VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. David Šebesta

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

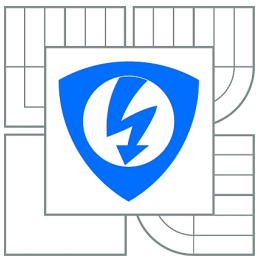
MONITOROVÁNÍ VÝPADKŮ ZKUŠEBNÍCH STANIC
MONITORING OF TEST STATION DOWNTIME

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. David Šebesta

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Lukáš Malina, Ph.D.

BRNO 2016



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. David Šebesta ***ID:*** 134625
Ročník: 2 ***Akademický rok:*** 2015/2016

NÁZEV TÉMATU:

Monitorování výpadků zkušebních stanic

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je poskytnout podklady pro Pareto analýzu a umožnit tak zvýšení využití zkušebních stanic. Práce bude obsahovat následující části: 1.Přesměrovat výpis chybových hlášení ze souborů do databáze. Soubory s výpisem chyb se nachází na řídicích počítačích u zkušebních stanic. Tyto počítače jsou v síti umístěny za ACL, přístup k nim je povolen z databázového serveru. 2. Chyby roztřídit a u každého výpadku poskytnout možnost zadání příčiny výpadku. Pouze prostoje delší než minimální čas (definovatelný) se budou zpracovávat jako samostatné položky, cílem je získat histogram prostojů dle příčiny. Chyba indikovaná při výpadku zpravidla není skutečnou příčinou. Čas prostoje bude dále tříděn do několika skupin dle příčiny výpadku. 3. Výpadek stanice nasměrovat do GSM modulu, který bude informovat vybrané uživatele o výpadku a první indikované chybě. 4. Vytvořit statistiky a analýzy chyb (např. histogramy, celkový čas prostoje, atd.). 5. Webová aplikace v ASP.NET (+VB), data budou uložena v databázi MS SQL.

DOPORUČENÁ LITERATURA:

[1] CLARKE, Gordon R, Deon REYNDERS a Edwin WRIGHT. Practical modern SCADA protocols: DNP3, 60870.5 and related systems. London: Elsevier, 2004, ix, 537 p. Engineering: instrumentation & control.
[2] McCRADY, Stuart G. Designing SCADA Application Software: A Practical Approach. Elsevier, 2013.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: Ing. Lukáš Malina, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

ABSTRAKT

Tato práce se zabývá monitorováním výpadků zkušebních stanic oddělení DS/ETC (Diesel Systems / Engineering Testing Center) společnosti Bosch Diesel s.r.o. v Jihlavě. Cílem monitorování je získání záznamu o každém prostoji, včasné upozornění na výpadek zkušební stanice stanice, umožnit zkušebním inženýrům editaci záznamu o prostoji a následně z nasbíraných dat vytvořit statistiky a Pareto analýzu. Na začátku práce bylo stručně popsáno oddělení ETC a dále pak obecně SCADA systémy. Následně byl navržen monitorovací systém na míru oddělení ETC a provedená implementace. Včasné upozorňování na výpadky bylo realizováno pomocí SMS zpráv z GSM modemu připojeném přímo k serveru. Zkušební inženýři mají k dispozici grafické rozhraní, kde mohou dané záznamy o prostoji editovat. Statistiky lze v grafickém rozhraní generovat individuálně na základě výběrového filtru. Implementovaný systém umožnil oddělení primárně se zaměřit pouze na řešení těch příčin výpadku, které povedou k největšímu snížení prostojů zkušebních stanic.

KLÍČOVÁ SLOVA

Pareto analýza, monitorování výpadků, zkušební stanice, GSM modem, AT příkazy, SMS, SCADA, ASP.NET

ABSTRACT

This thesis is about the monitoring of testing station downtimes at the DS/ETC (Diesel Systems / Engineering Testing Center) Bosh Diesel s.r.o. department in Jihlava. The goal of the monitoring is to get a record and an immediate warning for each machine downtime. The testing engineers are able to edit downtime reports and to create statistics and the Pareto analysis based on the reports. In the beginning of the thesis there are generally described the ETC department and the SCADA system. The monitoring system is designed and implemented based on the requirements of the ETC department. SMS messages sent via a GSM module what is connected directly to the server are used for immediate downtime warning. The test engineers have the graphic user interface what they can use to edit downtime reports. Statistics are generated individually based on the adjustable data filter. The department can focus primary on the finding out the reasons of the downtimes what leads to the biggest reduction of the test machine downtimes.

KEYWORDS

Pareto analysis, monitoring failures, test stations, GSM modem, AT commands, SMS, SCADA, ASP.NET

ŠEBESTA, David *Monitorování výpadků zkušebních stanic*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 79 s. Vedoucí práce byl Ing. Lukáš Malina, Ph.D

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Monitorování výpadků zkušebních stanic“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Lukáši Malinovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Zároveň bych rád poděkoval odbornému konzultantovi panu Ing. Jaroslavu Čápovi, Ph.D. za odborné rady, vysvětlení dané problematiky a jeho ochotu pomoci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešní době je všeobecná snaha všech větších firem vizualizovat proces výroby a zároveň zpracovávat co nejvíce dostupných dat z výrobního procesu. Z těchto dat lze vytvořit například podrobný záznam procesu, statistiky, grafy a nejrůznější druhy analýz. Pomocí těchto analýz lze identifikovat nejslabší místa v procesu výroby a posléze přijmout různá opatření k vylepšení celého procesu.

Systémy, které umožňují monitoring a zisk informací, jsou obecně nazývány SCADA (Supervisory Control and Data Acquisition). SCADA jsou nedílnou součástí každé moderní výroby. Řídící centrum dohlížecích systému se nazývá SCADA centrála. Z těchto central může operatér snadno reagovat na události vzniklé ve výrobě a okamžitě zasahovat do procesu výroby.

Systémů SCADA je na trhu poměrně velké množství. Některé výrobní procesy jsou ale natolik specifické, že musí být dohlížecí systém „ušit přímo na míru“, stejně jako monitorovací systém prostojů zkušebních stanic společnosti Bosch Diesel s.r.o., který je zadavatelem této práce.

Monitorovací systém zde bude sledovat výpadky zkušebních stanic a uživatelé budou moci editovat jejich skutečné příčiny. Z těchto dat následně budou vytvořeny statistiky, na základě kterých podnikne firma kroky pro minimalizaci prostojů.

Zároveň bude do monitorovacího systému zaveden modul, který bude pomocí SMS zpráv upozorňovat obsluhu na zastavení stanice.

V první kapitole práce se seznámíme s oddělením ETC, jeho činností a zkušebními stanicemi. Ve druhé kapitole podrobněji probereme Pareto analýzu a teorii SCADA systémů. Ve třetí kapitole provedeme podrobný rozbor požadavků zadavatele, rozvržení monitorovacího systému a navrhneme základní funkce našeho systému. Částečně se zde také zmíníme o technologii ASP.NET. Ve čtvrté kapitole navrhneme metodu sběru chybových hlášení ze zkušebních stanic, princip parsování staženého souboru a provedeme implementaci. V páté kapitole navrhneme metodu výpočtu prostojů a opět provedeme implementaci. V šesté kapitole vytvoříme uživatelské prostředí pro editaci záznamů o prostojích. V sedmé kapitole se v práci budeme zabývat SMS upozorňováním na výpadek zkušební stanice pomocí GSM modemu. V poslední osmé kapitole vytvoříme uživatelské prostředí pro generování Pareto analýz a statistik jednotlivých příčin výpadků.

1 ODDĚLENÍ DS/ETC JIHLAVA

Zadavatelem mé diplomové práce je společnost Bosch Diesel s.r.o. v Jihlavě. Hlavní zaměření této společnosti v Jihlavě je výroba vysokotlakých dieselových čerpadel, zásobníků stlačeného paliva (Rail) a vysokotlakých regulátorů tlaku pro vstřikovací systémy Common Rail. Společnost Bosch si od počátku své existence vždy zakládala na vysoké kvalitě a preciznosti svých produktů. Testování nových výrobků je náplní oddělení DS/ETC–Jh (Diesel Systems / Engineering Testing Center v Jihlavě). ETC v Jihlavě se zabývá především dlouhodobými zkouškami částí vstřikovacího systému Common Rail.

1.1 Dlouhodobá zkušebna

Tato zkušebna, umístěná na závodě II v Jihlavě je největší v rámci divize dieselových systémů v koncernu Bosch. Zároveň se jedná o jednu z nejmodernějších zkušeben na světě. Úkolem této dlouhodobé zkušebny je společně se zákazníky nalézt co nejlepší možné technické řešení dieselových vstřikovacích systémů s ohledem na jejich spolehlivost a robustnost. Zákazníci ETC jsou zpravidla jiná interní oddělení společnosti Bosch Diesel (např. vývojová oddělení nebo výrobní oddělení). Samotné oddělení se dělí na další tři podskupiny:

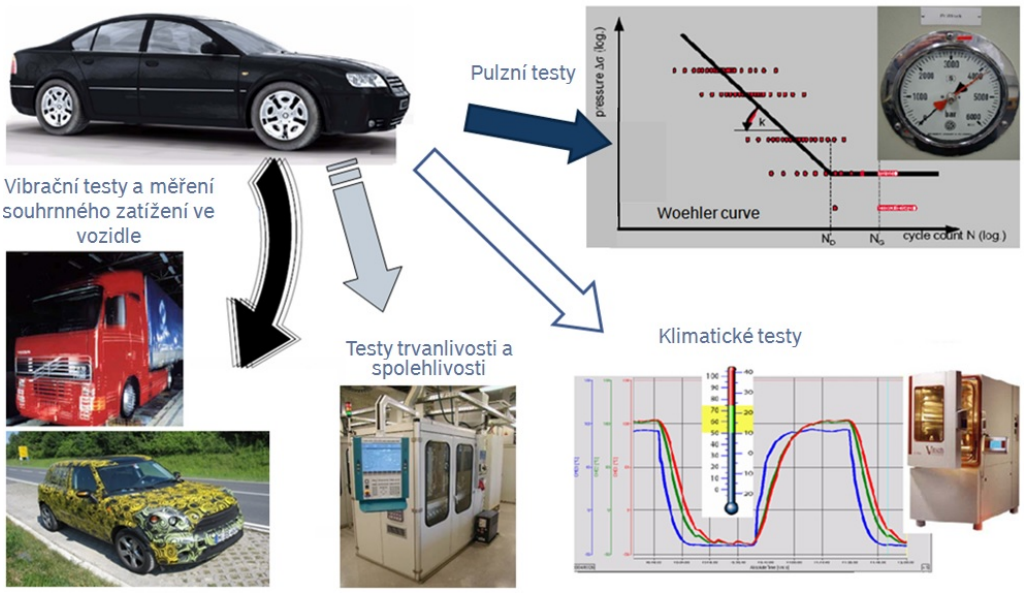
- **ETC1** dlouhodobé zkoušky CR systémů,
- **ETC2** pulzační, klimatické a vibrační zkoušky,
- **ETC3** správa a údržba stanic a zkušebního zařízení, kalibrace a funkční měření.

Ve zkušebně se provádí a zároveň navrhují dlouhodobé zkoušky tak, aby odhalily slabá místa jednotlivých komponent Common Rail systému. Při testování je systém vystaven vyšším tlakům, teplotám nebo rychlostem. Také palivo proudící systémem při zkoušce může mít z různých důvodů jiné požadované vlastnosti (např. mazivost, obsah různých příměsí). Hlavními cíli dlouhodobého testování Common Rail systému tedy je:

- prokázat životnost garantovanou prodejcem vozidla (160 tis. km až 1,6 mil. km),
- odhalit nežádoucí chování komponent,
- simulovat nejhorší možné podmínky, které mohou nastat u zákazníka,
- zajistit vysokou robustnost a spolehlivost vstřikovacího systému.

