovi odoslaná notifikácia obsahujúca debug level tohto log záznamu a jeho správu. Vytváraný systém teda bude pracovať iba s metrikami, ktoré majú numerickú povahu a sú do systému odosielané periodicky.

Koncové triedy, ktorými je možné klasifikovať namerané hodnoty metrík boli zvolené:

Ok služba sa chová podľa predpokladu užívateľa a teda nie je potrebné upozorniť užívateľa na stav služby.

Warning namerané hodnoty metrík sú vyššie ako pri leveli Ok, nejedná sa však o kritické hodnoty. Je vhodné upozorniť užívateľa na toto chovanie služby.

Error Namerané hodnoty metrík sú vysoké, blížia sa k maximálnym možným hodnotám. Je pravdepodobné že systém nie je v poriadku, je vhodné upozorniť užívateľa na stav služby.

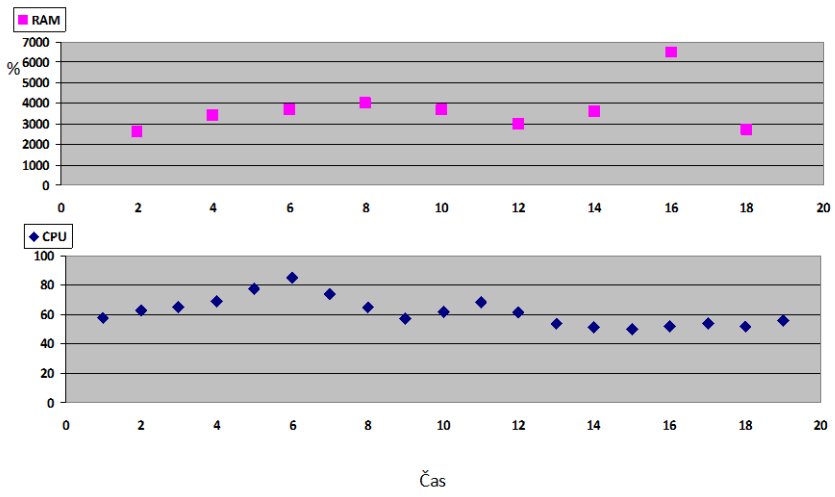
Fatal Jednotlivé merané hodnoty metrík dosahujú maximálne možné hodnoty a je vysoko pravdepodobné že inštancia systému vykazuje anomáliu v správaní. Je nutné upozorniť užívateľa na stav služby.

Pri voľbe koncových klasifikačných tried bolo prihliadané na logovacie levely využívané v logovacích nástrojoch akými sú napríklad log4net, NLog a iné[33]. Jednotlivé zvolené klasifikačné triedy reflektujú závažnosť výšky nameraných hodnôt podobne ako logovacie nástroje reflektujú svojimi levelmi závažnosť chyby v systéme. Koncové klasifikačné triedy boli zvolené tak aby zodpovedali týmto levelom používaným v logovacích nástrojoch.

2.1.1 Webové rozhranie

Na vizualizáciu grafov bude použitá JavaScriptová knižnica AmCharts, ktorá ponúka riešenie pre časovo orientované dáta, umožňuje použiť viac grafov v jednej grafovej komponente[23] a je ponúkaná bezplatne za podmienky, že na grafe bude uvedený odkaz na stránku produktu AmCharts(Kapitola 1.5.2) čo splňuje požiadavky stanovené pre systém.

Na webovom rozhraní budú s využitím knižnice AmCharts implementované grafové komponenty kde jedna grafová komponenta bude zobrazovať hodnoty jednej z nameraných metrík. Tieto komponenty budú zobrazené jedna pod druhou(Obr. 2.1) a nimi zobrazované budú synchronizované podľa času, aby mohol užívateľ jednoducho vytvoriť interval cez všetky zobrazené grafy. Dáta nebudú získavané priamo z Report service, ale z perzistencie.



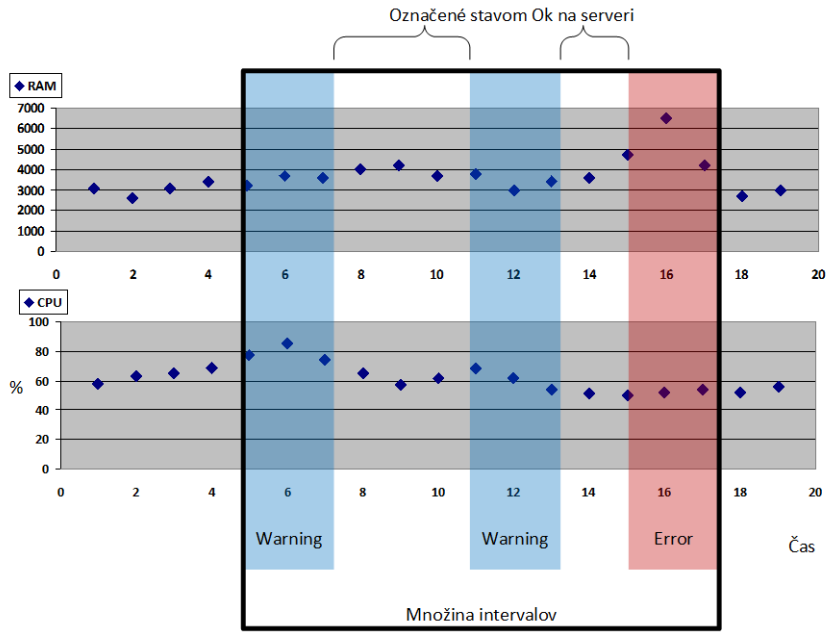
Obr. 2.1: Zobrazenie grafových komponent na webovom rozhraní. Synchronizácia dát podľa času umožní užívateľovi jednoducho zhodnotiť stav služby v čase na základe hodnôt zobrazených metrík

Je to z dôvodu, že na webovom rozhraní budú zobrazované okrem hodnôt nameraných v reálnom čase aj hodnoty z minulosti(Kapitola 1.4.1).

Pri získavaní dát z minulosti bude užívateľ zadávať dátum a čas začiatku a konca intervalu, podľa ktorého mu systém zobrazí namerané hodnoty dát. Maximálny interval, ktorý môže užívateľ zvoliť bude zhora obmedzený na maximum 48 hodín, pretože zvolenie väčšieho intervalu sa pre užívateľa stáva časovo neefektívne z pohľadu označovania hodnôt metrík. Keďže grafová komponenta zobrazujúca namerané hodnoty je veľkostne obmedzená rozlíšením monitoru na ktorom je zobrazovaná, zobrazenie hodnôt za viac ako 48 hodín má vysokú granularitu a tak by sa mohlo stať, že vysoké namerané hodnoty ostanú prehliadnuté. Preto bude na grafovej komponente implementovaná funkcia priblíženia (zoom), avšak ostáva platiť že užívateľovi je časovo efektívnejšie, v prípade že chce zobraziť interval väčší ako 48 hodín, rozdeliť ho na menšie intervaly a označiť ich samostatne.

Ďalším dôvodom pre načítanie dát z perzistencie a nie priamo z Report service je fakt, že dáta z Report service prichádzajú do systému vo forme dávky, ktorá môže obsahovať hodnoty všetkých merateľných metrík za určitý časový interval[1]. Avšak vstupné dáta pre graf musia byť v podobe zoznamu hodnôt metriky, ktorá je na grafe zobrazovaná.

Po označení intervalu užívateľ klasifikuje hodnoty v intervale jednou z preddefinovaných tried. Za účelom zjednodušenia práce užívateľovi bude užívateľovi stačiť, aby na webovom rozhraní označil intervaly triedami rôznymi od triedy Ok. Tieto naoznačované intervaly budú odoslané na server ako množina intervalov(Obr. 2.2). Každý odosielaný interval bude obsahovať dátum začiatku intervalu, dátum konca intervalu, koncovú triedu ktorou bol označený a identifikátor množiny intervalov do ktorej patrí. Odosielaná množina intervalov bude okrem zoznamu intervalov obsahovať dátum vytvorenia množiny a identifikátor inštancie služby ktorej dáta boli označované. Množina intervalov obsahuje aj čas začiatku množiny, ktorý je totožný s časom začiatku prvého intervalu odosielaného v množine a čas konca množiny, ktorý je rovnaký ako čas konca posledného intervalu. Vďaka tomu užívateľ nemusí na grafe označovať hodnoty do koncovej triedy Ok. Táto množina je ďalej spracovaná v serverovej časti systému.



Obr. 2.2: Grafické znázornenie odosielanej množiny intervalov reprezentujúce množinu obsahujúcu troch vytvorených intervalov.

Užívateľ bude mať možnosť zmeniť konfiguráciu vytváraného stromu, teda bude môcť zvoliť merané metriky a dopočítavané metriky, z ktorých chce vytvoriť rozhodovací strom. Ďalej bude môcť rozhodnúť či chce do tvorby stromu zahrnúť aj namerané hodnoty metrík, z ktorých už bol nejaký z minulých rozhodovacích stromov vytvorený alebo chce použiť iba novo označené hodnoty.

Z analýzy vyplýva že pre vytváraný systém je vhodnejšie použiť algoritmus C4.5(Kapitola 1.3.4). Vo vytváranom systéme však bude vytvorená aj možnosť použiť pre tvorbu rozhodovacieho stromu algoritmus ID3. Užívateľ si bude môcť zvoliť algoritmus ktorým bude ďalší rozhodovací strom vytváraný. Pri voľbe algoritmu však musí užívateľ brať do úvahy, že z povahy algoritmu ID3 vyplýva že nie je schopný pracovať s inými hodnotami ako diskrétnymi[17], pričom väčšina hodnôt metrík monitorovaných v Report service je spojitých. Jedinými metrikami, ktoré majú diskrétne hodnoty sú vo vytváranom systéme dopočítavané metriky. Použitie algoritmu ID3 bude vo vytváranom systéme implementované, avšak pôjde skôr o výhľadové využitie algoritmu v budúcnosti pri prípadnom rozšírení systému o nové metriky, ktoré majú diskrétne hodnoty.

2.1.2 Serverová časť

Na serveri, konkrétne v Management service, dôjde k uloženiu prijatých dávok do perzistencie. Proces prijatia intervalov do Management service bude nasledovaný kontrolou na počet užívateľom vytvorených intervalov, ktoré neboli použité pri vytváraní žiadneho z rozhodovacích stromov. V prípade, že tento

počet intervalov dosiahne požadovaný počet intervalov, Management service odošle pokyn na vytvorenie rozhodovacieho stromu do fronty úloh. Číslo označujúce dostatočný počet intervalov bude nastaviteľné v konfiguračnom súbore aplikácie. Pri tvorbe nového rozhodovacieho stromu bude použitá posledná užívateľom zadaná konfigurácia stromu. V prípade, že užívateľ žiadnu stromovú konfiguráciu nevytvorí, bude použitá predvolená konfigurácia, ktorá bude obsahovať všetky merané metriky, všetky dopočítavané metriky a strom bude vytvorený len z nameraných hodnôt, z ktorých ešte nebol vytvorený žiadny rozhodovací strom. V prípade, že bol prijatý určitý počet vytvorených dávok, odošle Management service pokyn na tvorbu rozhodovacieho stromu do fronty.

Správy z tejto fronty sú vyťahované samostatnou DecisionTree service implementovanou ako Windows service, ktorá sa stará o tvorbu tréningovej množiny pre rozhodovací strom a rozhodovacieho stromu. DecisionTree service z perzistencie získa stromovú konfiguráciu pre vytváraný rozhodovací strom aby získala metriky, z ktorých bude vytváraná tréningová množina pre rozhodovací strom. Táto služba spracuje množiny intervalov vytvorené užívateľom tak, že sú vytiahnuté namerané hodnoty metrík uvedených v stromovej konfigurácií v intervale podľa času začiatku a konca množiny a následne sú klasifikované do koncových tried. Klasifikácia prebieha takým spôsobom, že ak bola hodnota nameraná v čase medzi časom začiatku a konca intervalu, je jej priradená rovnaká koncová trieda ako je trieda intervalu. Ak bola hodnota nameraná v čase medzi začiatkom a koncom množiny, avšak nepatrí do žiadneho z intervalov v množine, je jej priradená koncová trieda OK.