1.2 Portfolio testů prováděných oddělením ETC

Zkušební inženýři na oddělení ETC provádí čtyři hlavní principy testování komponentů:



Obr. 1.1: Hlavní principy testování

Vibrační testy a měření souhrnného zatížení ve vozidle: jedná se o souhrnné měření zatížení přímo ve vozidle dodané zákazníkem, které provádí zaměstnanci DS/ETC–Jh na zkušebním okruhu v Německém Boxbergu,

Testy trvanlivosti a spolehlivosti: zkouška určená k odvození životnosti dané komponenty,

Klimatické testy: testované díly se zde namáhají vysokými teplotami či prudkým střídáním teplot,

Pulzní testy: při této zkoušce tlakové špičky ve zkoušených dílech dosahují až 4000 barů.

1.2.1 Modely průběhu testování

Při zkouškách se používají různé modely průběhu testování s ohledem na to, jaké podmínky u zákazníka se právě snaží zkušební inženýr nasimulovat a vyhodnotit jejich důsledky na testovaný komponent.

Typy dlouhodobých zkoušek:

Konstant (KDL) – při zkoušce jsou otáčky konstantní (maximální) po celou dobu zkoušky, případně s pravidelným krátkým odlehčením,

Program (PDL) – v tomto typu zkoušky se simuluje zatížení komponent v reálném provozu typickém pro danou aplikaci,

Motor Start-Stop (MSS) – zde se simulují dynamická zatížení při zrychlování a zpomalování motoru,

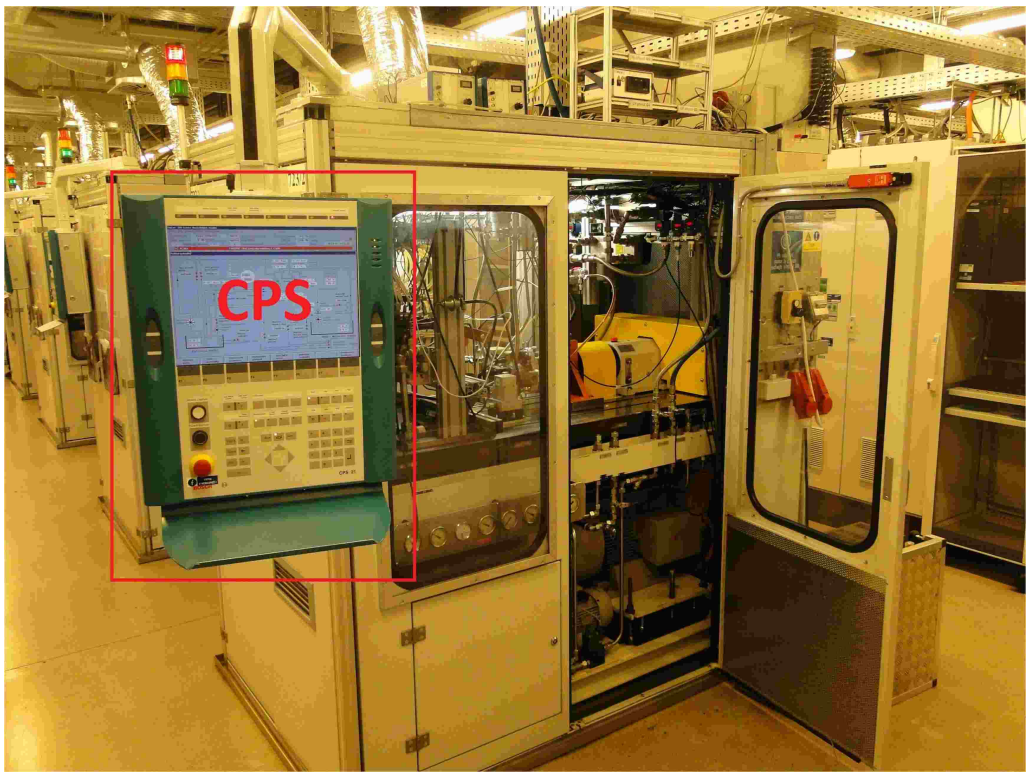
Highly Accelerated Lifetime Test (HALT) – dochází k postupnému zvyšování úrovně zatížení zkoušeného dílu (tlak, teplota paliva, otáčky),

Quantitative Accelerated Lifetime Test (QALT) – test zaměřený na vysoké zatížení v konkrétním pracovním bodě,

Highly Accelerated Stress Screening (HASS) – zde je zkoušený díl vysoce přetěžován (tlak, teplota, otáčky), doba zkoušky je krátká, kolem 10 h.

1.3 Zkušební stanice

Zkušební stanice (obr. 1.2) pro zkoušky *trvanlivosti a spolehlivosti* je navržená tak, aby v ní mohla dlouhodobá zkouška běžet bez nutnosti neustálého lidského dohledu.



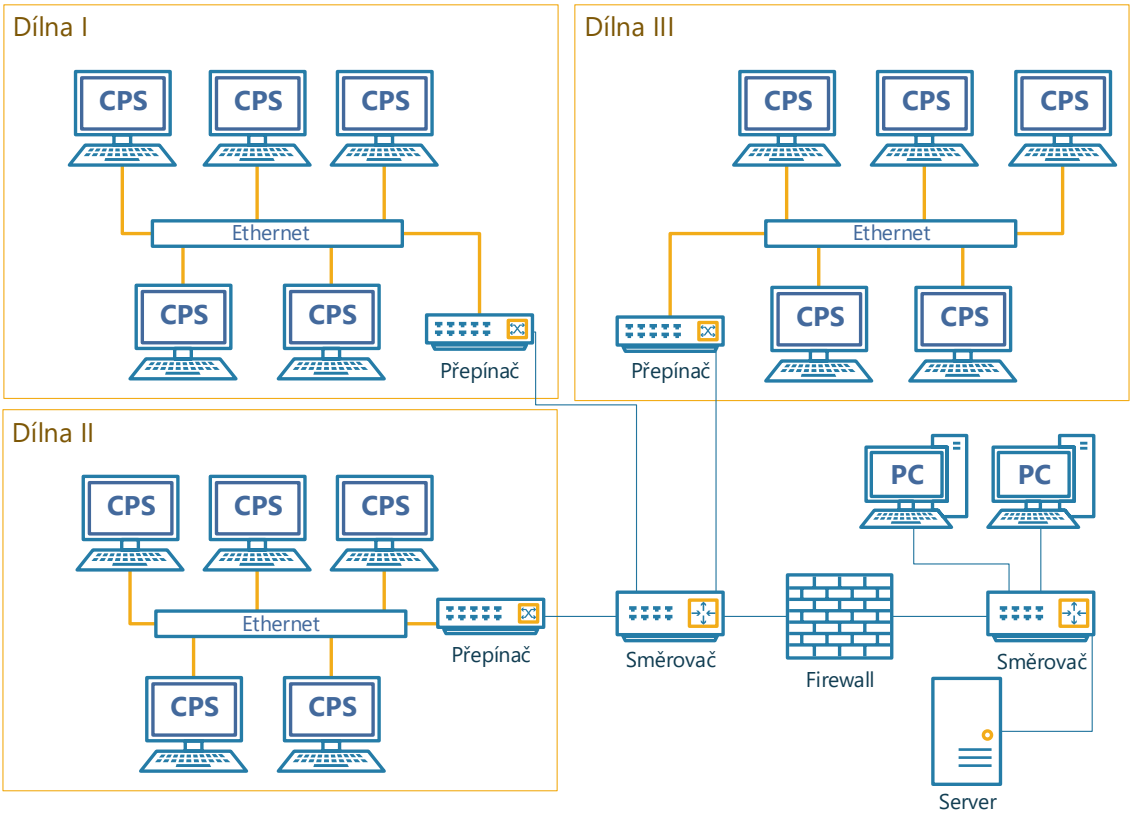
Obr. 1.2: Zkušební stanice

Každá zkušební stanice má svůj vlastní řídící počítač CPS (Control Panel System), na obrázku (1.2) je CPS orámován červeným čtvercem. Tento řídící počítač CPS má operační systém **Windows NT 4.0**.

Před každou zkouškou je zkušebním inženýrem do CPS nahrán program s průběhem dané zkoušky. Po spuštění zkoušky CPS ovládá analogové a digitální výstupy, sbírá data a zároveň hlídá překročení předepsaných limitů. Jestliže dojde k překročení některého z těchto limitů či jinému problému v průběhu zkoušky, CPS zapíše hlášení o tomto stavu do svého logovacího textového souboru, který se jmenuje *„Logfile.cps“*. Tento soubor je umístěn na pevném disku CPS, na jedné ze dvou diskových jednotek. Tato disková jednotka je sdílená do podnikové sítě prostřednictvím operačního systému CPS.

1.4 Architektura sítě

Oddělení ETC spravuje okolo devadesáti zkušebních stanic, které jsou rozmístěny ve třech samostatných „dílnách“ s omezeným přístupem. Každá jednotlivá stanice je zapojená do sítě Ethernet. Stručné znázornění sítě je vyobrazeno na obrázku (1.3).



Obr. 1.3: Rozvržení sítě

V každé ze tří dílen jsou stanice zapojeny do přepínačů v topologii hvězdy. Tyto přepínače jsou pak přes router a firewall připojeny do centrální podnikové sítě. Z bezpečnostních důvodů mají přístup do CPS přes firewall z vnější podnikové sítě pouze dva stolní PC a server, který podléhá oddělení ETC. Tento server je umístěn v zabezpečené serverové místnosti a jeho parametry jsou uvedeny v tabulce (1.1).

Tab. 1.1: Parametry serveru

Operační systém:	Windows Server 2012 R2 Standard
Procesor:	Intel(R) Xeon(R) CPU E5-2623 v3 @ 3.00 GHz
Paměť RAM:	64 GB
Typ systému:	64–bit

1.5 Nové trendy u dlouhodobých zkoušek

V poslední době je novým trendem u zkoušek *trvanlivosti a spolehlivosti* zkracování časového limitu na jednotlivé zkoušky. Důvody jsou zřejmé, zkrácení času na vývoj a zlevnění testování. Zatímco dříve se testoval celý vstřikovací systém až 3000 hodin nebo do konce své funkčnosti „End-of-Life“, nyní jsou cílem testování robustní jednotlivé komponenty a časové limity na jednotlivé zkoušky se zkracují. Zároveň s rostoucí technologickou náročností se zpřísňují limity pro předepsané parametry testu:

- teploty jednotlivých částí hydraulického okruhu a zkoušeného komponentu,
- teplota kapaliny vstupující do vstřikovacího systému,
- otáčky pohonu palivového čerpadla,
- tlaky kapaliny v různých částech CR systému,
- průtoky kapalin na přívodu nebo přepadu palivového čerpadla.

Jestliže při běžící zkoušce dojde k překročení některého z předepsaných limitů či vznikne jiný technický problém, dojde k okamžitému zastavení zkušební stanice.

Test stojí, dokud zkušební inženýr neprovede analýzu výpadku a neurčí příčinu přerušení zkoušky. Poté zpravidla dojde k odstranění příčiny výpadku a test je znovu spuštěn. V důsledku zpřísňování limitů předepsaných parametrů dochází k častějšímu překračování povolených mezních hodnot. Čas, po který test neběží a čeká na analýzu, odstranění příčiny a opětovné spuštění, se nazývá „prostoj“.

Z ekonomického hlediska a hlediska rychlosti vývoje je nutné tyto prostoje minimalizovat, protože kvůli nim dochází k časovým a finančním ztrátám. Na oddělení ETC je zhruba devadesát zkušebních stanic pro dlouhodobé testy a okolo dvaceti zkušebních inženýrů. Každý inženýr má na starosti několik zkušebních stanic. Zatím

neexistuje všeobecný přehled o nejčastějších příčinách prostojů. Na základě všeobecného přehledu a pomocí „Pareto analýzy“, by mohla být přijata opatření, která by ve svém důsledku vedla ke snížení počtu výpadků stanic. Tím by mělo dojít ke snížení celkové doby prostojů a zvýšení celkové efektivity zkušebního procesu.

Proto se pokusíme navrhnout systém, který bude pravidelně sbírat data z *„Logfile.cps“* na stanicích, tato data statisticky vyhodnocovat a následně je vizualizovat. Systém by také měl v co nejkratší době upozorňovat odpovědné osoby na zastavení běhu jejich stanice, aby se docílilo zrychlení reakce zkušebních inženýrů a tím tedy i zkrácení prostojů.

2 SLEDOVACÍ SYSTÉMY A ANALÝZY

Pro snížení prostojů je nutné podniknout taková opatření, aby k přerušování běžících zkoušek docházelo co nejméně. Řešení všech možných příčin simultánně je značně neefektivní. Proto je nutné, nejdříve identifikovat prioritní problémy, které vedou k nejčastějším pádům stanic a začít s odstraňováním právě těchto problémů. Které to jsou, zjistíme právě pomocí „Pareto analýzy“.

2.1 Pareto analýza

Tento nástroj nám pomůže vyjádřit relativní významnost jednotlivých příčin vedoucím k pádům stanic.

2.1.1 Princip Pareto analýzy

Myšlenkami „Paretovy analýzy“ se již v roce 1897 začal zabývat italský ekonom Vilfredo Pareto. Přišel s tezí, že 80 % bohatství země je v rukou 20 %lidí [1]. O definici a následnou popularizaci „Pareto analýzy“ se v roce 1941 postaral Joseph Moses Juran, který jeho myšlenku

> „za 80 % problémů může 20 % příčin“

modifikoval téměř do všech oblastí lidské činnosti. Pro ukázku uvedeme některé z jeho tezí [1] [2]:

- 20 % všech našich činností přináší 80 % zisku,
- 80 % příjmů získáte od 20 % zákazníků,
- 80 % tržeb vznikne prací 20 % zaměstnanců,
- 20 % vašich přátel stojí za 80 % vašeho zájmu,
- 80 % zmetků ve výrobě způsobuje 20 % příčin,
- 80 % odpočinku vám přinese prvních 20 % dovolené,
- 80 % znalostí jsme získali za prvních 20 % vynaloženého času,
- 80 % dat přenese v datové síti 20 % uživatelů,
- 80 % požadavků na infolinku vygeneruje 20 % zákazníků.

Joseph M. Juran také tuto myšlenku aplikoval na oblast řízení kvality. Došel k zajímavému a pro nás důležitému zjištění, že [1]:

> „**80 % odstávek výroby je způsobeno 20 % zařízení továrny.**“

Je-li toto tvrzení pravdivé, pak nemá smysl zabývat se všemi příčinami prostojů naráz, ale je vhodné se intenzivně zaměřit pouze na určitou část, kterou nám napoví právě Paretova analýza.

2.1.2 Postup Pareto analýzy

Pareto analýza se nejčastěji provádí v těchto sedmi krocích. Postup si ukážeme na příkladu [2]:

1. **Definování místa analýzy** – výběr procesu činnosti, kde chceme zvýšit zisk nebo efektivitu. Pro náš příklad to budou poruchy ve výrobě.
2. **Sběr dat** – pro analýzu je zapotřebí získat relativní data a následně vytvořit tabulku. V našem příkladě je to sloupeček s četností jednotlivých závad (tab. 2.1).

Tab. 2.1: Data Pareto analýzy

Závada	Četnost	Relativní četnost	Kumulativní rel. četnost
C	82	0,4293	0,4293
A	52	0,2723	0,7016
D	31	0,1623	0,8639
B	15	0,0785	0,9424
F	6	0,0314	0,9738
E	3	0,0157	0,9895
G	2	0,0105	1,0000
Celkem	191	1,0000	—

3. **Uspořádání dat** – získaná data seřadíme od největšího výskytu sledované hodnoty po nejmenší. Z těchto dat vytvoříme histogram (obr. 2.1), který odpovídá již dříve seřazené tabulce četností.
4. **Lorenzova kumulativní křivka** – Lorenzova křivka je spojnicovým grafem kumulativních relativních četností. Nejdříve si musíme vypočítat relativní četnost jednotlivých chyb pomocí vzorce (2.1) a poté kumulativní relativní četnost, což je součet relativních četností dané poruchy a všech poruch s vyšší četností než právě počítaná porucha, pomocí vzorce (2.2). Výsledná data opět vyneseme do grafu (obr. 2.1) jako spojnicový graf kumulativních relativních četností.

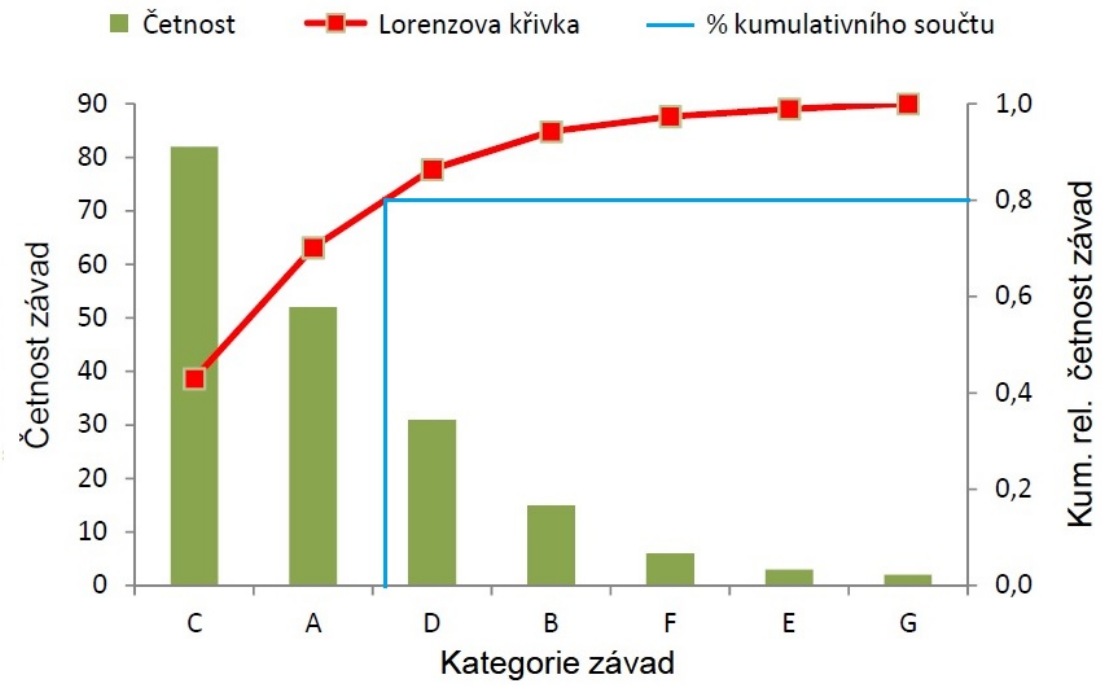
$$Rel\_\check{C}etnost(C) = \frac{\check{C}etnost(C)}{\check{C}etnost(celkem)} \tag{2.1}$$

$$K\_\check{C}etnost(A) = K\_\check{C}etnost(p\check{r}edchoz\acute{i}) + Rel\_\check{C}etnost(A) \tag{2.2}$$

5. **Stanovení kritéria rozhodování** – nyní si vybereme kolik procent prostojů bychom rádi odstranili. Nemusíme se vždy držet Paretova pravidla 80/20.

Mohlo by nám například stačit odstranění pouze 50 % poruch. My v našem příkladě zůstaneme u pravidla 80/20.

6. **Identifikování hlavních příčin** – Z hodnoty 0,8 vyneseme čáru na kumulativní Lorenzovu křivku (obr. 2.1). Z ní pak spustíme svislou čáru, která nám oddělí ty případy, příčiny, kterými se máme zabývat. V našem příkladě nám vyšlo,že pro odstranění cca 80 % poruch se musíme prioritně zabývat poruchami typu C a A.
7. **Stanovení nápravných opatření** – V tomto kroku Pareto analýzy se vyvodí konkrétní důsledky, které by měly zamezit vznikům poruch typu C a A.



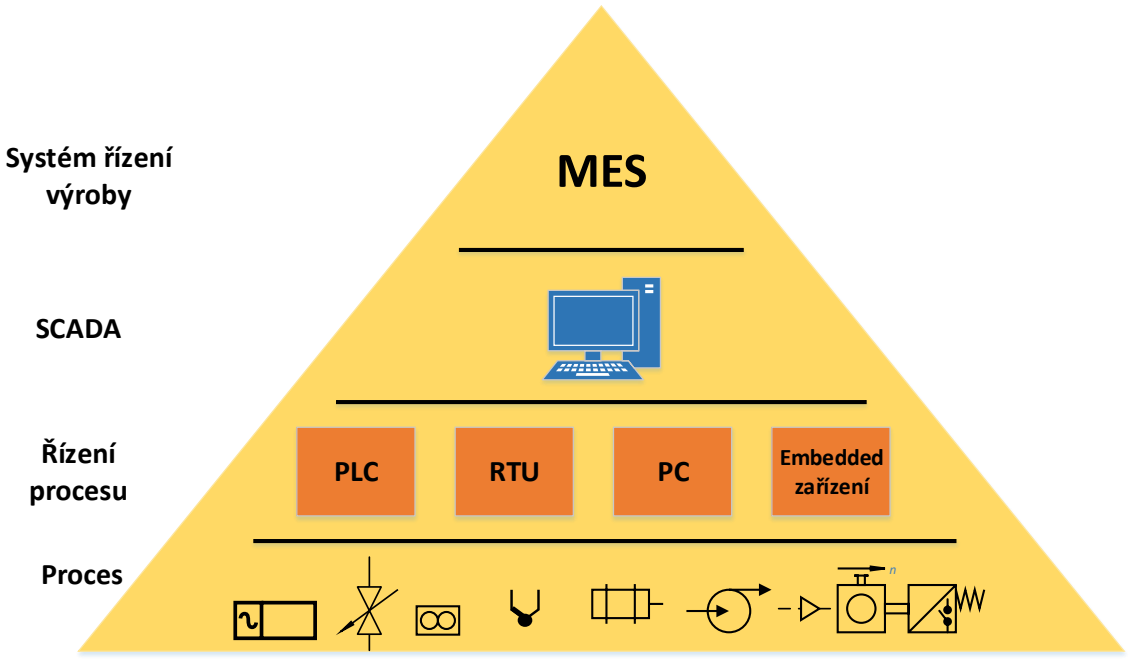
Obr. 2.1: Graf Pareto analýzy

2.2 Sledovací systémy pro sběr dat

První bod v postupu Pareto analýzy (Definování místa analýzy) je jasný, budeme sledovat zkušební stanice pro zkoušky *trvanlivosti a spolehlivosti*. Dalším bodem v postupu je sběr dat. V průmyslu se všeobecně k tomuto účelu používají SCADA (Supervisory Control and Data Acquisition) systémy.

2.2.1 SCADA systémy

SCADA systém je součást řídícího systému podniku (obr. 2.2). SCADA všeobecně v průmyslu umožňuje operátorům sledovat technologický provoz [3].



Obr. 2.2: Pyramida řídícího systému podniku

Operátoři vidí přehledně průběh celého procesu z jednoho místa a v případě potřeby mohou operativně reagovat konkrétními povely pro řídící systémy. Tento systém dokáže v reálném čase sbírat data z čidel v provozu, shromažďovat je a dále s nimi pracovat.

Základní architektura SCADA systému [4]:

Čidla a akční členy: pohony, snímače, čítače, spínače, ...

Řízení procesu: nejpoužívanějšími řídícími členy procesu jsou PLC (Programmable Logic Controller) tedy programovatelné automaty, které snímají hodnoty čidel a řídí akční členy podle předem naprogramovaného modelu. RTU (Remote Terminal Unit) mají podobnou funkci jako PLC ale pracují většinou bezdrátově ve větším zeměpisném měřítku. Použít také lze klasické PC nebo některé embedded zařízení, což je jednoúčelový systém, ve kterém je řídící počítač optimalizovaný a zcela zabudovaný do konkrétního zařízení.

Komunikační síť: komunikační síť mezi jednotlivými řídícími členy procesu. Může mít různou architekturu (např.: Profibus, Ethernet, Modbus, Radiomodem, GSM, RS–232, RS–485, ...).

Centrála SCADA systému: jedná se o centrální počítač se speciálně navrženým programem. Tento počítač musí mít přístup do sítě s PLC a RTU. Pomocí programu (nejpoužívanější jsou InTouch, Web Studio, Genesis32, WinCC, TIRSWeb ...) se pak připojuje na jednotlivá PLC, kde může vyčítat aktuální stavy vstupů, výstupů i proměnných a také je nastavovat. Ve výsledku může jedna SCADA centrála ovládat až několik tisíc vstupů a výstupů.