Pretože tvorba tréningovej množiny pre rozhodovací strom a tvorba rozhodovacieho stromu sú časovo náročné procesy, ktoré sú naviac vytvárané pre každú inštanciu služby, bude DecisionTree service implementovaná aby bežala paralelne vo viacerých vláknach. Spoločnými kritickými sekciami kde by potencionálne mohli nastať synchronizačné chyby bude fronta úloh, z ktorej si každé vlákno vytiahne správu a databáza, do ktorej vlákna zapisujú vytvorenú tréningovú množinu a vytvorený rozhodovací strom.

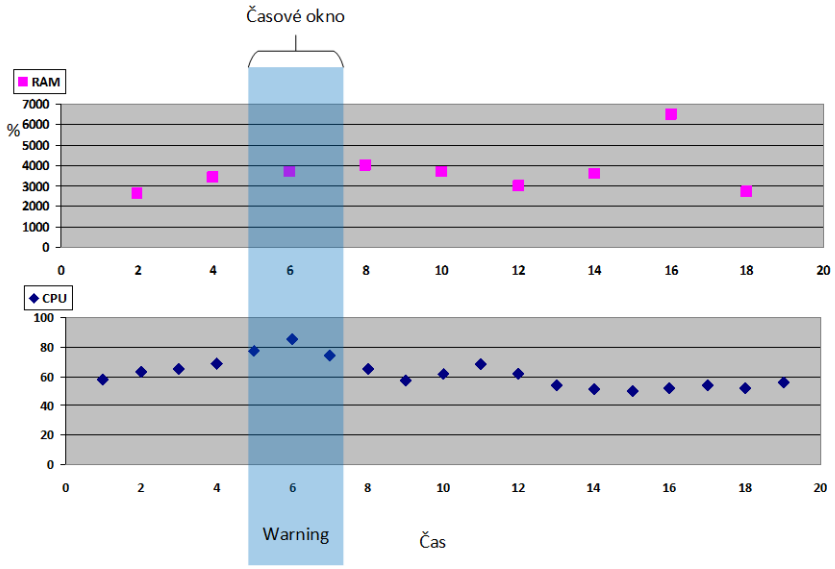
2.1.2.1 Spracovanie dát

Jedným z problémov pri integrácií so stávajúcim systémom pre monitoring služieb je spôsob získavania nameraných hodnôt metrík pre zobrazenie hodnôt na webovom rozhraní a využitie v rozhodovacích stromoch pri tvorbe vstupných dát.

Merané hodnoty v Report service nie sú merané rovnakou periódou[1]. To znamená, že dáta pre hodnoty vyťaženie CPU môžu byť merané napríklad každú sekundu, pričom hodnoty pre využitie operačnej pamäte môžu byť merané v perióde napríklad dvoch sekúnd(Obr. 2.3). Tento problém je viac

problémom pre tvorbu rozhodovacieho stromu ako pre zobrazenie na užívateľskom rozhraní. Zatiaľ čo na webovom rozhraní by v jednom časovom úseku nebolo náročné zobraziť viac nameraných hodnôt pre jeden typ metriky, pri tvorbe rozhodovacieho stromu to už problémom byť môže. Algoritmus musí mať unifikované vstupné dáta, teda nie je možné aby pre jeden záznam určitého časového úseku mal 4 hodnoty pre vyťaženie CPU pričom iný záznam bude mať hodnoty pre vyťaženie CPU napríklad len 2.

Tu sa naskytá niekoľko možností riešenia. Prvou možnosťou je použiť aritmetický priemer nameraných hodnôt jedného typu metriky za daný časový úsek. Tým pre jeden časový úsek vzniká jedna hodnota pre každý typ monitorovanej metriky. Inou možnosťou je použiť medián nameraných hodnôt. Obe techniky sa na prvý pohľad nemusia javiť ako úplne vhodné, pretože je zrejmé že pri nich dôjde k určitej strate presnosti. Je však potreba si uvedomiť, že za daný časový úsek, ktorý spravidla býva o dĺžke maximálne jednotiek sekúnd, je vysoko nepravdepodobné aby obor meraných hodnôt dosahoval veľký rozptyl.



Obr. 2.3: Počet nameraných hodnôt pre časový úsek sa pre jednotlivé metriky môže odlišovať. Pri vytváraní stromu je však potrebné mať pre jeden časový úsek len jednu hodnotu pre každú metriku.

2.2 Tvorba stromu

Na tvorbu rozhodovacieho stromu bolo zvolené riešenie Accord.NET Framework (Kapitola 1.7.2) a to z dôvodu, že ponúka riešenie implementované priamo pre platformu .NET a nie je potreba využívať žiadne ďalšie knižnice

pre jeho fungovanie[29]. Accord v rámci svojich klasifikačných algoritmov obsahuje priamo implementáciu rozhodovacích stromov[34]. Framework Accord je poskytovaný pod LGPL[30] licenciou.

Vstupom algoritmu pre tvorbu rozhodovacích stromov sú dáta vytvorené v tréningovej množine. Tieto dáta sú vo forme zoznamu, kde každý prvok tohto zoznamu je tvorený polom hodnôt všetkých metrík, z ktorých je vytváraný rozhodovací strom, a koncovou triedou, ktorá bola konkrétnemu záznamu pridelená užívateľom pomocou webového rozhrania. Každá z týchto metrík môže reprezentovať jeden stromový atribút, podľa ktorých bude algoritmus tvoriť rozhodovací strom. Algoritmus nemusí nutne vytvoriť stromový atribút pre každú z poskytnutých hodnôt, pretože môže existovať prípad, kedy namerané hodnoty metriky nemajú na klasifikáciu žiadny vplyv. Objekty Accord frameworku sú serializovateľné[29] a preto bude vytvorený strom zaserializovaný a uložený do perzistencie.

Meranými numerickými metrikami v Report service , a teda aj možnými stromovými atribútmi, sú:

Vyťaženie CPU Vyťaženie CPU prostredia, na ktorom beží monitorovaná služba

Využitie pamäti Využitie fyzickej/virtuálnej pamäti prostredia, na ktorom beží monitorovaná služba

Počet bežiacich vlákien počet bežiacich vlákien systému, na ktorom beží monitorovaná služba

Počet bežiacich procesov počet bežiacich procesov systému, na ktorom beží monitorovaná služba

Doba odpovedi časový interval od odoslania požiadavku na službu až po prijatie odpovedi

Disk I/O využitie pevných diskov v systéme

Dopočítavanými metrikami, ktoré môžu tvoriť stromové atribúty, sú:

Deň v týždni Poradové číslo dňa v týždni, kde začiatkom týždňa je nedeľa

Deň v mesiaci Poradové číslo dňa v mesiaci

2.3 Použitie rozhodovacieho stromu

Po tom ako bol pre inštanciu vytvorený prvý rozhodovací strom, začne systém monitorovať danú inštanciu podľa tohto stromu. Vstupnými dátami pre klasifikáciu stavu služby podľa vytvoreného stromu sú namerané hodnoty tých

metrík, ktoré boli použité pri vytváraní stromu, teda tie isté, ktoré užívateľ zvolil v konfigurácií stromu. Tieto hodnoty sú zoradené do pola hodnôt, kde každá hodnota reprezentuje jednu z metrík. Na základe tohto pola rozhodovací strom určí koncovú triedu pre daný vstup.

Keďže Report service posiela balíky dát do Management service periodicky je možné tieto dáta využiť na zistenie aktuálneho stavu inštancie ešte pred tým ako budú uložené do perzistencie. Prijatý balík dát obsahuje jednu alebo viac hodnôt pre každú meranú metriku[1]. Z tohto balíku dát budú vytiahnuté metriky, ktoré užívateľ zvolil v stromovej konfigurácií a pre každú z týchto metrík bude vypočítaný aritmetický priemer z nameraných hodnôt tejto metriky. Tak bude pri každom prijatí dát od Report service vytvorený vstup, ktorý bude následne rozhodovacím stromom vyhodnotený, čoho výsledkom bude jedna z koncových tried reprezentujúca stav inštancie služby. Každá z inštancií bude mať iniciálny stav Ok, a pri každej zmene tohto stavu bude na webovom rozhraní v reálnom čase zobrazená nová hodnota stavu pomocou indikátoru stavu služby. Ďalej bude táto zmena uložená do perzistencie a to z dôvodu, aby mohol byť užívateľ po príchode na webové rozhranie systému upozornený na neštandardné správanie inštancie za čas, kedy webové rozhranie spustené nemal.

2.3.1 Zobrazenie rozhodovacieho stromu

Jednou z požiadaviek je grafické zobrazenie vytvoreného rozhodovacieho stromu. V tomto rozhodovacom strome musia byť zobrazené základné informácie o každom stromovom uzle, aby užívateľ mohol jednoducho určiť hraničné hodnoty kedy sa monitorovaná inštancia dostáva do iného stavu. Pred tým, ako bude rozhodovací strom zobrazený na webovom rozhraní, je potrebné ho získať z perzistencie, deserializovať a premapovať jednotlivé uzly stromu tvorené triedami z frameworku Accord na lokálne triedy systému, ktoré budú zaslané webovému klientovi.

Na grafické zobrazenie vytvoreného stromu bude použitá JavaScriptová open source knižnica Treant JS vydaná pod licenciou MIT[35]. Knižnica slúži na vizualizáciu stromových grafov a preto je ideálnym riešením pre vizualizáciu vytvoreného rozhodovacieho stromu[36].

Vo vytvorenom strome budú graficky odlíšené uzly priraďujúce koncovú klasifikačnú triedu - stromové listy. Obrazovka so stromom musí byť skrolovateľná, aby v prípade, že vytvorený rozhodovací strom bude príliš veľký bolo možné ho celý zobraziť. Pre pridanie prehľadnosti pre užívateľa bude v strome implementovaná funkcionalita schovávania a rozbaľovania stromových vetví (Na obrázku Obr. 3.1 znázornené žltou farbou).

2.3.2 Indikátor stavu služby

Management service prijíma správy s nameranými hodnotami metrík od Report service v intervaloch dlhých prevažne pár sekúnd. Po tom, ako bol vytvorený rozhodovací strom, Management služba každú prijatú správu vyhodnotí pomocou stromu patriacemu inštancií služby, pre krotú boli prijaté dáta namerané. Výsledkom hodnotenia je aktuálny stav služby, ktorý je, okrem iného, potrebné zobraziť na webovom rozhraní v reálnom čase.

K implementácií tejto funkcionality bude použitá knižnica ASP.NET SignalR. ASP.NET SignalR je knižnica, ktorá uľahčuje implementáciu real time komunikácie. Umožňuje obojsmernú komunikáciu medzi serverom a klientom. Zmeny odoslané serverom na klienta sú na klientovi viditeľné ihneď. SignalR podporuje použitie WebSocketov a je kompatibilný aj so staršími webovými prehliadačmi. Zároveň obsahuje API pre správu pripojení, hromadné pripojenia a autorizáciu[37].

Keďže rozhodovací strom je vytváraný pre každú inštanciu monitorovanej služby (Kapitola 1.2.2), na každej obrazovke na webovom rozhraní týkajúcej sa inštancie služby bude zobrazený indikátor stavu služby(Obr. C.4). Vo webovom klientovi bude implementované rozhranie pomocou SignalR, vystavujúc dve metódy.

Metóda pre zmenu indikátoru
: Metóda starajúca sa o zmenu indikátoru o stave služby. Management service po vyhodnotení prijatej správy s nameranými hodnotami metrík skontroluje, či nastala zmena stavu služby proti poslednej vyhodnotenej správe. Ak zmena nastala, Management service odošle signál na rozhranie webového klienta, ktorý sa postará o zmenu indikátoru podľa príslušného výsledného stavu na aktuálne otvorenej webovej stránke.