SCADA systém spadá pod informační systém MES (Manufacturing Execution System), který je v celkovém informačním systému podniku propojovacím článkem mezi řízením technologických procesů a ostatními informačními systémy.

2.2.2 Bezpečnost SCADA systémů

Sítě SCADA jsou budovány především pro maximální spolehlivost a lze je tedy používat značně dlouho. Z toho důvodu, především starší systémy neobsahují mechanismy chránící před neoprávněným přístupem nebo vložením škodlivého kódu. Jakmile by hacker pronikl do systému má možnost sledovat celý systém, poškodit ho a případně i kompletně zastavit provoz. V případě takzvaných „páteřních“ SCADA systémů (obslužné systémy elektráren, vodohospodářství a dalších životně důležitých systému pro chod státu) by úspěšné napadení znamenalo možné ochromení ekonomiky státu [6] [8].

Proto je třeba zavádět nejrůznější druhy zabezpečení jako je například fyzické oddělení SCADA sítě od zbytku podnikové sítě, případně oddělení pomocí VLAN sítí. Aby mohl útočník do takto fyzicky oddělené sítě proniknout, musel by se k dané síti fyzicky dostat a připojit se do ní. V případě propojení průmyslové sítě s běžnou podnikovou sítí, či nutnosti vzdáleného přístupu je nezbytné použít silný firewall, který bude účinně filtrovat provoz na perimetru průmyslové sítě. Firewall by měl zamezit neoprávněnému přístupu na jakýkoli prvek v síti, případně zabránit zkolabování sítě způsobeným některým ze zahlcovacích útoků [6] [7].

Důležitou součástí SCADA systému je také autentizace, ať už je to autentizace personálu, který má do systému přístup, ale i ostatních systémů co se SCADA systémem spolupracují. Novým trendem je i autentizace mezi akčním členem a řídícím prvkem procesu. Aby například nemohlo dojít k neoprávněné výměně snímače za snímač, který byl upraven útočníkem. Běžnou součástí SCADA systému, na ne tak úplně zabezpečené síti, je i kompletně šifrovaná komunikace, která zabraňuje únikům citlivých dat.

Nezbytná je i adekvátní bezpečnostní politika pracoviště. Personál by měl být dostatečně a pravidelně školen o pravidlech provozu a chování na pracovišti. Na všech přístupných zařízeních v provozu by měly být zakázány veškeré nepotřebné

komunikační porty. Personál by také měl mít přísný zákaz připojovat jakékoli vlastní paměťové či elektronické zařízení do podnikové sítě.

3 ROZBOR A NÁVRH SYSTÉMU

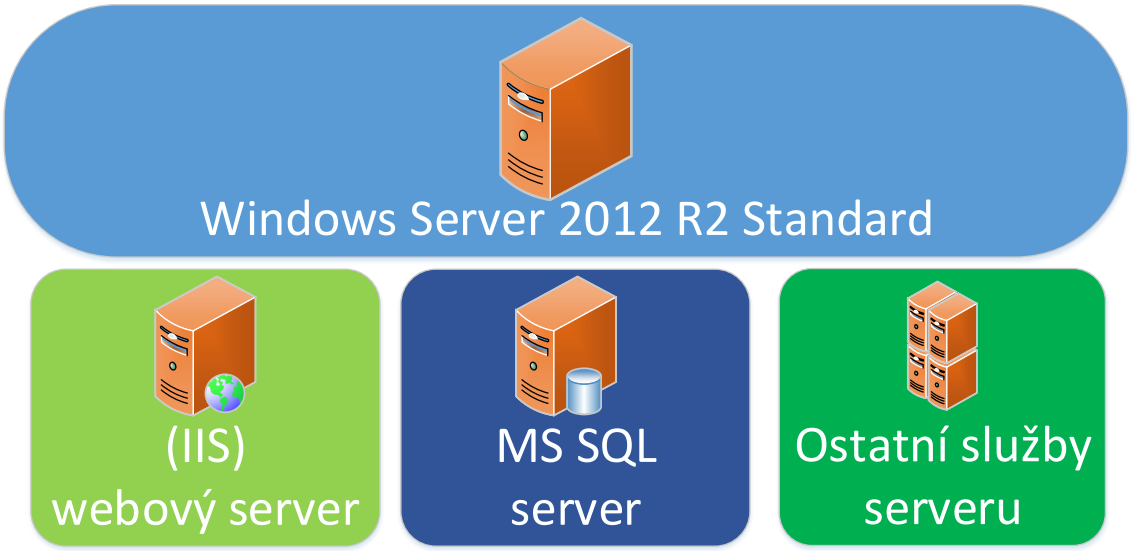
Nyní se již budeme zabývat konkrétním návrhem systému pro oddělení ETC tak, aby jsme co nejvíce dodrželi požadavky zadavatele a napomohli tím oddělení k sofistikovanějšímu přehledu o průběhu zkoušek.

3.1 Rozbor požadavků oddělení ETC

V našem případě bylo hlavním požadavkem zadavatele celý monitorovací systém, který bude sbírat data z *„Logfile.cps“* na zkušebních stanicích a dále je zpracovávat, realizovat na serveru oddělení ETC. Tento požadavek vznikl na základě toho, že oddělení se snaží různorodé systémy a postupy, které jsou potřeba k hladkému chodu celého oddělení, sjednotit pod jednotnou správu právě na tomto serveru. Výhodou tohoto řešení pro nás je, že tento server má přístup do CPS u jednotlivých stanic.

3.1.1 Integrace do stávajícího systému

Jak je znázorněno na obrázku (3.1), je na serveru jako jedna z hlavních služeb spuštěna služba IIS (Internet Information Services). Jedná se o softwarový webový server vytvořený společnosti Microsoft. Na webovém serveru je hostován informační sys-



Obr. 3.1: Služby serveru

tém „CML (Centrální Mozek Lidstva)“. CML je informační formulářový web pro oddělení ETC, psaný technologií ASP.NET (Active Server Pages) a pomocí programovacího jazyka Visual Basic. Ukázka vzhledu CML je na obrázku (3.2). Jelikož náš

systém bude integrován právě do CML, dalším požadavkem tedy je, aby náš systém byl psán pomocí stejné technologie jako výše zmíněné CML. Pro ukázku je pomocí technologie ASP.NET realizován například komerční systém TIRSWeb [5] a mnohé další SCADA systémy.

Jako úložiště dat CML používá MS SQL Server 2014 (Microsoft Structured Query Language). Jedná se o velmi výkonnou a bezproblémovou relační databázi vyvinutou společností Microsoft. Tento MS SQL server je nainstalován jako služba na našem serveru (3.1). Kdybychom do našeho systému nasadili databázový systém od jiného výrobce, nesmyslně bychom správcům zkomplikovali údržbu a orientaci v systému. Z těchto důvodu je zřejmé, že náš systém bude využívat tutéž relační databázi systém jako CML.

Obr. 3.2: Ukázka grafiky CML

3.1.2 Třídění výpadků podle příčiny

Dalším požadavkem zadavatele je, aby u každého vzniklého prostoje byla možnost přiřazení skutečné příčiny, tedy co způsobilo tento konkrétní prostoj. Přiřazení příčiny ke konkrétnímu prostoji bude provádět inženýr, který je zodpovědný za danou stanici, na které vznikl prostoj. Případně budou prostoje zpracovávány hromadně na ranních poradách.

Po přiřazení příčiny výpadku je ještě nutné rozdělit celkový prostoj do jednotlivých podskupin. Tento čas opět budou rozdělovat zkušební inženýři. Rozdělení prostoje do podskupin je nutné z toho důvodu, že každá stanice může být v provozu denně 24 hodin. Když je v provozu po celou tuto dobu, naúčtuje se zákazníkovi tento

čas jako strojní hodiny. Když není v provozu celých 24 hodin, pak se vzniklý prostoj podle příčiny buďto zákazníkovi účtuje se sníženými sazbami (montáž a ladění systému, čekání na díly) nebo se neúčtuje vůbec (výpadky infrastruktury, poruchy stanice, údržba). Zároveň slouží rozepisování hodin do kategorií pro interní účely ke sledování, zda jsou plněny cíle pro dané období, např. minimalizace prostojů kvůli mezizkouškám[1].

V informačním systému CML (jak je vidět na obrázku 3.2), je kladen velký důraz na jednoduchost celého systému. To znamená, že ovládání našeho systému by mělo být intuitivní bez ovládání zbytečných funkcí navíc. V momentě, kdy se zkušební inženýr rozhodne přiřadit příčiny prostoje a rozdělit čas, mělo by vést k uskutečnění tohoto kroku co nejméně kliknutí myší. Zároveň je ale nutné dodržet přehlednost. Musíme tedy zvolit vyvážený poměr mezi přehledností a rychlostí jednotlivých kroků v postupu přiřazování prostojů.

3.1.3 Informování o výpadku

Pro efektivní snížení prostojů vznikl požadavek, aby v případě, kdy dojde k zastavení zkušební stanice, byl zodpovědný inženýr v co nejkratší době o této skutečnosti informován. Navrhli jsme tedy řešení, že při vzniku prostoje na zkušební stanici, budou zodpovědné osoby informovány pomocí SMS (Short Message Service) zprávy. Tyto SMS zprávy budou odesílány pomocí GSM modemu.

Nezbytnou nutností je také umožnit zodpovědným osobám poskytnout možnost nastavení, kdy chtějí být SMS zprávami upozorňováni a kdy naopak nechtějí.

Mělo by jít tedy o volbu:

- rozmezí hodin během dne kdy chtějí být upozorňováni,
- zahrnutí či vyloučení víkendů do upozorňování,
- komplexní vypnutí tohoto upozorňování v případě státních svátků,
- vypnutí upozornění při dovolené.

Funkce odesílání zpráv přes tento GSM modul by mělo být přístupné k dalšímu využití za jinými účely.

Mohlo by se jednat například o:

- upozornění mechanika na pravidelnou údržbovou činnost,
- upozornění na vypršení kalibrace měřícího přístroje,
- interní sdělení.

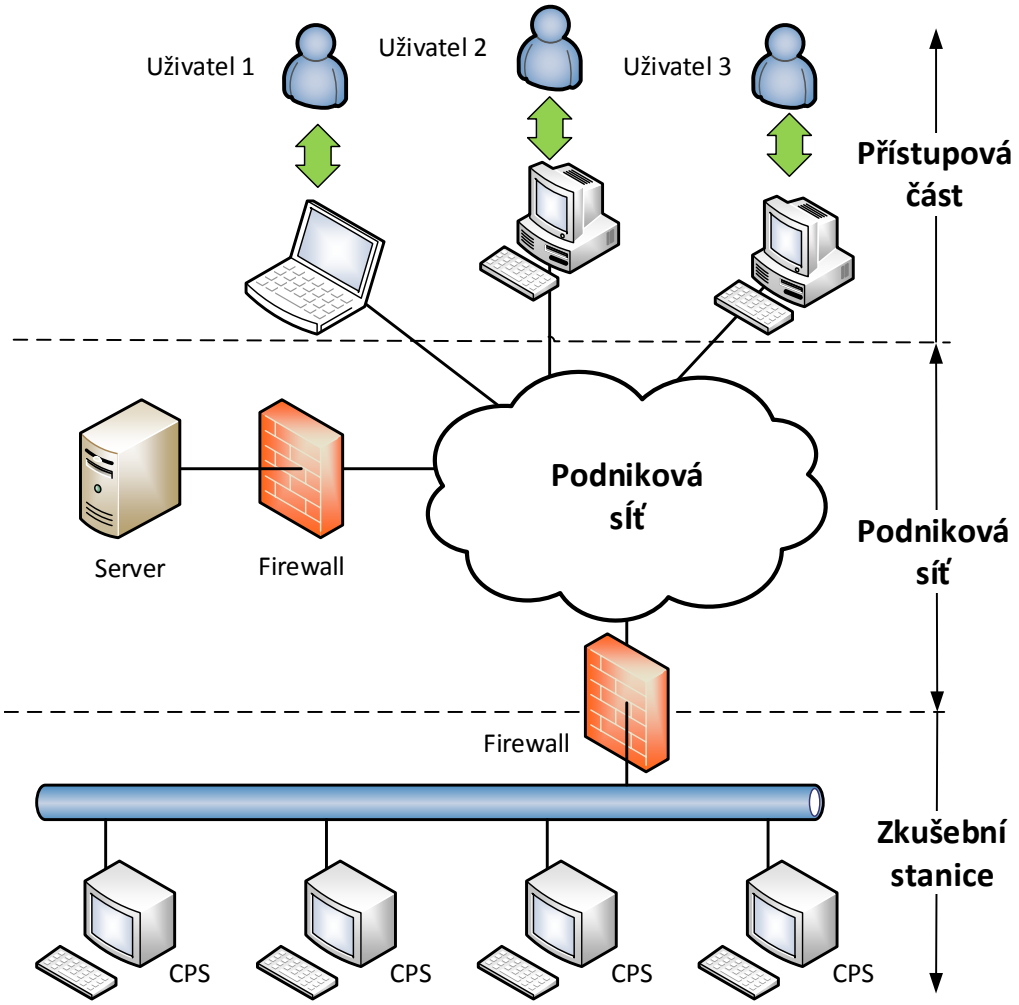
[1]Mezizkouška je funkční test dílů prováděný na specializovaném zkušebním zařízení. Provádí se několikrát v průběhu dlouhodobé zkoušky v pevně daných intervalech.

3.1.4 Zpracování dat

Požadovaným výstupem našeho systému je vytvoření statistik a analýz. Konkrétněji je úkolem vytvořit určitou formu výše zmíněné Pareto analýzy. Dále pak celkové časy prostojů kvůli konkrétním přiřazeným příčinám. U statistik bude volba časového intervalu. Díky tímto statistikám pak bude zřejmé, kam má firma směřovat svoji pozornost, případně podniknout další kroky, které budou pro snížení prostojů nejúčinější.

3.2 Rozvržení monitorovacího systému

Na obrázku (3.3) je znázorněno rozvržení našeho monitorovacího systému. Tato topologie sítě je již v provozu a nic na ní prozatím měnit nebudeme.



Obr. 3.3: Rozvržení systému

3.2.1 Zkušební stanice

Síť se zkušebními stanicemi je silně hlídaná firewallem, přes který v podstatě dovoluje přímý přístup pouze serveru a dvěma dalším zabezpečeným počítačům. Toto silné zabezpečení je nezbytné z toho důvodu, že CPS u měřicích stanic běží na operačním systému Windows NT 4.0.

Tento systém byl uveden na trh v roce 1996 a jeho oficiální podpora skončila již v roce 2004. Také zde není nainstalován a pravidelně udržován antivirový program. Samostatná obranyschopnost tohoto operačního systému je logicky mizivá. Případný přechod na novější operační systém v CPS je pro oddělení ETC poměrně velkým oříškem, protože aplikace potřebné pro chod zkušební stanice a průběh zkoušky, jsou psané na míru právě pro tento operační systém.

Jak již bylo řečeno, CPS ukládají údaje o chybách a překročených limitech do *„Logfile.cps“*. Tento soubor je v operačním systému umístěn na sdílené diskové jednotce *D*. Pro přístup na tuto diskovou jednotku je nutné znát přístupové jméno a heslo. Tyto přihlašovací údaje jsou pro všechny stanice totožné. Pro zvýšení bezpečnosti systému by bylo možné vytvořit pro každou stanici unikátní přihlašovací údaje.

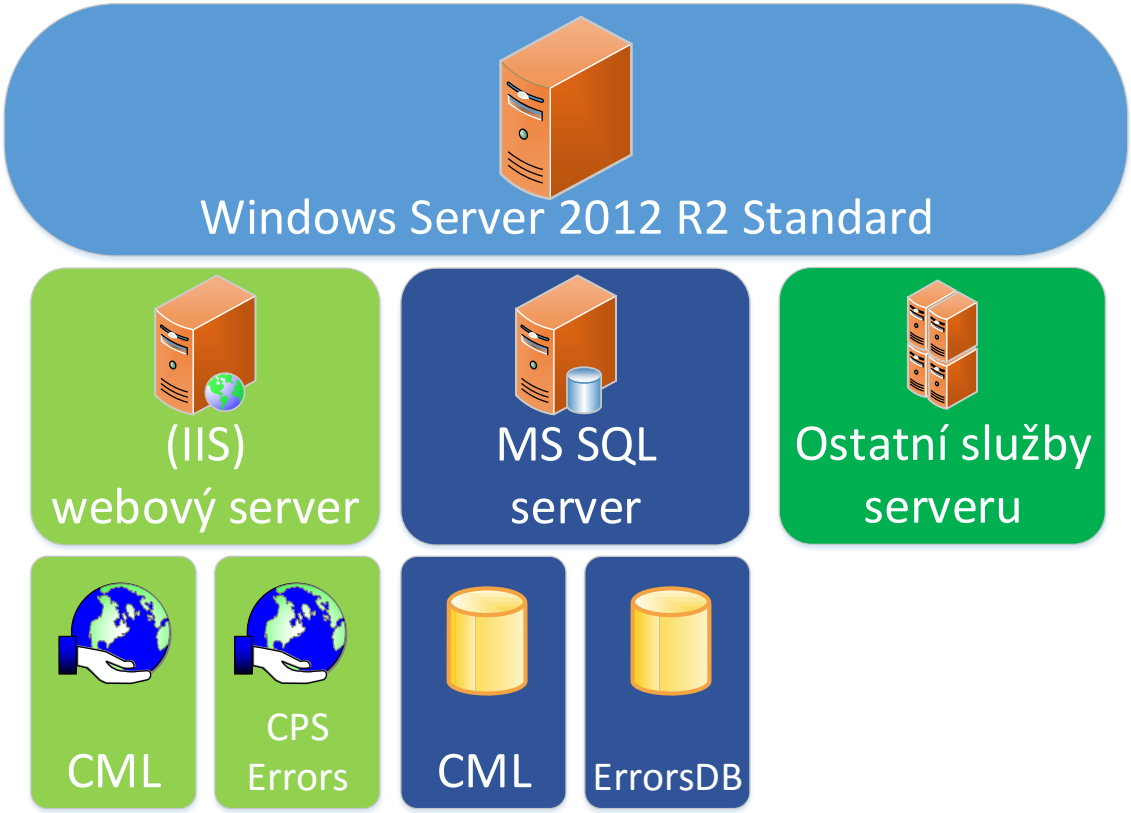
Náš systém tedy bude vždy po určitém časovém intervalu tento soubor prozkoumávat a hledat nově vzniklé události. Následně tyto události zpracuje a uloží do databáze. Pro snížení náročnosti tohoto procesu na síťovou komunikaci nebude tento soubor prohledáván na stanici, ale soubor bude vždy nejprve stažen na server a zpracován až na serveru. Tento postup je také nezbytný z hlediska délky časového intervalu, po který přistupujeme k souboru v CPS. Existuje minimální šance, že CPS se rozhodne zapsat novou událost do souboru v momentě, kdy k němu přistupuje právě náš systém. CPS tedy již nebude moci soubor otevřít, tím pádem dojde k neočekávané chybě a zastavení průběhu zkoušky. Proto je tedy nutné dobu našeho přístupu minimalizovat.

3.2.2 Podniková síť

Podniková síť společnosti Bosch Diesel je celkově velmi kvalitně zabezpečená. Veškerá komunikace dovnitř i ven z podnikové sítě jde přes hlídaný proxy sever umístěný v Německu. Náš server, který je v podstatě naše SCADA centrála, je umístěn za dalším firewallem, který odfiltrovává škodlivou komunikaci a chrání tím bezpečnost serveru.

Pro vývoj a zkušební provoz našeho systému bude na serveru vytvořen nový webový systém s názvem *CPS_Errors*. Jak je vidět na obrázku (3.4), tento systém poběží paralelně k systému CML. Stejně tak vytvoříme novou databázi s názvem

ErrorsDB. Tento postup jsme zvolili z toho důvodu, abychom nijak neohrozili hladký provoz systému CML a jeho data uložená v databázi CML.



Obr. 3.4: Služby serveru

Náš systém bude provádět:

- navázání spojení s CPS a stažení souboru *„Logfile.cps“*,
- vyhledání novinek v tomto souboru, zpracování a uložení do databáze,
- hlídání pravidelného opakování tohoto procesu,
- hlídání provozu stanic a počítání prostojů,
- umožní zkušebním inženýrům prohlížení jednotlivých prostojů,
- umožní zkušebním inženýrům přiřazení příčiny prostoje,
- umožní zkušebním inženýrům rozdělení času prostoje,
- upozorní na výpadek stanice pomoci GSM modemu,
- obsluha GSM modemu,
- na požadavek vygeneruje statistiky nebo analýzy.

3.2.3 Přístupová část

Přístupová část jsou počítače oddělení ETC, na které se uživatelé přihlašují pomocí WIS (Windows Integrated Security), což je celopodnikové doménové zabezpečení. Každý zaměstnanec má svůj účet, na který se přihlašuje pomocí jména a hesla na kterémkoli počítači v podnikové síti, který je určen právě pro uživatele. Zároveň jsou ke každému účtu přiděleny určitá oprávnění, které určují pravomoce daného uživatele, tedy i kam má přístup a kde mu bude přístup zamítnut. Po přihlášení se pak přes webový prohlížeč může připojit do našeho systému CPS_Errors.

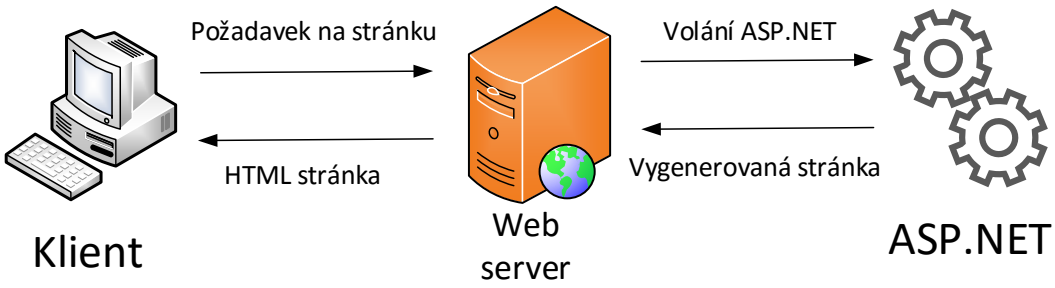
Ze zřejmých důvodů není žádoucí, aby do našeho systému mohli nahlížet všichni zaměstnanci společnosti BOSCH, proto budeme muset celopodnikové doménové zabezpečení WIS částečně využít i v našem systému. Tím bude zvýšena bezpečnost celkového systému.

3.3 Použitá technologie ASP.NET

ASP.NET je webový framework. Jedná se o sadu knihoven, která nám umožní tvorbu webových aplikací v jazyce VB. Tyto knihovny obsahují řešení základních úloh, které ve webových aplikacích vystávají [9].

3.3.1 Základní princip technologie ASP.NET

Technologie je založena na architektuře klient – server. Aplikace v ASP.NET je tedy program, jehož výstupem je HTML stránka. ASP.NET tedy běží na serveru, na základě požadavků od klienta vygeneruje webovou stránku a pošle ji klientovi. Ten vidí už jenom výsledné HTML (HyperText Markup Language), ve které se technologie ASP.NET již nevyskytuje [9]. Grafické znázornění je na obrázku (3.5).



Obr. 3.5: Princip technologie ASP.NET

3.3.2 Vytváření webových aplikací pomocí WebForms

WebForms je takové přenesení standardní formulářové desktopové aplikace na web. Aplikace se navenek chová jako desktopová, ale na pozadí je skrytá složitá logika, protože protokol HTTP (HyperText Transfer Protocol) je bezstavový [9] [10].