Metóda pre notifikácie
: Metóda na odosielanie notifikácií pre užívateľa. Management service pomocou tejto metódy informuje užívateľa o prípadných validáciách alebo o výsledkoch užívateľských požiadaviek. V prípade, že užívateľ nemal spusteného webového klienta, ale systém pre detekciu anomálií spustený bol a niektorá z inštancií služieb sa dostala do stavu rôzneho od Ok, slúži táto metóda na informovanie užívateľa o týchto zmenách stavu.

2.4 Architektúra služby

V tejto časti bude popísaná navrhovaná architektúra vytváraného systému. Systém bude rozdelený do niekoľko častí. Pri implementácií budú rozšírené už

existujúce časti stávajúceho systému a vznikne nová časť starajúce sa o tvorbu rozhodovacieho stromu(Obr. 2.6). Týmito časťami sú:

Report service - nie je potrebné robiť ďalšie zmeny

Klientsky framework - nie je potrebné robiť ďalšie zmeny

Mobilní klienti - nie je potrebné robiť ďalšie zmeny

Management service(serverová časť) - Management service bude rozšírená o biznis logiku aplikácie. Bude sa starať o prijatie rozdelených dát od užívateľa prichádzajúcich z webového klienta. Ďalej bude služba implementovať logiku pre odosielanie pokynov pre tvorbu rozhodovacích stromov do fronty úloh. Služba sa bude starať o premapovanie rozhodovacieho stromu do tried implementovaných vo vytváranej aplikácií za účelom zobrazenia stromu na webovom rozhraní. Služba bude implementovať funkcionalitu pre odosielanie pokynov na zmenu stavu indikátoru služby v reálnom čase.

Webový klient - Webový klient bude rozšírený o nové grafické rozhranie, ktoré bude umožňovať užívateľovi rozšírenú prácu s nameranými hodnotami metrík. Ďalej bude klient rozšírený o obrazovky slúžiace na zobrazenie rozhodovacieho stromu, indikátor stavu služby a históriu stavov služby. Na klientovi bude implementovaná služba komunikujúca s Management service, ktorá sa bude starať o zmenu indikátoru stavu služby v reálnom čase.

Databáza - Databáza bude rozšírená aby bola schopná uložiť dáta potrebné k vytváraniu tréningovej množiny a dáta vznikajúce po vytvorení množiny. Ďalej musí byť databáza schopná uložiť vytvorený rozhodovací strom aj s všetkými dátami spojenými s týmto stromom. Databáza bude ukladať informácie o stave systému, ktoré budú zobrazované užívateľovi.

Služba pre tvorbu rozhodovacieho stromu - Z dôvodu časovej náročnosti vytvárania tréningovej množiny(Kapitola 1.4) pre rozhodovací strom a vytvárania samotného rozhodovacieho stromu bude vo vytváranom systéme implementovaná služba, ktorá bude mať na starosti tieto procesy(Kapitola 1.6). Služba pobeží vo viacerých vláknach a bude implementovaná tak, aby bola jednoducho horizontálne škálovateľná. Vo vytváranom systéme bude implementovaná nová Windows service s názvom DecisionTree service, ktorá bude mať na starosti tvorbu tréningovej množiny pre rozhodovací strom a tvorbu samotného rozhodovacieho stromu. Služba bude mať prístup k databáze, z ktorej bude využívať dáta. Služba bude komunikovať s Management service cez frontu úloh, z ktorej bude DecisionTree service vyťahovať správy s informáciami pre tvorbu stromu.

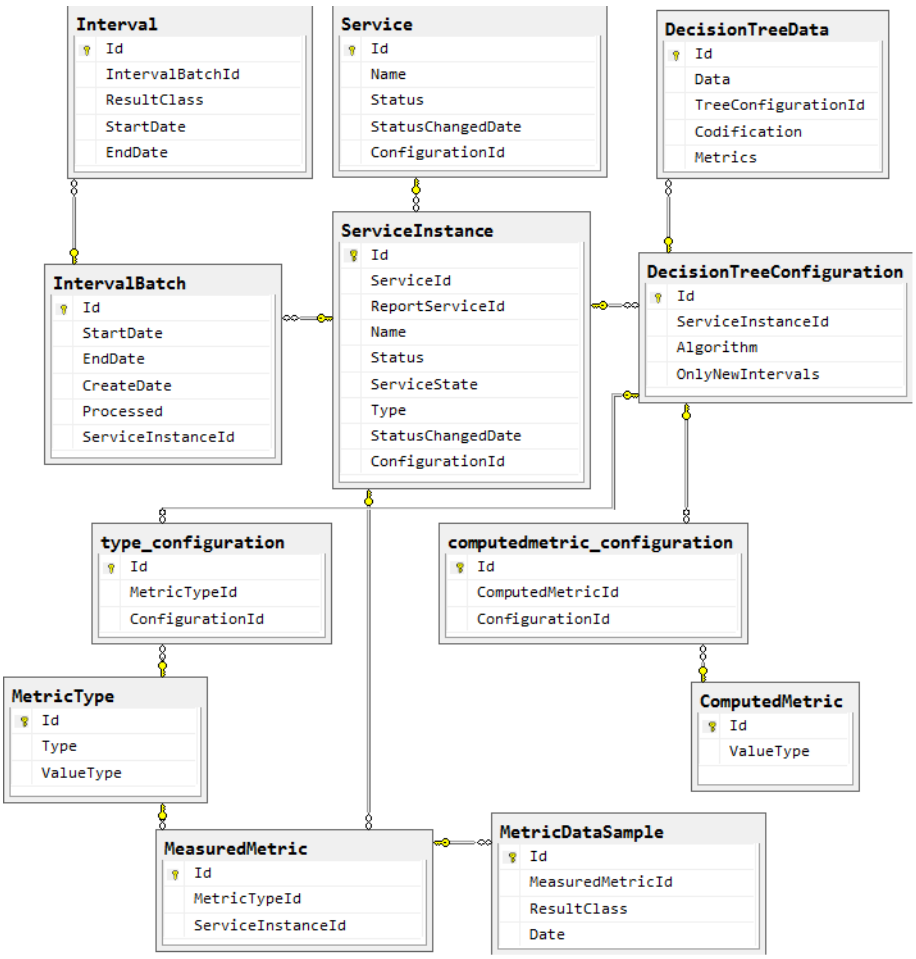
2.4.1 Webový klient

Webový klient je momentálne prispôsobený na zobrazenie nameraných dát pre jednotlivé služby a aj pre ich inštancie. Ďalej je možné tieto služby a inštancie rôzne konfigurovať. Každá z týchto služieb a rovnako aj každá z inštancií služieb je zobrazovaná na vlastnej webovej stránke. Tvorba rozhodovacích stromov bude prebiehať pre každú z inštancií monitorovaných služieb zvlášť (Kapitola1.2.2)a pre reflektovanie týchto požiadaviek bude musieť byť táto stránka rozšírená o novú funkcionalitu. Z dôvodu prehľadnosti je nutné stránku rozdeliť na záložky a každá z týchto záložiek bude implementovať jednu z funkcionalít(Obr. C.5). Na každej zo záložiek bude vytvorený indikátor stavu služby zobrazujúci aktuálny stav inštancie podľa výsledku z rozhodovacieho stromu. Domovskou záložkou na stránke bude stávajúca stránka inštancie implementovaná v existujúcom systéme zobrazujúca základné dáta o inštancií. Ďalej pribudne záložka, na ktorej budú vytvorené grafické komponenty pre zobrazenie nameraných hodnôt a označovanie týchto hodnôt ťahom po grafe. Táto záložka bude slúžiť na vytváranie tréningovej množiny užívateľom. Ďalšou bude záložka, ktorá bude zobrazovať rozhodovací strom vytvorený pre konkrétnu inštanciu služby. Poslednou bude záložka, na ktorej budú zobrazované zmeny stavu inštancie služby podľa výsledkov z rozhodovacieho stromu.

Pri implementácií webového klienta bude využívaný front-endový framework Bootstrap[38]. Bootstrap zjednodušuje vývoj webového grafického rozhrania a je poskytovaný pod licenciou MIT[35].

2.4.2 Databáza

Relačná databáza momentálne slúži na ukladanie informácií o meraných službách a ich inštanciách, ukladanie meraných typov metrík a ich hodnôt a informácie o inštanciách Report service(Kapitola 1.1.4). Databáza musí byť rozšírená tak, aby umožňovala uložiť dáta slúžiace na tvorbu tréningovej množiny, teda je potrebné vytvoriť nové tabuľky pre intervaly a množinu intervalov, rozšíriť tabuľky pre namerané hodnoty metrík aby im bolo možné priradiť koncovú triedu a vytvoriť tabuľku pre rozhodovací strom a jeho konfiguráciu(Obr. C.8).



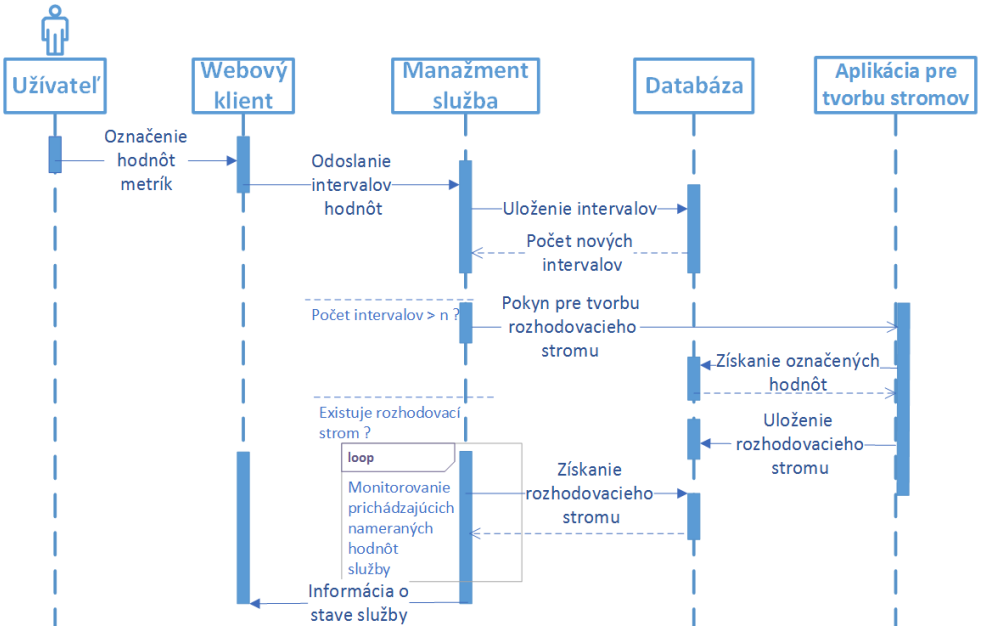
Obr. 2.4: Časť databázového modelu zobrazujúca entity používané vytváraným systémom

2.4.3 Management service

Management service je hlavnou serverovou časťou systému(Kapitola 1.1.2.2). Jej úlohou je obsluha pripojenia všetkých ostatných častí systému. Služba vystavuje webovú službu poskytujúcu potrebné metódy nutné pre funkčnosť Report service, od ktorej prijíma namerané dáta. Ďalej Management service uchováva všetky namerané hodnoty metrík vo svojej lokálnej databáze. Táto služba bude ďalej rozšírená, aby umožňovala:

- spracovať dáta potrebné pre tvorbu tréningovej množiny
- odoslať požiadavku na fronty úloh pre Windows service, ktorá bude vytvárať rozhodovacie stromy

- deserializovať a premapovať vytvorený rozhodovací strom do vlastných objektov a tak umožnila zobrazenie stromu na webovom rozhraní
- spracovať a pomocou stromu klasifikovať novo prijaté dáta od Report service
- zmeniť indikátor stavu služby podľa výsledkov z rozhodovacieho stromu v reálnom čase



Obr. 2.5: Sekvenčný diagram pre tvorbu nového rozhodovacieho stromu.