Stav jsou vyplněná data v polích, případně zaškrtnuté checkboxy a mnohé další. Pokud tedy například vyplníme formulář jen z poloviny a odešleme ho na server, tak ve stránce, kterou nám odeslal server jako odpověď většinou chceme, aby zůstaly tyto hodnoty vyplněny. Stav formuláře se tedy musí uchovávat. U ASP.NET toto obstarává takzvané ViewState. ViewState není nic jiného, než skryté formulářové pole, které na výsledné HTML stránce uchovává stav aplikace a odešle ho s dalším dotazem serveru, který u sebe obnoví stav aplikace, jaký měla předtím, provede nějakou akci a opět vygeneruje ViewState a stav zapomene. Malou nevýhodou je zvýšená velikost HTML stránky, protože je přenášeno i pole WiewState [9] [10].

Každá stránka v ASP.NET aplikaci obsahuje jednak HTML se serverovými komponentami, tak i kód v pozadí (tzv. code–behind), který ošetřuje události komponent a stará se o funkcionalitu celé stránky. Tento kód je vždy mimo HTML. Máme tedy oddělen vzhled a obsah aplikace od aplikační logiky, díky čemuž je aplikace přehlednější [10].

4 NÁVRH METODY SBĚRU A PARSOVÁNÍ DAT A JEJÍ IMPLEMENTACE

V této části práce se budeme zabývat přístupem na stanici, tedy do její CPS a následovnému stažení *„Logfile.cps“*. Dále pak provedeme rozbor souboru a navrhneme postup zpracování a uložení dat do databáze. V neposlední řadě provedeme implementaci tohoto navrženého postupu zpracování.

Hlavním důvodem sběru těchto dat je, že uživatel nutně potřebuje vědět v kolik hodin vypnutí stanice proběhlo, a jaké všechny chyby byly vygenerovány při a po vypnutí stanice k tomu, aby dokázal rozhodnout, co vlastně bylo příčinou zastavení zkušební stanice.

4.1 Stažení souboru

Pro práci se sítí má programovací jazyk VB.NET integrovanou třídu `Network`. Ta obsahuje metodu `DownloadFile`, která může být použita ke stažení vzdálených souborů a jejich uložení na lokální disk.

Parametry metody `DownloadFile`:

- cesta ke zdroji a cesta, kam se má soubor uložit
- přihlašovací jméno a heslo
- zapnutí či vypnutí informování o průběhu stahování
- povolený časový limit pro pokus o připojení v milisekundách
- zapnutí či vypnutí přepisování souboru v cíli.
- Syntaxe:

```
Network.DownloadFile(("\\10.10.10.10\d\OpconData\Logfile.cps"),
       _"D:\Logfile.cps", "Login", "Pasw", False, 1000, True)
```

Po implementaci byl tento přístup k souborům na stanici velice nespolehlivý. Docházelo k častým potížím při připojování. Pomineme-li, že u dané stanice se nepovedlo stáhnout aktuální *„Logfile.cps“*, ale také opakovanými pokusy o stažení se neúměrně protahoval čas, potřebný k jednomu cyklu aktualizace dat ze všech stanic.

Proto jsme zvolili jiný přístup. Nejdříve si pomocí třídy `Process` , která umožňuje přístup k systémovým procesům a třídy `ProcessStartInfo` sloužící k definici parametrů třídy `Process`, otevřeme spojení s adresářem na stanici, ve kterém je umístěn požadovaný soubor.

```
ProcessStartInfo.FileName = "net"
ProcessStartInfo.Arguments = "use \\"+IP+"\d\Opcondata USER:Log Pas
System.Diagnostics.Processs.Start(ProcessStartInfo)
```

Tento proces *„net use“* se běžně používá k namapování síťového disku na konkrétní písmeno lokální jednotky. Pokud se ale do argumentu písmeno diskové jednotky nezadá, spojení je také vytvořeno a uchováno. Tyto spojení si po určité časové době nebo v případě potřeby, zavírá sám operační systém. Po vytvoření spojení, pak jen stačí použít metodu `Copy` ze třídy `System.IO.File`.

```
File.Copy("\\"+IP+"\d\Opcondata\Logfile.cps","C:\cílová cesta")
```

Na serveru má každá stanice svou složku, do které se konkrétní *„Logfile“* dané stanice ukládá. Při ukládání se soubor uloží do složky stanice a přepíše tak soubor z předchozí relace s konkrétní stanicí. V případě, kdy byl v cílovém adresáři přepisován všemi stanicemi jeden cílový soubor, docházelo při následném zpracování *„Logfile.cps“* a požadavku o stažení dalšího souboru z následující stanice, ke konfliktu přístupu k již používanému souboru, přestože byl přístup k souboru korektně ukončen. Proto jsme zvolili toto řešení separace souboru z jednotlivých stanic do jednotlivých složek, kde již ke konfliktu nedochází.

Celý proces stahování souborů v tomto stavu nyní funguje bez sebemenších problémů.

4.2 Rozbor logovacího souboru

V souboru *„Logfile.cps“* se nachází 2002 řádků s daty. První dva řádky jsou informační. Na prvním je uvedena verze logovacího souboru a na druhém pro nás nepotřebné popisky. Zbylých 2000 řádků jsou pro nás užitečné data.

V případě kdy CPS na stanici zapisuje do souboru novou událost, neboli zapíše nový řádek s chybovým hlášením, zapíše ho na pozici následujícího řádku, než do kterého zapisoval naposledy. Soubor se tedy postupně přepisuje odshora dolů. Jakmile dorazí na řádek 2002, další řádek zapíše na pozici třetího řádku a znovu pokračuje od shora dolů.

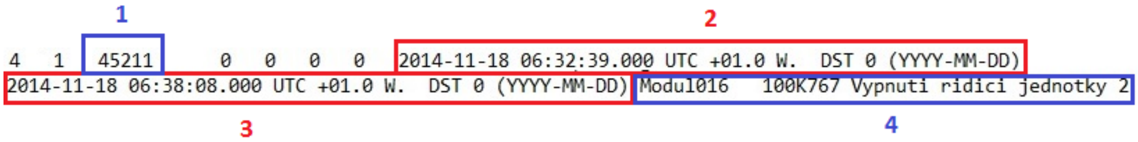
4.2.1 Popis řádku

Ukázka řádku je na obrázku (4.1). Řádek byl pro znázornění na obrázku rozdělen na dva, ve skutečnosti se jedná o data z jednoho řádku a data jsou seřazeny postupně za sebou.

Významy jednotlivých polí:

1 Identifikační číslo: Číslo identifikuje důvod, který zapříčinil zápis události.

2 Čas vzniku: Čas ve který byla událost zapsána do souboru. Jedná se o čas UTC (Coordinated Universal Time), tedy korigovaný světový čas. V řádku je také



Obr. 4.1: Řádek z Logfile

uvedeno *+01.0 W*, což značí ve kterém časovém pásmu se stanice nachází. Po načtení tohoto času budeme tedy přičítat hodinu. Dále se v tomto poli nachází informace DST 0, neboli (Daylight Saving Time) informace o letním čase. Opět tedy budeme muset přičítat jednu hodinu v závislosti na tom, zda je letní nebo zimní čas.

3 Čas vymazání: Systém zápisu času UTC a DST je stejný jako u času vzniku. Tento čas je ale v momentě vzniku nového řádku nevyplněn. Je doplněn až v momentě, kdy obsluha stanice vymaže v CPS hlášení o vzniklých událostech.

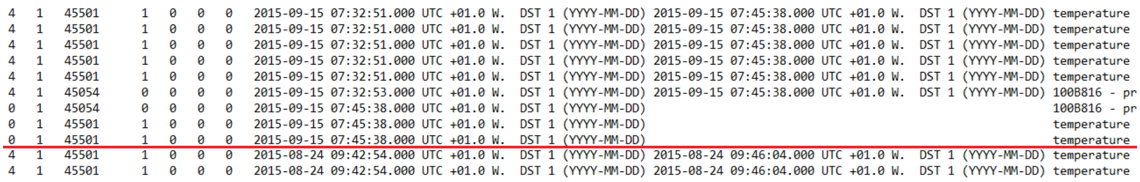
4 Text: Zde je uveden text události. Někdy se v textu vyskytuje znakový pár *%s*, který značí, že na toto místo patří dosadit hodnota uvedená na konci řádku.

Ostatní Zbylé indikátory v řádku jsou pro nás nepotřebné.

4.2.2 Princip zapisování

Události jsou zapisovány do souboru *„Logfile"* postupně tak, jak vznikají. Ať už je stanice v provozu nebo je zastavena, události se do souboru neustále zapisují. Pokud je tedy stanice v provozu a dojde k chybě, tato událost se zapíše a následovně pak může dojít k zastavení stanice, ale také nemusí. Záleží na tom, jestli je událost vyhodnocená jako důvod pro zastavení stroje či nikoli.

Pokud tedy dojde k zastavení stroje do *„Logfile"* jsou i nadále zapisovány následující události jako je například pokles teploty pod minimální hodnotu a mnohé další. Jak již bylo řečeno výše, u nově vzniklých událostí není zapsaný čas vymazání. Nových událostí se běžně nashromáždí hned několik. *„Logfile"* pak může vypadat asi takto (obr. 4.2).



```
4  1  45501  1  0  0  0  2015-09-15 07:32:51.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45501  1  0  0  0  2015-09-15 07:32:51.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45501  1  0  0  0  2015-09-15 07:32:51.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45501  1  0  0  0  2015-09-15 07:32:51.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45501  1  0  0  0  2015-09-15 07:32:51.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45054  0  0  0  0  2015-09-15 07:32:53.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 100B816 - pr
0  1  45054  0  0  0  0  2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD)                                                     100B816 - pr
0  1  45501  1  0  0  0  2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD)                                                     temperature
0  1  45501  1  0  0  0  2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD)                                                     temperature
4  1  45501  1  0  0  0  2015-08-24 09:42:54.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-08-24 09:46:04.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
4  1  45501  1  0  0  0  2015-08-24 09:42:54.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) 2015-08-24 09:46:04.000 UTC +01.0 W.  DST 1 (YYYY-MM-DD) temperature
```

Obr. 4.2: Ukázka Logfile

K dopsání času vymazání dochází u všech vzniklých událostí naráz v momentě,

kdy obsluha stanice stiskne tlačítko na vymazání hlášení o chybách. Stejně tak může obsluha kdykoli odstraňovat hlášení o událostech, které vznikly za provozu a nezapříčinily zastavení stanice.

V momentě, kdy stanice zapisuje čas vymazání u většího počtu vzniklých událostí, je minimální šance, že v průběhu této operace dojde k překlopení aktuálního času o jednu vteřinu a následující část událostí bude mít čas vymazání o sekundu vyšší.

Samotné zapisování událostí do *„Logfile“* operačním systémem CPS, je v některých případech nespolehlivé. Může dojít k situaci, že některé řádky jsou přeskočeny nebo naopak nejsou zapsány vůbec. Někdy je hlášení zapsáno až později, mezi tím, co byly již zapsány nové aktuálnější.

První zapsaná chyba je zpravidla ta, která způsobila vypnutí, ale skutečná příčina se může skrývat mezi těmi následnými chybovými hlášeními.

Také není pravidlem, že první hlášení, které zapříčiní zastavení stanice je reálnou příčinou zastavení, ale skutečná příčina se může skrývat mezi těmi následnými chybovými hlášeními. Proto při identifikaci příčiny prostoje nepůjde automaticky na tyto události spoléhat. Typický příklad je prasklé vysokotlaké potrubí, které způsobí vypnutí na zvýšenou teplotu na poblíž umístěném termočlánku. Zvýšená teplota by se dala v tomto případě mylně interpretovat jako narůstající opotřebení.

4.2.3 Rozpoznání nových událostí

Po samotném stažení souboru *„Logfile“*, bude nutné v datech rozpoznat nově vzniklé události. Proto bude vhodné uchovávat si na serveru pozici posledního zpracovaného řádku v předchozí relaci. Nově vzniklé události pak můžeme rozpoznat podle času vzniku jednotlivých hlášení.

pozice naposledy zpracovaného řádku v předchozí relaci

```
2015-09-15 07:32:51.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:32:51.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:32:51.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:32:51.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:32:51.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:32:53.000 UTC +01.0 W.   2015-09-15 07:45:38.000 UTC +01.0 W.  DST 1
2015-09-15 07:45:38.000 UTC +01.0 W.
2015-09-15 07:45:38.000 UTC +01.0 W.
2015-09-15 07:45:38.000 UTC +01.0 W.
2015-08-24 09:42:54.000 UTC +01.0 W.   2015-08-24 09:46:04.000 UTC +01.0 W.  DST 1
2015-08-24 09:42:54.000 UTC +01.0 W.   2015-08-24 09:46:04.000 UTC +01.0 W.  DST 1
```

Obr. 4.3: Detail časových údajů

Jak je vidět na detailu časových údajů na obrázku (4.3), po otevření souboru si nejprve najedeme na naposledy zpracovaný řádek a načteme z něj čas vzniku. Ten

pak porovnáme s časem vzniku v následujícím řádku. Pokud je v následujícím řádku čas vyšší nebo stejný, znamená to, že v souboru přibyla nová hlášení. Na obrázku jsou tyto nová hlášení za pomyslnou zelenou čárou.

Poté procházíme řádek po řádku a opět porovnáváme, zda následující řádek má větší či stejný čas vzniku než aktuální řádek. Dokud nenarazíme na stav, kdy následující řádek má čas menší než aktuálně zpracovávaný řádek, na obrázku (4.3) jsou to data okolo pomyslné červené čáry. V tom případě obsahují následující řádky stará již jednou zpracovaná data. Na serveru si uložíme pozici aktuálně zpracovávaného řádku jako poslední zpracovanou pozici v tomto *„Logfile“*.

4.2.4 Třídění nových událostí

Nové události je nutné roztřídit. Po konzultaci se zadavatelem jsme jako nejužitečnější možnost třídění zvolili možnost třídění do balíků chybových hlášení podle data vymazání. To znamená, do jednoho balíku chyb zahrnout všechna hlášení, která byla vymazána ve stejný čas s tím, že hlášení o chybě, které v daném balíku vzniklo jako první, bude uváděno jako chyba hlavní.

Při implementaci bude nutné počítat s již výše zmíněnou možností posunutí času v průběhu doplňování času vymazání do jednotlivých řádků. Takže balík chyb bude obsahovat hlášení s časem vymazání v rozmezí dvou sekund.

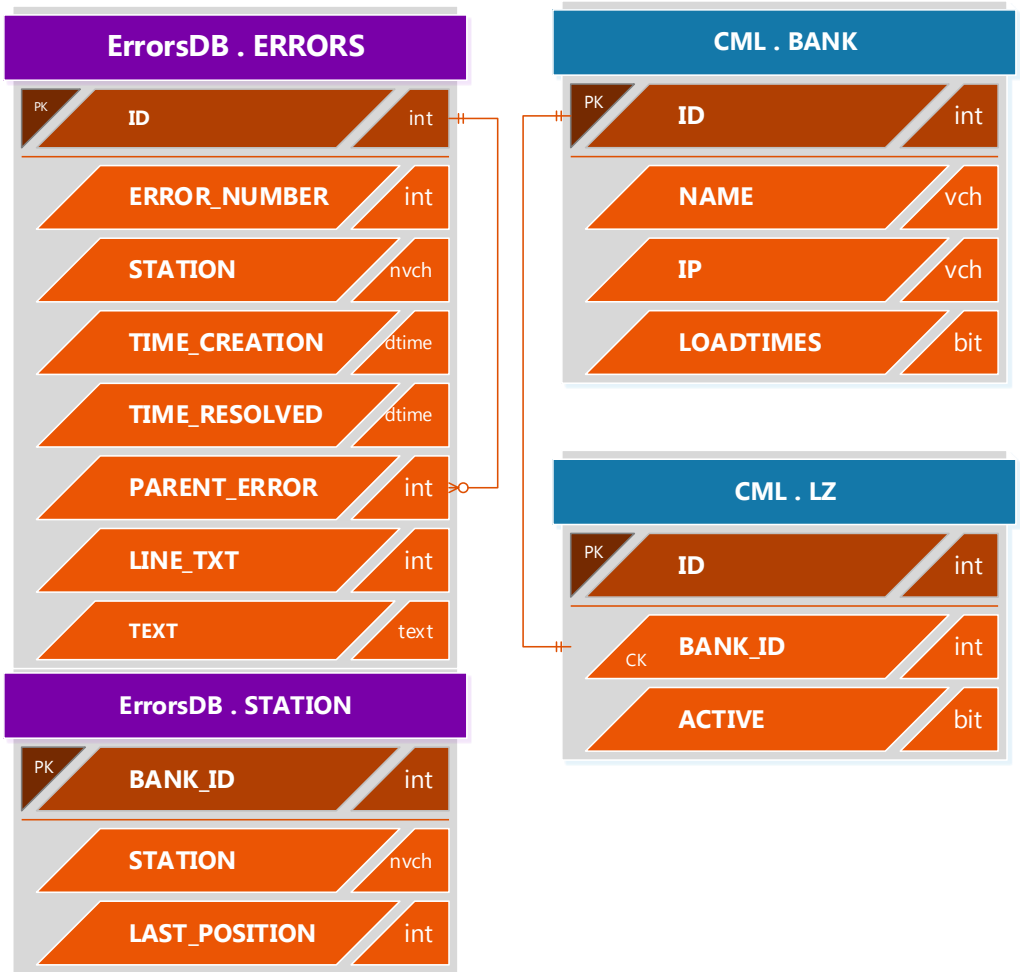
V případě, kdy u nových událostí dosud nebude vyplněn čas vymazání, budeme předpokládat, že budou odstraněny všechny v jeden čas. To znamená, že z těchto událostí bude vytvořen také balík, stejně jako by byl čas vymazání již vyplněn.

Z dat na obrázku (4.3), budou z nových dat mezi pomyslnou zelenou a červenou čárou vytvořeny dva balíky chyb. Jeden balík bude obsahovat šest hlášení s časem vymazání 07:45:38 a druhý se třemi hlášeními s nevyplněným časem.

4.3 Datová struktura pro sběr hlášení ze stanic

Pro ukládání nových událostí, tedy jednotlivých hlášení o chybách do serverové databáze, je nejprve nutné navrhnout strukturu databáze *ErrorsDB*, kterou budeme používat. Návrh struktury naší databáze je znázorněn na obrázku (4.4).

V našem systému budeme také používat některá již existující data z databáze *CML*. Nejprve si ale popíšeme význam jednotlivých polí ve dvou nově vzniklých tabulkách v databázi *ErrorsDB*. Tabulku *ERRORS* pro ukládání událostí z *„Logfile“* a tabulku *STATION* pro ukládání poslední zpracované pozice řádku v *„Logfile“*.



Obr. 4.4: Struktra databáze pro sběr hlášení

Tabulka **ErrorsDB.ERRORS**:

ID: Jednoznačný identifikátor jednotlivých hlášení. Číslo se s nově vznikajícími řádky postupně inkrementuje.

ERROR_NUMBER: Hodnota z pole 1 na obrázku (4.1). Toto hodnota identifikuje důvod, který zapříčinil zápis události. Momentálně tuto hodnotu nebudeme nijak využívat, ale v budoucnu by mohla být k něčemu platná.

STATION: Název stanice ze které hlášení o chybě pochází.

TIME_CREATION: Hodnota z pole 2 na obrázku (4.1), tedy čas vzniku chyby. Čas je korigovaný o UTC pásmo a DST.

TIME_RESOLVED: Hodnota z pole 3 na obrázku (4.1). Toto je nepovinný údaj, protože čas vymazání nemusí být vždy vyplněn. Čas je opět korigovaný o UTC pásmo a DST.

PARENT_ERROR: Toto je pro nás velmi potřebný údaj, pomocí kterého budeme indexovat, do kterého balíku hlášení o chybách, právě toto hlášení patří.

Provedeme to tak, že do tohoto pole uložíme **ID** hlavní chyby z balíku do kterého má toto hlášení patřit.

LINE_TXT: Do tohoto pole se uloží pozice řádku v *„Logfile“* jen v tom případě, když nebude vyplněn čas vymazání. Poslouží nám to později, při pozdějším dohledávání těchto nevyplněných časů vymazání. Toto pole je opět nepovinný údaj.

TEXT: Hodnota z pole 4 na obrázku (4.1). Zde uložíme text chybového hlášení.

Tabulka **ErrorsDB.STATION**:

BANK_ID: Identifikační číslo pod kterým je vedena stanice v systému CML.

STATION: Název stanice.

LAST_POSITION: Pozice posledního zpracovaného řádku v konkrétním *„Logfile“* na stanici.

Z tabulek v databázi CML budeme zjišťovat aktuální informaci, které stanice jsou do monitorování zahrnuty a jejich aktuální IP adresu.

4.4 Implementace sběru hlášení ze stanic

V této části se budeme zabývat samostatnou implementací procesu zpracování *„Logfile“*. Celý proces zpracování poběží ve webovém systému *CPS_Errors* na serveru.

4.4.1 Periodické spouštění

Nejprve je nutné zajistit spolehlivé periodické spouštění, které spustí celý proces zkontrolování, zda se v *„Logfile“* vyskytly nové události. U technologie ASP.NET, je takové periodické spuštění procesu poměrně problematické.

Například jednoduchá možnost vytvoření nekonečné smyčky, ve které by se daný proces volal po minutě stále dokola, je silně nespolehlivá. ASP.NET sám o sobě zavírá vlákna, které běží již dlouhou dobu a v případě velkého zatížení serveru, může server některá vlákna náhodně zavírat. V neposlední řadě je potřeba, aby se proces periodického spouštění po restartu serveru opět sám aktivoval.

Proto jsme se rozhodli, vyřešit tento problém pomocí Task Scheduler. Task Scheduler je komponenta Microsoft Windows, která umožňuje spouštět naplánované úlohy v určitý čas, případně je periodicky opakovat.

Jako naplánovanou úlohu jsme vytvořili spuštění VB skriptu *hide.vbs*. Tento skript se bude volat Task Schedulerem periodicky vždy po minutě.

Akce prováděná naplánovanou úlohou v scheduleru:

```
wscript C:\...\hide.vbs
```

Obsah skriptu *hide.vbs*:

```
command = "powershell.exe -nologo -command C:\...\periodicGet.ps1"
set shell = CreateObject("WScript.Shell")
shell.Run command,0
```

Ve skriptu *hide.vbs* se provede zavolání Windows PowerShell skriptu *periodicGet.ps1*. V případě, kdy by se ze Scheduleru spouštěl rovnou skript *periodicGet.ps1*, by na obrazovce serveru vždy probliklo okno s příkazovým řádkem, přestože v parametrech bylo nastaveno, aby se okno nezobrazovalo. Skript *hide.vbs* tedy způsobí že celá operace proběhne „neviditelně“.

Obsah skriptu *periodicGet.ps1*:

```
$web = New-Object system.Net.WebClient
$web.UseDefaultCredentials=$true
Try{
$web.Downloadstring("http://jh2etc01.cz.bosch.com/.../Errors.aspx")
}
Catch { Write-Warning "$($error[0])" }
```

Ve skriptu *periodicGet.ps1* se odešle HTTP GET na zadanou URL (Uniform Resource Locator) adresu. Při generování HTTP odpovědi ze stránky *Errors.aspx*, se automaticky zavolá funkce `Page_Load` v code–behind, tedy kódu na pozadí této stránky. Z této funkce pak můžeme spustit proces zkontrolování nových událostí.

Celé toto řešení je velice robustní a není náchylné na restartování služby IIS, ani na restartování celého serveru.

4.4.2 Programové parsování řádku

Za zmínku také stojí postup, jakým v programu rozebíráme data na jednotlivých řádcích ve staženém *„Logfile“*. Na parsování dat v řádku použijeme třídu `Regex`, která nám umožňuje pracovat s regulárními výrazy. Regulární výraz je textový řetězec složený z určitých znaků (tzv. Pattern). Tyto znaky definují typ znaku nebo skupiny znaků, které v prohledávaném textovém řetězci hledáme.