2.4.4 Služba pre tvorbu rozhodovacieho stromu

Vytvorená bude nová Windows service - DecisionTree service, ktorá pobeží na rovnakom prostredí ako implementovaný systém pre detekciu anomálií. DecisionTree service bude mať za úlohu tvorbu tréningovej množiny pre rozhodovací strom a tvorbu samotného rozhodovacieho stromu. DecisionTree service bude prijímať správy z fronty úloh, ktoré sú do fronty posielané Management service. Jednotlivé správy budú obsahovať potrebné informácie o inštancií, pre ktorú bude mať byť vytvorený rozhodovací strom. DecisionTree service bude mať prístup do databázy, z ktorej bude využívať dáta pre tvorbu tréningovej množiny a tvorbu rozhodovacieho stromu. Keďže oba tieto procesy sú časovo náročné, DecisionTree service pobeží paralelne vo viacerých vláknach, pričom počet vlákien bude nastaviteľný v konfiguračnom súbore služby.

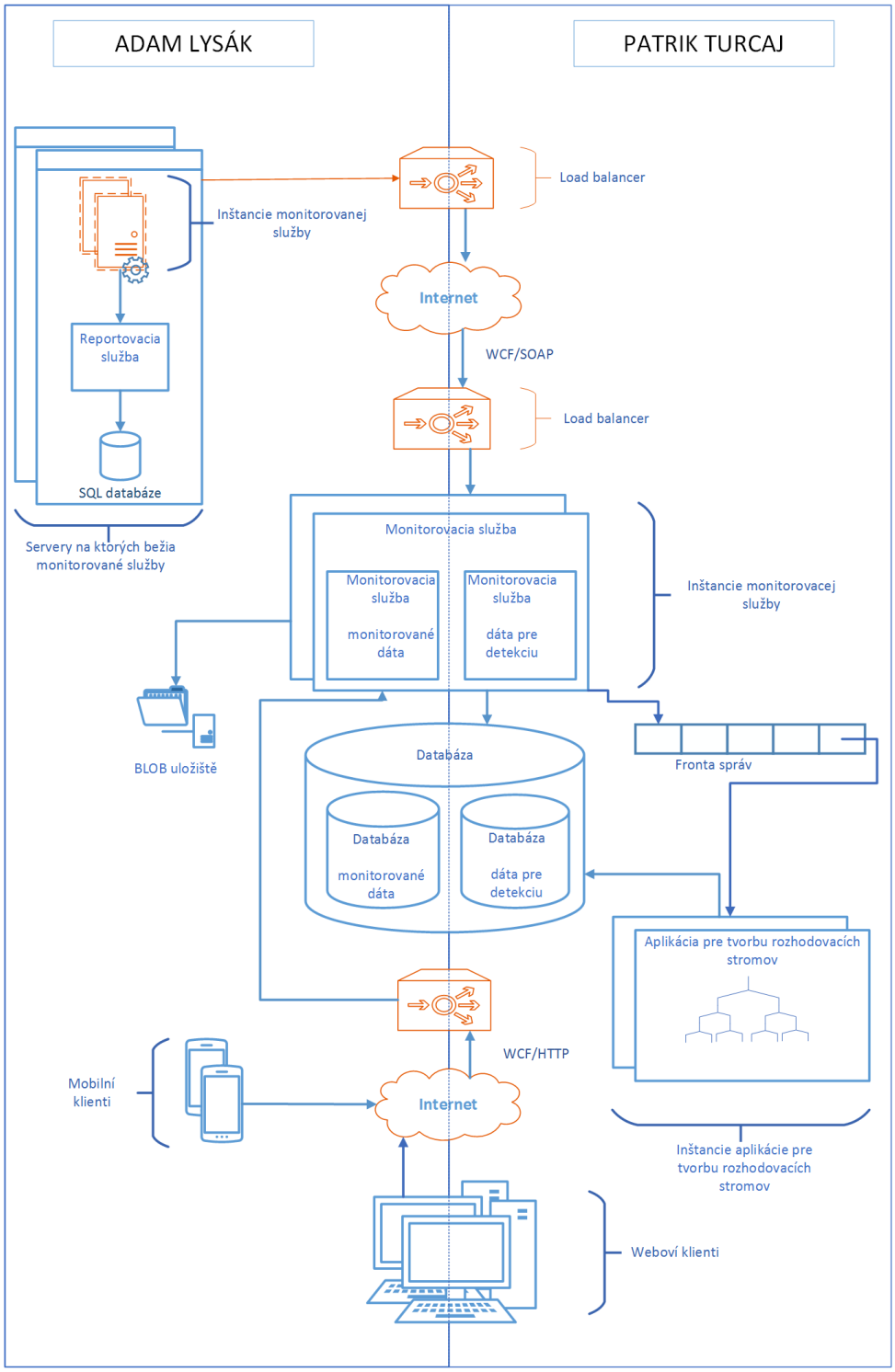
Ako implementácia fronty úloh bude zvolená ActiveMQ[39]. ActiveMQ je open source middleware orientovaný na správy od Apache Software Foundation [40]. ActiveMQ je vydávaný pod Apache 2.0 License[32]. Pre prijímanie správ z fronty pre službu bežiacu vo viacerých paralelných vláknach je možné zvoliť dva spôsoby implementácie:

Prijímanie po 1 správe

Prvou možnosťou bolo implementovať triedu, ktorá spustí zadaný počet vlákien. Každé z týchto vlákien si vytiahne správu z fronty, spracuje ju a ďalej vytvára tréningovú množinu a rozhodovací strom. Východzie nastavenie prefetch politiky fronty realizovanej ActiveMQ je na hodnote 1000. To znamená, že každý konzument fronty prijme 1000 správ z fronty naraz. Pri tomto riešení je teda potreba nastaviť prefetch politiku fronty na hodnotu 1. To zaručí, že každé vlákno, ktoré je konzumentom fronty, dostane presne 1 správu. Ak je fronta prázdna, vlákna pomocou metódy busy waiting čakajú na príchod novej správy do fronty.

Prijímanie po viac ako 1 správe

Druhou možnosťou bolo implementovať triedu, ktorej hlavné vlákno je konzument fronty. Toto vlákno by prijalo od fronty nie 1, ale vyšší zvolený počet správ a tieto správy by uložilo do concurrent safe kolekcie. Následne by bol vytvorený zvolený počet vlákien. Každé z týchto vytvorených vlákien by si vytiahlo správu z kolekcie, spracovalo ju a ďalej vytváralo tréningovú množinu a rozhodovací strom. Pri tejto možnosti by bolo možné nastaviť prefetch politiku na väčšie číslo, čo by zvýšilo časovú efektivitu pretože sa obmedzí dotazovanie do fronty.



Obr. 2.6: Navrhnutá architektúra vytváraného systému. Na obrázku je schematicky znázornené rozdelenie komponent práce podľa autorov, ktorí stoja za ich tvorbou. Oranžoví časti nie sú implementované.

Kapitola 3

Implementácia

Táto kapitola popisuje implementáciu vytváraného systému. V prvej časti je popísaný software pomocou ktorého bol vytvárný systém implementovaný. Nasleduje časť v ktorej sú popísané vybrané časti implementácie.

3.1 Vývojový software

Keďže vytváraný systém priamo naväzuje na stávajúci systém pre monitoring služieb, ktorý bol implementovaný na platforme .NET Framework, bola jednou z požiadaviek implementácia na platforme .NET Framework. Vývoj prebiehal na jedinom stroji s operačným systémom Windows. Pre vývoj systému boli použité nástroje:

Microsoft Windows 7 Professional x64
: Vývoj prebiehal na operačnom systéme Windows 7 Professional, ktorý je potrebný k behu vývojového prostredia Visual Studio 2015 Enterprise IDE.

Microsoft Visual Studio Enterprise 2015
: Integrované vývojové prostredie od spoločnosti Microsoft, podporujúce vývoj na platforme .NET Framework.

Microsoft SQL Server 2016 Developer Edition
: Relačný databázový systém spoločnosti Microsoft.

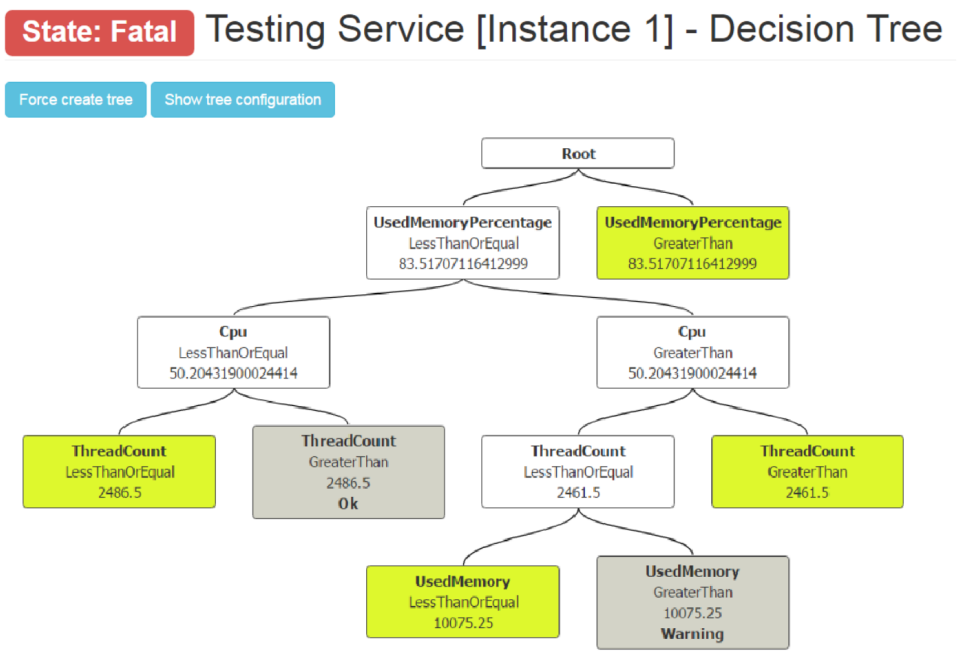
3.2 Vybrané časti implementácie

3.2.1 Služba pre tvorbu rozhodovacieho stromu

Pre tvorbu DecisionTree service bol využitý framework Topshelf, ktorý poskytuje knižnice podporujúce tvorbu Windows service za použitia platformy

.NET[41]. Za účelom jednoduchšieho vývoja a testovania bola vytvorená konzolová aplikácia, ktorá bola po dokončení implementácie transformovaná na Windows service za použitia Topshelf. Framework Topshelf je vydávaný pod licenciou Apache 2.0 License[32].

Keďže tvorba vstupnej tréningovej množiny pre rozhodovací strom a tvorba samotného stromu sú časovo náročné procesy, bola vytvorená DecisionTree service, ktorá sa stará práve o tieto procesy. Pre zvýšenie efektivity služby bola implementovaná tak aby bola jednoducho horizontálne škálovateľná a naviac je DecisionTree service implementovaná aby bežala vo viacerých vláknach. Služba pracuje tak, že získava správy z fronty úloh. Každá táto správa obsahuje informáciu o inštancií služby, pre ktorú má DecisionTree service vytvoriť rozhodovací strom(Obr. 3.1).



Obr. 3.1: Screenshot vytvoreného rozhodovacieho stromu monitorovaných metrík pre jednu z inštancií služby.

V návrhu boli zvažované dva spôsoby prístupu k vyťahovaniu požiadaviek z fronty. (Kapitola 2.4.4) Aj keď je druhá zvažovaná možnosť časovo efektívnejším riešením, bolo pre implementáciu zvolené prvé riešenie a to z viacerých dôvodov. Prvým dôvodom, prečo nebola zvolená druhá možnosť je fakt, že pri druhej možnosti je potreba počítať s výrazne náročnejšou implementáciou plánovania vlákien. Ďalším z dôvodov je skutočnosť, že povaha služby nepredpokladá

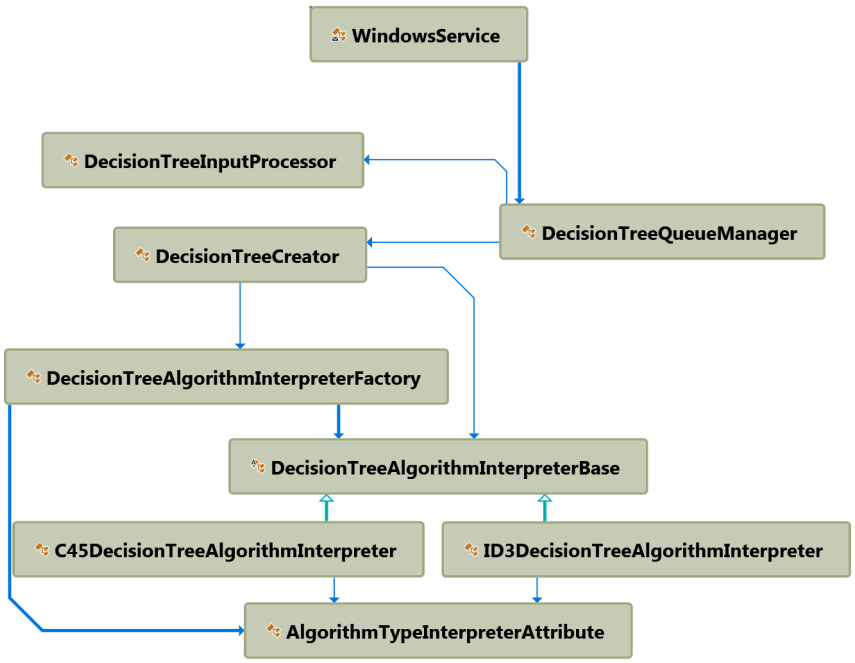
také extrémne množstvá prijatých správ do fronty, pri ktorých by bola časová náročnosť dotazovania sa do fronty každým spusteným vláknom postrehnuteľná.

DecisionTree service je rozdelená do troch hlavných častí:

DecisionTreeQueueManager
: Časť DecisionTree service starajúca sa o obsluhu fronty správ.

DecisionTreeInputProcessor
: Časť DecisionTree service zabezpečujúca tvorbu tréningovej množiny pre rozhodovací strom.

DecisionTreeCreator
: Časť DecisionTree service v ktorej je implementovaná tvorba rozhodovacieho stromu a jeho ukladanie do databázy.



Obr. 3.2: Diagram hierarchie tried aplikácie pre tvorbu rozhodovacích stromov. Šípky znázorňujú využívanie triedy.

3.2.2 Implementácia metrík

V existujúcom systéme pre monitoring služieb bolo postačujúce, aby zoznam monitorovaných metrík bol reprezentovaný ako výčet typov meraných metrík.

Vo vytváranom systéme bolo nutné transformovať reprezentáciu jednotlivých meraných metrík z dôvodu že metriky boli v existujúcom systéme využívané len na definíciu meraných metrík. Každá uložená hodnota nameranej

metriky obsahovala identifikátor metriky, aby bolo možné rozpoznať ktorej metrike nameraná hodnota patrí. Keďže vo vytváranom systéme sa počíta s rozšírením funkcionality pre tieto metriky, bolo nutné transformovať stávajúce riešenie tak, aby umožňovalo rozšírenú prácu s metrikami a zároveň aby bola zachovaná jednoduchá rozšíriteľnosť o nové metriky pri prípadných budúcich rozšíreniach systému. To bolo dosiahnuté zavedením nových levelov abstrakcie do systému a pre každý typ metriky je vo vytvorenom systéme implementovaná nová trieda.

Pre rešpektovanie prehľadnosti a rozšíriteľnosti systému bolo vo vytvorenom systéme implementované rozhranie, ktoré za použitia návrhového vzoru abstract factory a vlastných atribútov realizuje prácu s meranými metrikami. Rozhranie implementuje abstraktné triedy pre numerické a dopočítavané metriky. Každá trieda rozširujúca jednu z abstraktných tried je označená vlastným atribútom reprezentujúcim typ meranej metriky a poskytuje biznis logiku pre spracovávanie nameraných dát konkrétnej metriky. Systém pri prvom spustení prehľadá linkované knižnice a na základe vytvorených atribútov nájde typy tried, ktoré implementujú jednotlivé metriky a uloží ich do zoznamu. V prípade použitia danej metriky je podľa tohto typu vytvorená inštancia triedy, ktorá reprezentuje typ konkrétnej metriky a ďalej spracováva namerané hodnoty tejto metriky. Vďaka tomuto riešeniu je vytvorený systém jednoducho rozšíriteľný o nové typy metrík.

Pri tvorbe rozhodovacieho stromu je potrebné pre každý možný typ stromového uzlu vytvoriť možný stromový atribút. Stromový atribút je vytváraný dvoma spôsobmi v závislosti na tom, či sa jedná o diskrétnu alebo spojitú veličinu. V prípade, že je veličina diskrétna, je potrebné poznať počet možných výstupných hodnôt (napríklad 7 pre dopočítavanú metriku "Deň v týždni").

Ďalej pre každý užívateľom zvolený typ metriky musí byť možné pripraviť vstupné dáta z nameraných hodnôt pre rozhodovací strom. Vstupné dáta sú pripravované rozdielne podľa toho, či sa jedná o tvorbu stromu alebo o klasifikáciu hodnôt.

Príprava dát pre tvorbu stromu

Z relačnej databázy sú vytiahnuté dáta jedného časového intervalu pre všetky užívateľom zvolené metriky. Pre každú metriku je vytiahnuté pole nameraných hodnôt a koncová klasifikačná trieda priradená užívateľom. Za účelom minimalizovania počtu dotazov do databázy bola pri dotazovaní využívaná vnútorná optimalizácia databázy, ktorá vyťahovaní dáta ihneď spracovávala tak, aby výstup zodpovedal požadovanému formátu dát kde musí byť jedna hodnota pre každú metriku za jeden časový úsek.

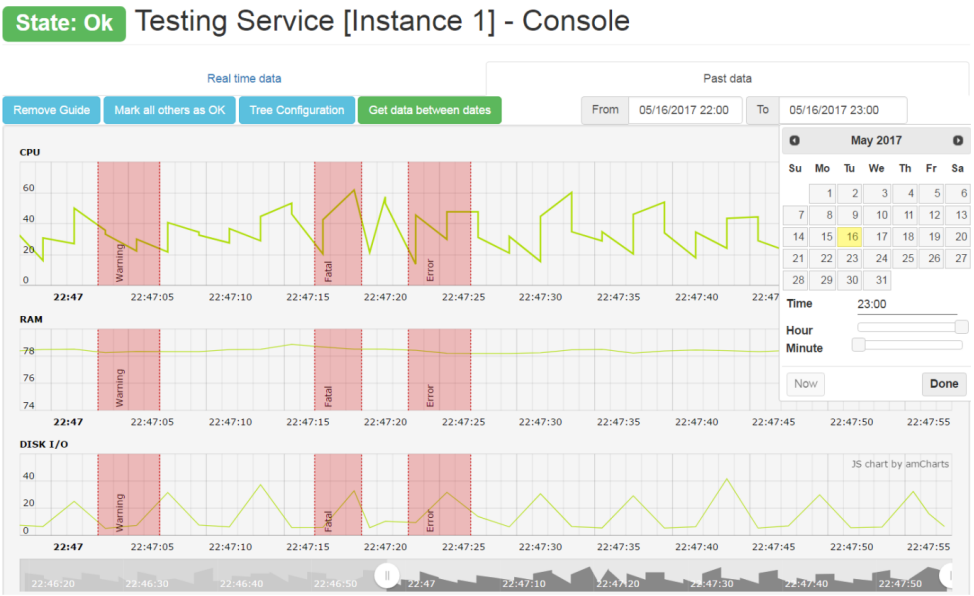
Príprava dát pre klasifikáciu hodnôt

Tento typ prípravy dát prebieha za účelom vyhodnotenia hodnôt pomocou stromu. Hodnoty nie sú získavané z databázy ale dáta sú vyhodnotené ihneď po ich prijatí z Report service za účelom vyhodnotenia aktuálneho

stavu systému. Dáta z Report service prichádzajú v nepravidelných intervaloch vo forme balíku, ktorý obsahuje jednu alebo až niekoľko hodnôt pre každú monitorovanú metriku. Tento balík je potrebné spracovať na jedno pole hodnôt, kde každá hodnota patrí jednej metrike. Spracovanie hodnôt je riešené pomocou aritmetického priemeru nameraných hodnôt pre každú z metrík.

3.2.3 Implementácia intervalov

Podľa návrhu boli na webovom rozhraní implementované grafické komponenty umožňujúce užívateľovi rozdeliť namerané hodnoty dát do preddefinovaných koncových tried pomocou ťahu kurzora po jednom z grafov(Obrázok 3.3).



Obr. 3.3: Screenshot webového rozhrania pre označovanie nameraných hodnôt pre konkrétnu inštanciu služby do klasifikačných tried.

Takto vytvorené dáta sú odoslané na server do Management service kde sú následne spracované a uložené do perzistencie. Ukážka spracovaného formátu množiny intervalov je zobrazená v tabuľke 3.1.

Interval Batch					
Id	Start Date	End Date	Create Date	Processed	Instance Id
1	22:47:03 16.05.2017	22:47:27 16.05.2017	22:49:36 16.05.2017	false	1
2	22:49:17 16.05.2017	22:49:42 16.05.2017	22:51:07 16.05.2017	false	1

Tabuľka 3.1: Zobrazenie formátu spracovaných dát prijatej množiny intervalov.

Príslušné dáta sú následne spracované procesom popísaným v kapitole 2.1.2. Na základe týchto dát je v službe pre tvorbu rozhodovacích stromov vytváraná tréningová množina a to tak, že pre jednotlivú dávku intervalov sú získané dáta intervalov na základe ktorých je priradená koncová trieda nameraným hodnotám metrík. Dáta intervalov pre množinu intervalov v tabuľke 3.1 sú znázornené v tabuľke 3.2.

Interval				
Id	Interval Batch Id	Result Class	Start Date	End Date
1	1	1	22:47:03 16.05.2017	22:49:07 16.05.2017
2	1	3	22:47:17 16.05.2017	22:49:20 16.05.2017
3	1	2	22:47:23 16.05.2017	22:49:27 16.05.2017

Tabuľka 3.2: Zobrazenie formátu spracovaných dát prijatých intervalov pre množinu intervalov zobrazenú v tabuľke 3.1

Kapitola 4

Testovanie a výsledky

Táto kapitola sa zaoberá testovaním implementovaného systému pre detekciu anomálií. Vzhľadom k tomu že systém je pri reálnom použití distribuovaný na rôzne výpočtové nody bolo testovanie zamerané predovšetkým na funkčnosť a spoľahlivosť celého systému. Výsledkom testovania je konštatovanie, či je technika rozhodovacích stromov vhodná pre tento typ problému.

4.1 Testovanie v priebehu vývoja

Testovanie v priebehu vývoja prebiehalo za pomoci troch testovacích aplikácií vytvorených v stávajúcom systéme pre monitoring služieb. Aplikácie boli vytvorené tak, aby čo najviac simulovali typy služieb, ktoré budú monitorované v produkčnom prostredí. Týmito aplikáciami sú:

- Vzorová testovacia aplikácia bežiaca na platforme ASP.NET
- Testovacia konzolová aplikácia bežiaca na platforme Windows
- Vzorová služba bežiaca na platforme Windows (Windows service)

V priebehu celého vývoja bol systém testovaný proti týmto aplikáciám simulujúcim cieľové platformy, vďaka čomu dochádzalo k odhaľovaniu množstva chýb systému už pri samotnom vývoji. V rámci testovania bola vizuálne kontrolovaná správnosť vytvoreného rozhodovacieho stromu pomocou webového rozhrania. Vzhľadom k tomu že systém umožňuje zobraziť užívateľovi rozhodovací strom za účelom zobrazenia vzťahov medzi metrikami pri detekcií anomálneho chovania služby, bolo možné jednoducho kontrolovať správnosť vytvárania stromov pri postupnom označovaní nameraných hodnôt. Týmto spôsobom bolo kontrolované fungovanie systému popri vývoji až po finálne nasadenie systému.

4.1.1 Testovanie vyhodnocovania stromu

Vzhľadom k tomu že rozhodovací strom bol tvorený externou knižnicou bolo nutné za účelom zobrazenia na webovom rozhraní premapovať vytvorený strom do tried vytvoreného systému(Kapitola 1.7.2). Pri užívateľskej požiadavke na zobrazenie stromu na webovom rozhraní je rozhodovací strom vytiahnutý z databázy a premapovaný z objektov patriacich frameworku Accord.NET do objektov vytváraného systému. Účelom testu bolo overiť správnosť zobrazeného stromu na webovom rozhraní, pretože podľa zobrazeného stromu môže užívateľ posúdiť správanie inštancie webovej služby. Podľa zobrazeného stromu boli vytvorené cesty od koreňového uzlu až po listové uzly a následne boli vytvorenému stromu upravené vstupné dáta podľa týchto ciest, pričom koncové klasifikačné triedy sa museli zhodovať.

4.1.2 Kontrola paralelného behu

V priebehu celého vývoja aplikácie bola kontrolovaná DecisionTree service pre tvorbu rozhodovacích stromov, ktorá beží simultánne vo viacerých vláknach súčasne či nedôjde k synchronizačným chybám medzi vláknami. DecisionTree service bola nakonfigurovaná aby bežala v rôznych počtoch paralelne bežiacich vlákien, pričom bolo spustených jedna až desať inštancií tejto služby. Vlákna pracovali nad rovnakou inštanciou, aj nad rôznymi inštanciami monitorovanej služby.

4.1.3 Testovanie grafického rozhrania

Systém rieši problematiku označenia neoznačených vzoriek pomocou rozšírenia webového rozhrania o grafové komponenty s funkciou označovania pomocou ťahu kurzoru po grafe. Bola testovaná správnosť vytvárania množiny intervalov a jednotlivých intervalov, pričom testované boli intervaly s veľkým rozostupom medzi intervalmi, odosielanie intervalov v rôznych počtoch, intervaly zoradené podľa času, intervaly zoradené proti času, náhodne zoradené intervaly, množina intervalov kde medzi intervalmi chýbali namerané dáta a iné. Zámerom bolo odhaliť chyby vzniknuté zadávaním intervalov, či už pri ukladaní intervalov alebo pri tvorbe tréningovej množiny pre rozhodovací strom.

4.1.4 Testovanie dopočítavaných metrík

Vo vytvorenom systéme vznikli aj dopočítavané metriky Deň v týždni a Deň v mesiaci. Testovanie týchto metrík je časovo podmienené, pretože aby boli v rozhodovacom strome tieto metriky použité ako atribúty stromu, musela by služba bežať minimálne týždeň vkuse, aby tieto metriky mali na rozhodnutie o stave služby nejaký vplyv. Toto správanie bolo simulované zmenou systémového času na testovanom prostredí. Preto bol vytvorený strom, ktorý tieto metriky

obsahoval. Následne boli do stromu vkladané testovacie dáta upravené aby prechádzali týmito stromovými atribútmi.

4.1.5 Simulácia útoku na službu

Systém bol testovaný na úspešnosť detekcie útoku na monitorovanú službu. V testovacom prostredí bola nasadená služba, ktorá nevytvárala vysokú záťaž na systém a u ktorej sa predpokladali nízke hodnoty meraných metrík. Útok bol simulovaný spustením benchmarku na testovacom prostredí, čo malo za následok vyťaženie systému z čoho plynú vysoké namerané hodnoty metrík. Tento test prebiehal tak, že po spustení systému bol na testovacom prostredí spustený benchmark a za pomoci webového rozhrania systému boli v čase behu benchmarku vytvorené intervaly priraďujúce namerané hodnoty do klasifikačných tried. Na základe týchto intervalov systém vytvoril rozhodovací strom. Očakávaným výsledkom bolo že po ďalšom spustení benchmarku bude systém hlásiť anomálie v chovaní služby.

Pri tomto teste bola zistená silná závislosť vytvoreného stromu na testovacej množine. Čím precíznejšie užívateľ označí namerané hodnoty dát, tým podáva strom presnejšie výsledky. Jednou z nevýhod systému je nezávislosť vyhodnotených dát na predchádzajúcich výsledkoch vyhodnotení. Pri teste sa stávalo, že po tom ako bol vytvorený strom, podával systém falošné hlásenia o útoku. Tieto hlásanie podával v prípade, keď bol krátkodobo vyťažený systém napríklad spustením nejakej z ďalších aplikácií v testovacom prostredí.

4.2 Výsledky testovania

Technika rozhodovacích stromov prináša mnoho výhod a jedným z najväčších prínosov je generovanie ľudsky čitateľného výsledku. Vytvorený strom je možné vizualizovať, čo užívateľovi pomáha k pochopeniu behu a jednotlivých súvislostí medzi meranými metrikami služby a užívateľ môže jednoznačne určiť, za akých podmienok sa inštancia služby dostáva do konkrétneho stavu.

Nevýhodou je absencia robustnosti rozhodovacích stromov pre jednotlivé inštancie. Aj veľmi malá zmena v konfigurácií stromu, prípadne v meraných dátach znamená kompletné vytvorenie nového stromu. Pre systémy bežiace v stovkách monitorovaných inštancií môže vytváranie stromu pre každú z inštancií znamenať značné vyťaženie systému. Riešenie je teda lepšie aplikovať na službu bežiacu v menej inštanciách alebo v prípade služby bežiacej vo veľa inštanciách nakonfigurovať vytváranie stromu na menej časté intervaly.

Z dôvodu, že aj kvôli malej zmene v dátach je nutné vytvoriť nový rozhodovací strom, sa stáva použitie rozhodovacích stromov silne závislé na vstupnej tréningovej množine vytváranej užívateľom.

Ďalšou z nevýhod zistených v konečných fázach testovania, ktoré z analýzy a návrhu nevyplývali môže byť fakt, že vyhodnocovanie vstupného záznamu pomocou rozhodovacieho stromu nie je závislé na predchádzajúcich výsledkoch vyhodnotených záznamov. To znamená, že vytvorený systém upozorňuje užívateľa na anomálne chovanie služby už pri prvom výskyte vysokých nameraných hodnôt. Vytvorený systém vo väčšine prípadov správne odhalil simulovaný útok na službu, avšak v systéme sa vyskytla rada falošných hlásení práve z dôvodu nezávislosti na predchádzajúcich záznamoch. Falošné hlásenie nastávalo vtedy, keď systém z určitého dôvodu (spustenie nejakého programu) vykazoval vysoké hodnoty, aj keď len na krátky čas. Namerané dáta za tento krátky čas boli stromom vyhodnotené ako nebezpečné a systém ihneď upozornil užívateľa, aj keď sa jednalo iba o krátke vyťaženie z dôvodu spustenia nejakého programu.

Je však pravdepodobné, že závislosť na predchádzajúcich výsledkoch testovaných záznamov nemusí byť v takto škálovateľnom systéme dostačujúca a jednoducho implementovateľná a bolo by potrebné podľa konkrétnych prípadov použitia doplniť do systému nové metriky, ktoré by kombinovali dáta viacerých metrík, akými by mohli byť napríklad počet dotazov na službu verzus vyťaženie CPU.

Technika rozhodovacích stromov je vhodná na použitie monitorovania anomálneho chovania distribuovaných služieb, pretože dokázala odhaliť väčšinu simulovaných útokov na monitorovanú službu v testovacom prostredí. Vďaka vizualizácií stromu pomáha užívateľovi pochopiť vzťahy medzi jednotlivými monitorovanými metrikami za účelom detekovania stavu služby. Pri jej použití je však nutné počítať vyťažením použitého hardwaru a prispôsobiť tomu nasadenie služby na vytváranie rozhodovacích stromov. Rovnako je nutné počítať s citlivosťou spôsobenou nezávislosťou vyhodnocovaných dát na predchádzajúcich výsledkoch, vďaka čomu sa pri krátkodobom vyťažení systému môžu vyskytnúť falošné hlásenia. Nakoniec musí užívateľ počítať so závislosťou vytvoreného stromu na vytvorenej tréningovej množine a vytvárať množinu s čo najväčšou precíznosťou. Technika rozhodovacích stromov môže byť použitá na detekciu anomálneho chovania v distribuovaných službách a s veľkou pravdepodobnosťou odhalí prebiehajúci útok na službu, avšak z dôvodu falošných hlásení nie je dostatočne spoľahlivá na to aby za jej rozhodnutiach bola implementovaná ďalšia biznis logika a má skôr informatívny charakter.

Záver

Cieľom práce bolo analyzovať projekt pre monitoring škálovateľných služieb vytvorený Adamom Lysákom a s využitím dát z tohto systému navrhnúť a implementovať systém, ktorý má za úlohu detekovať anomálne chovanie distribuovaných služieb. Na základe toho boli stanovené funkčné a nefunkčné požiadavky, ktoré výsledný systém všetky splnil.

Pomocou webového užívateľského rozhrania systém umožňuje užívateľovi zobraziť namerané hodnoty metrík služieb a na základe užívateľovho úsudku tieto hodnoty rozdeliť do klasifikačných tried jednoduchým ťahom kurzoru po grafe. Výsledný systém tieto hodnoty spracováva a za použitia techniky strojového učenia z nich vytvára rozhodovací strom, na základe ktorého ďalej vyhodnocuje správanie služby. Systém užívateľovi umožňuje vytváraný strom rôzne konfigurovať a po jeho vytvorení ho zobraziť na webovom rozhraní za účelom zhodnotenia kritických hodnôt pri detekcií anomálneho chovania. Výsledky klasifikácie z rozhodovacieho stromu sú ukladané a v prípade nepredpokladaného správania služby je užívateľ na toto správanie upozornený. Tvorba rozhodovacích stromov bola implementovaná aby bola jednoducho horizontálne škálovateľná a výsledný systém bol navrhnutý tak, aby v prípade ďalšieho rozšírenia systému bolo pridanie nových monitorovaných metrík čo najjednoduchšie.

Technika rozhodovacích stromov sa na použitie pri detekcií anomálneho chovania v distribuovaných službách ukázala ako použiteľná a s vysokou pravdepodobnosťou odhalenia útoku, avšak z dôvodu falošných hlásení nie je dostatočne spoľahlivá na to, aby na jej rozhodnutiach bola postavená ďalšia biznis logika aplikácie.

Možné rozšírenia systému

Vytvorený systém sa zaoberá veľmi zaujímavou problematikou a aj keď boli splnené všetky požiadavky kladené na systém, stále existuje rada možných rozšírení tohto systému týkajúcich sa tejto problematiky. V priebehu vývoja

a následného testovania vyplynulo niekoľko tém ktoré by stáli za budúce rozšírenie vytvoreného systému:

Formy strojového učenia
Vo vytvorenom systéme bolo implementované strojové učenie pomocou techniky rozhodovacích stromov. Zaujímavým rozšírením by bolo implementovať do systému ďalšie služby starajúce sa o strojové učenie, ktoré by však pre toto učenie používali iné techniky (napr. Bayesove siete) a navzájom tieto techniky porovnať.

Time-based databáza
Zaujímavým rozšírením by mohlo byť pridanie Time-based databázy do systému čím by sa umožnilo vytvorenie nových monitorovaných a dopočítavaných metrík, ktoré sa v systéme nevyskytujú periodicky.

Špecializované metriky
Za účelom obmedzenia falošných hlásení systému by bolo vhodné implementovať do systému špeciálne metriky, ktoré by kombinovali viac meraných metrík, napríklad počet dotazov na webovú službu verzus vyťaženie CPU, prípadne by monitorovali ďalšie metriky ako teplota CPU a iné.

Literatúra

[1] LYSÁK, A.: Systém pro vzdálený monitoring a konfiguraci škálovatelných služeb. [cit. 2017-01-19].

[2] SQLite team: SQLite is the most used database engine in the world. [cit. 2017-01-19]. Dostupné z: `https://www.sqlite.org/`

[3] Wikimedia Foundation, Inc.: Object-relational mapping. 2016. [cit. 2017-01-19]. Dostupné z: `https://en.wikipedia.org/wiki/Objectrelational_mapping`

[4] NHibernate Community: NHibernate Relational Persistence for Idiomatic .NET. [cit. 2017-01-19]. Dostupné z: `http://nhibernate.info/doc/nh/en/index.html`

[5] Prashanth; Begum, S.: Review of Load Balancing in Cloud Computing [online]. [cit. 2017-01-19]. Dostupné z: `https://pdfs.semanticscholar.org/067e/71b38170ea384f38e29d7dccdcfa31f6500b.pdf`

[6] MacDonald, M.; Freeman, A.: *Pro ASP.NET 4 in C# 2010, Fourth Edition*. Berkely, CA, USA: Apress, čtvrté vydání, 2010, ISBN 1430225297, 9781430225294.

[7] Robinson, W. N.: Monitoring Web Service Requirements. In *RE*, IEEE Computer Society, 2003, ISBN 0-7695-1980-6, s. 65–74.

[8] Flanders, J.: *RESTful .NET: Build and Consume RESTful Web Services with .NET 3.5*. O'Reilly Media, Inc., 2008, ISBN 0596519206, 9780596519209.

[9] Mitchell: Decision tree learning [online]. [cit. 2017-01-19]. Dostupné z: `http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf`

[10] Matuška, M.: Rozhodovacie stromy. [cit. 2017-01-19]. Dostupné z: http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0405/matuska/zs.htm

[11] Quinlan, J. R.: Induction of decision trees. *Machine Learning*, ročník 1, č. 1, 1986: s. 81–106, ISSN 1573-0565, doi:10.1007/BF00116251.

[12] Das, K.; Schneider, J.: Detecting Anomalous Records in Categorical Datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, New York, NY, USA: ACM, 2007, ISBN 978-1-59593-609-7, s. 220–229, doi:10.1145/1281192.1281219.

[13] Murthy, S. K.: Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, ročník 2, 1997: s. 345–389.

[14] Wikimedia Foundation, I.: Decision tree learning. 2017 [online]. [cit. 2017-01-19]. Dostupné z: https://en.wikipedia.org/wiki/Decision_tree_learning

[15] Baah, G. K.; Gray, A.; Harrold, M. J.: On-line Anomaly Detection of Deployed Software: A Statistical Machine Learning Approach. In *Proceedings of the 3rd International Workshop on Software Quality Assurance*, SOQUA '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-584-3, s. 70–77, doi:10.1145/1188895.1188911.

[16] HSSINA, B.; MERBOUHA, A.; EZZIKOURI, H.; aj.: A comparative study of decision tree ID3 and C4.5 [online]. [cit. 2017-01-19]. Dostupné z: http://saiconference.com/Downloads/SpecialIssueNo10/Paper_3-A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf

[17] Bahety, A.: Extension and Evaluation of ID3 – Decision Tree Algorithm. [cit. 2017-01-19]. Dostupné z: https://pdfs.semanticscholar.org/bfae/5daa9ca4a6429b1ab837429f562d0a78df7d.pdf

[18] SEBBAN, M.; NOCK, R.; CHAUCHAT, J.; aj.: Impact of learning set quality and size on decision tree performances [online]. [cit. 2017-01-19]. Dostupné z: http://perso.univ-st-etienne.fr/sebbanma/ps/sncr_ilsqsdtp00.pdf

[19] THRUN ET, A.: The monk‘s problem: a performance comparison of different learning algorithms. [cit. 2017-01-19]. Dostupné z: http://www.mli.gmu.edu/papers/91-95/91-28.pdf

[20] Pošík, P.: Decision Trees. University Lecture, Czech Technical University in Prague, 2013. Dostupné z: https://cw.fel.cvut.cz/wiki/_media/courses/a3m33ui/prednasky/dectrees-slides.pdf

[21] Wu, X.; Kumar, V.; Quinlan, J. R.; aj.: Top 10 algorithms in data mining, Published online: 4 December 2007 [online]. [cit. 2017-01-19]. Dostupné z: http://www.cs.uvm.edu/~icdm/algorithms/10Algorithms-08.pdf

[22] AnyChart: AnyStock and Financial Charts is a JavaScript based financial web charting solution. [cit. 2017-01-19]. Dostupné z: http://www.anychart.com/

[23] amCharts: Programming libraries and tools for all your data visualization needs. [cit. 2017-01-19]. Dostupné z: https://www.amcharts.com/stock-chart/

[24] Mike Bostock: Data-Driven Documents. [cit. 2017-01-19]. Dostupné z: https://d3js.org/

[25] Regents of the University of California: The 3-Clause BSD License. Dostupné z: https://opensource.org/licenses/BSD-3-Clause

[26] Machine Learning Group at the University of Waikato: Data Mining Software in Java. [cit. 2017-01-19]. Dostupné z: http://www.cs.waikato.ac.nz/ml/weka/

[27] Free Software Foundation: GNU General Public License. Dostupné z: http://www.gnu.org/licenses/gpl.txt

[28] Jeroen Frijters: IKVM.NET. [cit. 2017-01-19]. Dostupné z: https://www.ikvm.net/

[29] Accord.NET Framework Team: Machine learning, computer vision, statistics and general scientific computing for .NET. [cit. 2017-01-19]. Dostupné z: http://accord-framework.net/index.html

[30] Free Software Foundation: GNU Lesser General Public License. Dostupné z: http://www.gnu.org/licenses/lgpl.txt

[31] Heaton Research: Encog Machine Learning Framework. [cit. 2017-01-19]. Dostupné z: http://www.heatonresearch.com/encog/

[32] Apache Software Foundation: Apache License, Version 2.0. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0

[33] Apache Software Foundation: The Apache log4net library. [cit. 2017-01-19]. Dostupné z: https://logging.apache.org/log4net/

[34] Accord.NET Framework Team: Standard classification problems. [cit. 2017-01-19]. Dostupné z: https://github.com/accord-net/framework/wiki/Classification

[35] Massachusetts Institute of Technology: The MIT License. Dostupné z: https://opensource.org/licenses/MIT

[36] Peručić, F.: Treant JS, JavaScipt library for visualization of tree diagrams. [cit. 2017-01-19]. Dostupné z: http://fperucic.github.io/treant-js/

[37] Corporation, M.: Learn About ASP.NET SignalR. [cit. 2017-01-19]. Dostupné z: https://www.asp.net/signalr/overview/getting-started

[38] Bootstrap Core Team: Bootstrap The world's most popular mobile-first and responsive front-end framework. [cit. 2017-01-19]. Dostupné z: http://getbootstrap.com

[39] Apache Software Foundation: Apache ActiveMQ Features. 2016. [cit. 2017-01-19]. Dostupné z: http://activemq.apache.org/features.html

[40] Snyder, B.; Bosanac, D.; Davies, R.: *ActiveMQ in Action*. Greenwich, CT, USA: Manning Publications Co., 2011, ISBN 1933988940, 9781933988948.

[41] Travis Smith, Chris Patterson Dru Sellers, Henrik Feldt et al.: Topshelf, Put Your Apps on the Topshelf. [cit. 2017-01-19]. Dostupné z: https://topshelf.readthedocs.io/en/latest/

Dodatok A

Slovník

API Application Programming Interface.

BLOB Binary Large Object.

CPU Central processing unit.

CSV Comma-separated values.

HTTP Hypertext Transfer Protocol.

IDE Integrated development environment.

IIS Internet Information Services.

MVC Model View Controller.

RAM Random Access Memory.

REST Representational state transfer.

SOAP Simple Object Access Protocol.

URL Uniform Resource Locator.

WCF Windows Communication Foundation.

XML Extensible markup language.

Dodatok B

Návod k nasadeniu systému

Tento návod popisuje nasadenie jednotlivých častí systému. Návod je čiastočne prevzatý z diplomovej práce Systém pro vzdálený monitoring a konfiguraci škálovatelných služeb[1] od Adama Lysáka a je doplnený o nové informácie pre časti, ktoré boli vo vytvorenom systéme vytvorené alebo rozšírené.

B.1 Serverová časť

Minimálne prerekvizity:

- Windows Server 2012
- Microsoft IIS 8.0
- Microsoft SQL Server 2012
- .NET Framework 4.5 s povoleným WCF HTTP Activation
- Apache ActiveMQ 5.14.4

Vytvorenie databáze:

1. Vytvorte databázu s názvom RSM
2. Spusťte databázový skript s názvom CreateScript.sql zo zložky bin\Server\Database

B.1.1 Monitorovacia služba

1. Vytvorte v IIS webovú službu pod názvom RSM.ManagementService bežiacu v .NET Frameworku 4.5
2. Nakopírujte do umiestnenia vytvorenej služby súbory zo zložky bin Server ManagementService
3. V súbore RSM.ManagementService.Container.config upravte:

- Connection string pre prístup k databáze
- V prípade potreby zmeňte umiestnenie lokálneho BLOB úložiska

4. V súbore Web.config upravte:
   - URL Adresu ActiveMQ serveru
   - Adresu webového klienta (Hub service - notifikačná služba)
5. Spusťte webovú službu v IIS

B.1.2 Služba pre tvorbu rozhodovacích stromov

Postup inštalácie a spustenie služby DecisionTree service:

1. Spustite program `cmd.exe` ako administrátor
2. Navigujte konzolu do umiestnenia služby
3. V zložke bin\Debug zadajte príkaz pomocou ktorého službu nainštalujete: `./RSM.MachineLearning.Core.exe install`
4. Službu spustíte príkazom `./RSM.MachineLearning.Core.exe start`

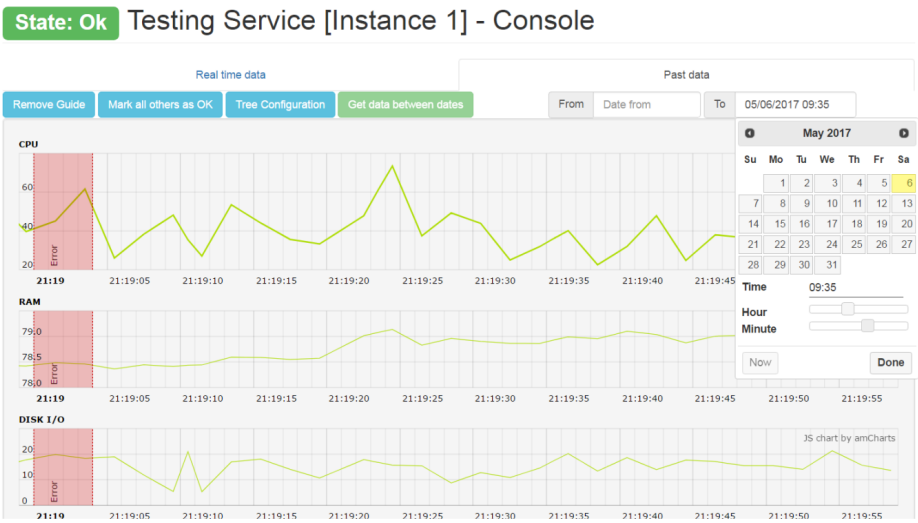
B.2 Webový klient

Postup nasadenia webového klienta:

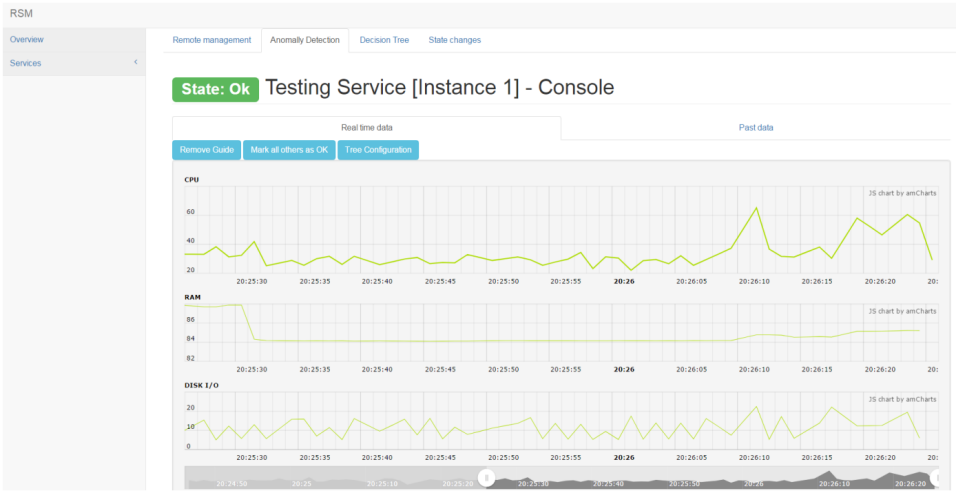
1. Vytvorte v IIS webovú aplikáciu pod názvom RSM.WebClient bežiacu na platforme .NET Framework 4.5
2. Nakopírujte do umiestnenia vytvorenej webovej aplikácie súbory zo zložky bin\Client\WebClient
3. V súbore Web.config v sekcií system.serviceModel/client upravte adresu Management service.
4. Spusťte webovú aplikáciu v IIS.

Dodatok C

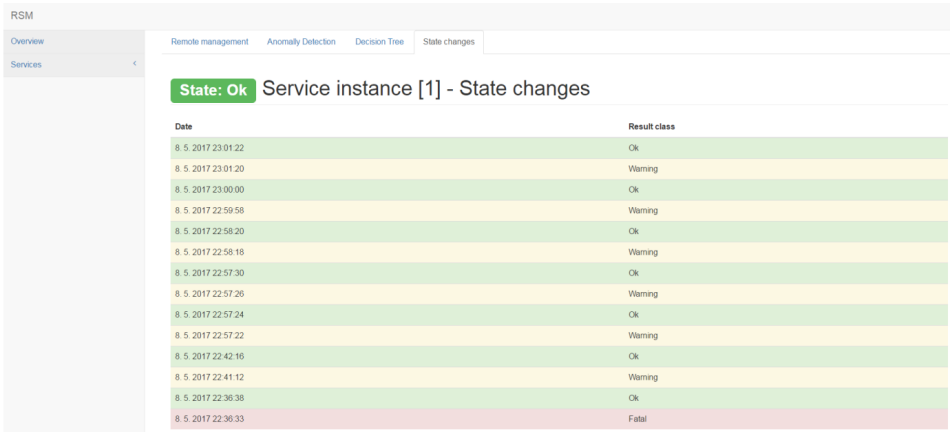
Snímky obrazovky webového rozhrania



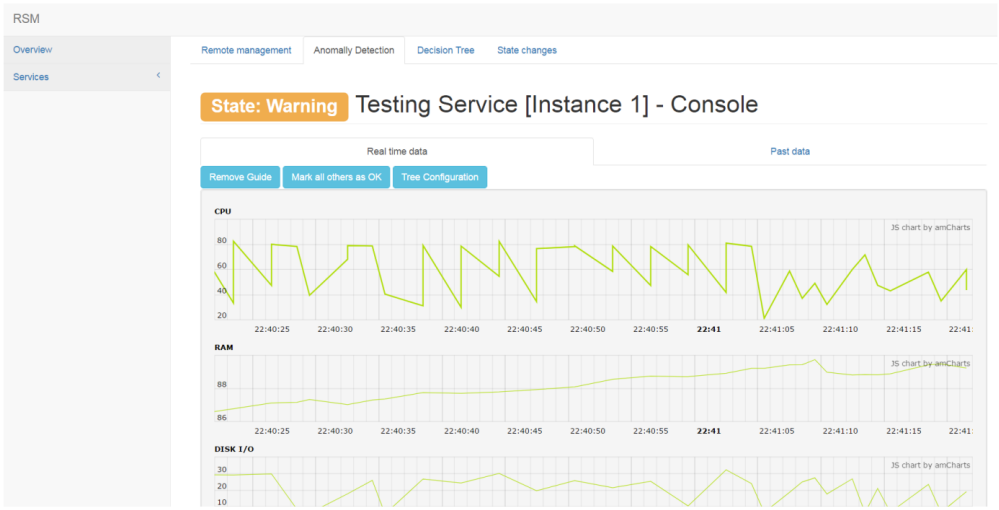
Obr. C.1: Screenshot zachytávajúci zobrazenie nameraných dát z minulosti na grafovej komponente umožňujúcej vytváranie intervalov.



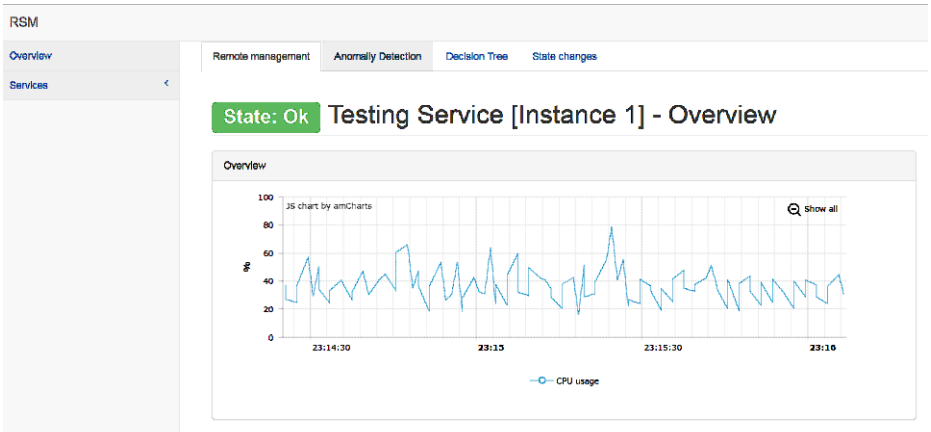
Obr. C.2: Screenshot zachytávajúci zobrazenie nameraných dát v reálnom čase na grafovej komponente umožňujúcej vytváranie intervalov.



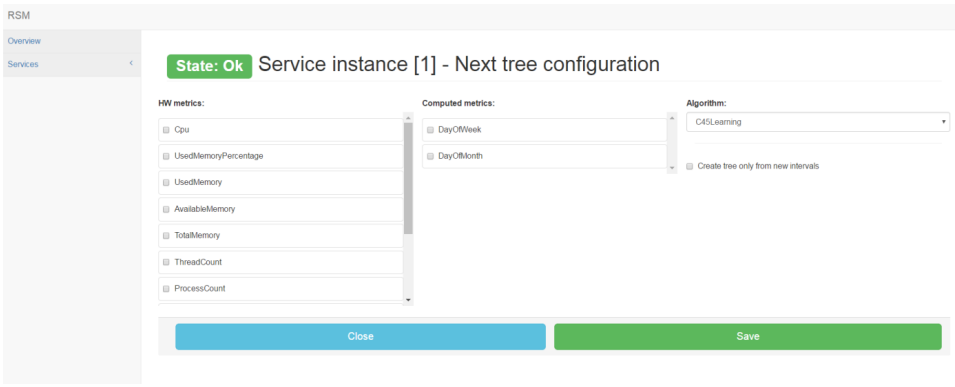
Obr. C.3: Screenshot zachytávajúci históriu stavov inštancie služby vyhodnotených podľa vytvoreného rozhodovacieho stromu.



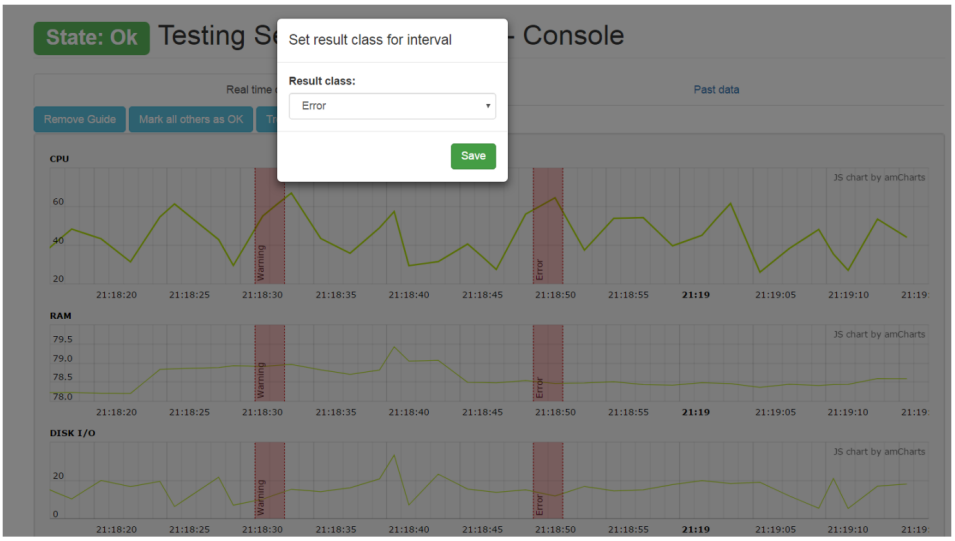
Obr. C.4: Screenshot indikátoru stavu služby upozorňujúceho na službu v stave Warning z dôvodu vysokých nameraných hodnôt RAM.



Obr. C.5: Screenshot záložiek na stránke inštancie

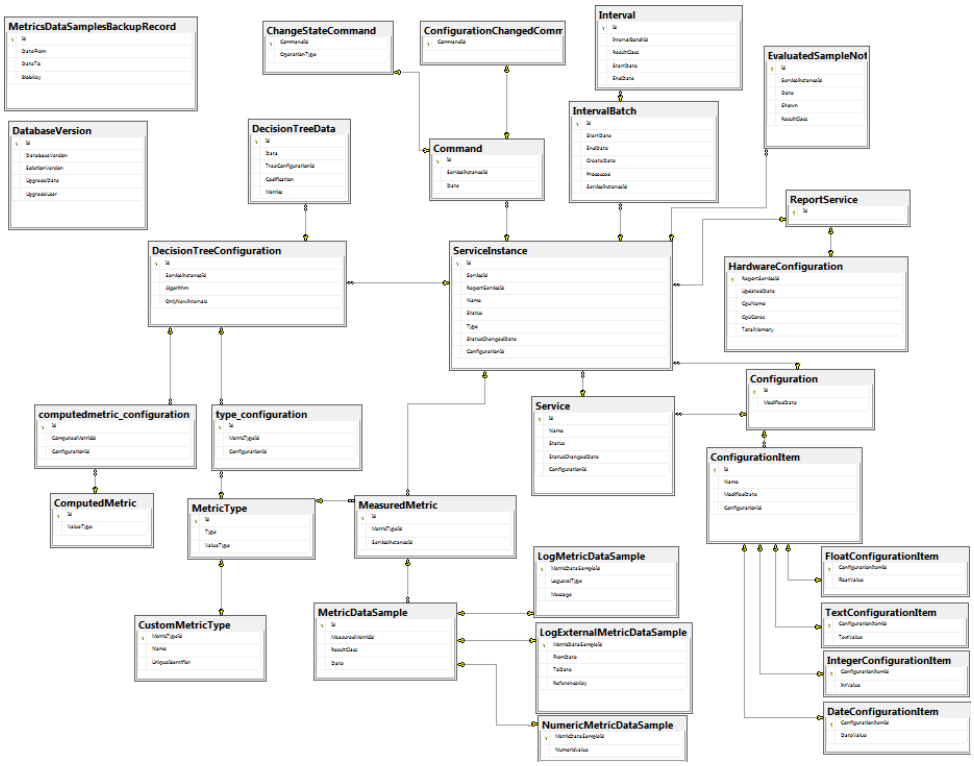


Obr. C.6: Screenshot zachytávajúci formulár pre vytvorenie novej stromovej konfigurácie.



Obr. C.7: Screenshot zachytávajúci voľbu koncovej klasifikačnej triedy po vytvorení intervalu na grafe.

C.1 Databázový model



Obr. C.8: Úplný relačný databázový model vytvoreného systému

Dodatok D

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `bin` ..... binárne a spustiteľné súbory
  - `ManagementService` ..... súbory Monitorovacej služby
  - `ReportService` ..... súbory notifikačnej služby
  - `MachineLearningService` ..... súbory služby pre tvorbu stromov
  - `WebClient` ..... súbory webového klienta
- `src` ..... zdrojové kódy
  - `Project` ..... zdrojové kódy implementácie
  - `Thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `Turcaj_Patrik_2017.pdf` ..... text práce vo formáte PDF