V kódu pod tímto textem máme ukázku, kdy pomocí třídy `Regex` prohledáváme hlášení z *„Logfile“*. Do konstruktoru třídy `Regex` předáme onen regulární výraz. Zavoláním metody `Match`, jako parametr vložíme prohledávaný text, která nám vrátí instanci třídy `Match`, ze které si následně vytáhneme, zda se prohledávaný text shodoval s regulárním výrazem, případně i hledaná data.

```
Dim pattern As String = "(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)
     _\s+(\d+)\s+(\d+-\d+-\d+\s+\d+:\d+:\d+.\d+)\s+UTC\s+\+0(\d).0\s+
     _\S+\s+DST\s+(\d)\s+\(YYYY-MM-DD\)\s+(.*)"
Dim rNoDate As Regex = New Regex(pattern, RegexOptions.IgnoreCase)
Dim vysledeky As Match = rNoDate.Match(textRadku)
```

V systému samozřejmě používáme mnoho patternů. Tento pattern výše je v systému použit pro rozpoznání řádků, které nemají vyplněný čas vymazání. Ve výsledku se nám tedy krásně povedlo v několika řádcích kódu rozkódovat informace v *„Logfile“*. Díky tomu jsme si ušetřili spoustu práce s postupným rozebíráním řádku po znacích.

4.4.3 Stručný popis základního funkčního principu implementovaného systému sběru hlášení ze stanic

Pomocí Task Scheduleru a skriptů se nám po uplynutí nastavitelného intervalu, který je předběžně nastaven na jednu minutu, vyvolá aktivita webové stránky *Errors.aspx*, tedy i provedení funkce `Page_Load` v code–behind. V této funkci zavoláme proceduru na spuštění prohlédnutí všech stanic a vyhledání nových hlášení. Samotné prohledávání pak běží v samostatném vlákně na pozadí.

Zjednodušeně řečeno se tato procedura nejprve podívá, zda předchozí prohledávání již skončilo a vlákno s ním bylo zavřeno nebo zda prohledávání stále probíhá. Jestliže předchozí prohledávání probíhá, nové prohledávání nebude aktivováno. Pokud ale předchozí prohledávání již skončilo, procedura aktivuje nové samostatné vlákno, které začne provádět nové prohledávání.

V tomto nově vzniklém vlákně si nejdříve z databáze CML vytáhneme jména a IP adresy stanic, které budeme prohledávat. Připojíme se na první stanici v pořadí, stáhneme aktuální *„Logfile“* na server, na serveru z něj vyparsujeme nové hlášení o chybách, data uložíme do databáze `ErrorsDB` společně s poslední zpracovanou pozicí v *„Logfile“* a přejdeme na další stanici v pořadí.

4.4.4 Popis doplňujících postupů a procedur implementovaných v systému sběru hlášení ze stanic

V systému sběru hlášení jsou připravené některé programy a postupy, které zlepšují jeho využití.

Volitelný čas aktivace: V systému je implementován princip, který nám umožňuje spustit proceduru prohledávání, až při x–tém načtení webové stránky *Errors.aspx*. Ve výsledku nám to umožňuje si nezávisle na Windows Task

Scheduleru měnit po kolika minutách se má znovu provádět procedura prohledávání.

Doplňování času vymazání: Při x–tém načtení stránky nebo při zavolání je také možné spustit proceduru, která se podívá do tabulky ErrorsDB.ERRORS, z této tabulky si zjistí, u kterých hlášení nebyl vyplněn čas vymazání a tento čas se pokusí znovu dohledat v aktuálních *„Logfilech“* na stanicích. Je zde možnost volby maximálního možného stáří hlášení o chybách, které se budou takto doplňovat.

Hlášení o Interních chybách: V systému je také zabudované zapisování hlášení, o vzniklých interních chybách v průběhu prohledávání a zpracování dat. Do textového souboru se tedy zapisují výjimky vzniklé v průběhu programu, případně i další informační hlášení tohoto systému. Vše se zapisuje s časovým razítkem.

Prohledání vybraných stanic: Na zavolání je možné prohlédnout a aktualizovat data jen z vybrané stanice.

5 NÁVRH METODY VÝPOČTU PROSTOJŮ A JEJÍ IMPLEMENTACE

Nyní se budeme zabývat výpočtem prostojů. Prostoj je tedy čas, po který zkušební stanice není v provozu. Uživateli nestačí vědět, že se stanice vypnula v nějakém čase, ale chce vědět, jak dlouho zkušební stanice stála. V případě pokud zkušební stanice stále stojí, bude se doba prostoje s časem měnit. Proto bude nutné jí pravidelně aktualizovat.

Zároveň bude nutné ke každému nově vzniklému záznamu o prostoji automaticky přiřadit inženýra, který je aktuálně za danou stanici zodpovědný.

Prostoje nemůžeme počítat automaticky na základě nasbíraných chybových hlášení z CPS na stanicích, protože zapisování nových hlášení, které do *„Logfile“* zapisuje CPS, je poměrně nespolehlivé. Proto budeme muset zvolit jiný postup počítání prostojů.

5.1 Princip výpočtu prostojů

Pro výpočet prostojů použijeme data z tabulky `EVENTS` v databázi `CML`. Tato tabulka byla vytvořena pro naše účely a data v ní aktualizuje aplikace, která načítá aktuální stav hodin na stanici. Ukázka dat z této tabulky je na obrázku (5.1). Události do této tabulky jsou zapisovány řádek po řádku tak, jak se stanice postupně zapínají (AKCE : START) a vypínají (AKCE : STOPP).

ID	BANK_ID		AKCE	DATETIME_PREV	1.případ	2.případ
29264	2		START	2015-12-02 17:06:23.000		
29265	80	1	STOPP	2015-12-02 17:00:13.000		
29266	80		START	2015-12-02 17:17:00.000	p_1	
29267	102	1	STOPP	2015-12-02 17:23:55.000		
29268	80		STOPP	2015-12-02 17:47:13.000		p
29269	31	2	START	2015-12-02 18:00:11.000		
29270	72		STOPP	2015-12-02 18:18:49.000	p_2	
29271	80		START	2015-12-02 18:00:34.000		
29272	80	2	STOPP	2015-12-02 19:07:20.000		

Obr. 5.1: Data tabulky EVENTS

Z těchto dat již lze sledovat jednotlivé stavy stanic a dopočítávat jednotlivé prostoje. Požadavkem zadavatele také bylo nereagovat na spuštění a chod stanice, které je kratší než definovaný časový limit. To znamená, že i toto krátké spuštění bude zahrnuto do prostoje. Většinou se totiž jedná o zkušební spuštění, které má za účel identifikovat místo závady, například při prasklém vysokotlakém potrubí.

Pro názornost je na obrázku (5.1) zobrazen příklad zpracování prostojů, které vznikly na stanici s `BANK_ID = 80`. Ve dvou případech bude ukázán princip zakládání prostojů.

1.případ Zde je výše zmíněný volitelný minimální čas běhu stanice pro následovné založení prostoje nastaven na **30 minut**. Na počátku byla stanice v provozu, než přišel STOPP č.1. Po tomto stopu stanice stála 17minut a byla znovu spuštěna START č.1. Běžela po dobu 30 minut a 13 sekund, než přišel STOPP č.2. Tento čas je větší než nastavených 30 minut, takže prostoj *p_1* bude vypočten jako časový rozdíl STOPP č.1 a START č.1. Zároveň se po STOPP č.2, začne počítat nový prostoj *p_2*.

2.případ V tomto případě si řekněme, že čas bude nastaven na **31 min**. Takže čas 30 minut a 13 sekund mezi START č.1 a STOPP č.2 je menší než nastavená minimální doba pro založení nového prostoje 31 minut. To znamená, že na tuto událost spuštění stanice a opětovnému zastavení, nebude reagováno a čas prostoje *p* bude vypočten z rozdílu STOPP č.1 a START č.2.

5.2 Datová struktura pro výpočet prostojů

Na obrázku (5.2) je znázorněná datová struktura databáze, do které budeme ukládat nově vzniklé prostoje.

Tabulka **ErrorsDB.CENTRAL**:

ID: Jednoznačný identifikátor jednotlivých prostojů. Číslo se s nově vznikajícími prostoji postupně inkrementuje.

BANK_ID: Identifikační číslo, pod kterým je stanice vedena v systému CML.

STATION: Název stanice, ze které hlášení o chybě pochází.

GRP_ID: Číslo balíku hlášení o chybách, které byly nasbírány z *„Logfile“* na stanicích. Toto číslo ukazuje na balík chyb v `ErrorsDB.ERRORS`. Detaily budou probrány později.

DATABASE_ID: Tato kolonka je připravená pro případ nově pořízených CPS, které by měly databázi s chybovými hlášeními zabudovanou již v sobě.

REAL_REASON_ID: Tato kolonka bude sloužit pro zadání reálné příčiny výpadku z předdefinovaných v tabulce `ErrorsDB.ERROR_NAME`.

Obr. 5.2: Struktura databáze pro výpočet prostojů

PROSTOJ: Zde bude uváděn aktuální nebo již výsledný čas prostoje.

ERROR_START: Časové razítko, kdy prostoj začal.

ERROR_STOP: Časové razítko, kdy prostoj skončil. Bude vyplněno jen v případě, že byl prostoj již ukončen.

IN_STOP: Indikace, zda prostoj je stále aktivní, tedy zda stanice stojí nebo ještě nebyla spuštěna na dobu delší než definovaná doba pro minimální čas provozu.

ETC_ID: Identifikátor zkušebního inženýra který je aktuálně za danou stanici zodpovědný.

STATISTIC: Bit indikuje zda bude tento prostoj zahrnut do statistik.

LOCK: Bit indikující uzamčení záznamu.

NOTE: Poznámka.

APPROVED: Identifikátor zda již byl prostoj editován.

APPROVED_TIME: Časové razítko poslední editace prostoje.

A_USER_ID: Identifikace posledního editora prostoje.

+8 kategorií: Osm kategorií do kterých budou zkušební inženýři při editaci roz-

dělovat celkový prostoj. Tyto kategorie budou blíže specifikovány dále v textu.

Tabulka **ErrorsDB.ERROR_NAME**:

ID: Jednoznačný identifikátor předdefinovaných příčin výpadků. Číslo se s nově vznikajícími prostoji postupně inkrementuje.

ERROR_TEXT: Text příčiny.

TYPE: Oblast příčin odkazuje na předvolené oblasti reálných příčin výpadku v tabulce `ErrorsDB.ERROR_TYPE`. (Elektrická, Hydraulická, Mechanická, Ostatní, SW, ...).

ACTIVE: Ukazatel zda je tato příčina stále nabízena k přiřazení k nově vzniklým prostojům.

5.3 Implementace výpočtu prostojů

V této části práce se budeme zabývat implementací výpočtu prostojů, která poběží paralelně ke sběru hlášení z *„Logfile“* na stanicích. Celý proces výpočtu prostojů samozřejmě poběží ve webovém systému *CPS_Errors* na serveru.

5.3.1 Periodické spouštění

Celý proces indikování nových událostí v tabulce `CML.EVENTS` musí být vždy nějak inicializován. S výhodou tedy využijeme již funkčního principu aktivace činnosti procesu pomocí Windows Task Scheduleru. Každou minutu Task Scheduler pomocí skriptu `hide.vbs` a `periodicGet.ps1` vyvolá aktivitu na stránce *Errors.aspx*, tím se automaticky zavolá funkce `Page_Load` v code–behind a z této funkce budeme volat zahájení metody obsluhující prostoje.

5.3.2 Stručný popis implementované procedury pro výpočet prostojů

Stejně jako u sběru hlášení z *„Logfile“* pracuje tento proces výpočtu prostojů v samostatném vlákně. Pokud se tedy tyto dvě vlákna aktivují naráz, poběží současně nezávisle na sobě.

ID poslední zpracované události se opět uchovává v paměti serveru. Na začátku se pro jistotu procedura podívá, zda již byla předchozí procedura dokončena a samostatné vlákno uzavřeno. Jestliže ano, aktivuje procedura nové vlákno, ve kterém se nejdříve podíváme do tabulky `CML.EVENTS`, zda se v tabulce nacházejí nové události s akcí. Pokud ano, zpracujeme je a podle postupu v kapitole 5.1 v případě potřeby založíme nové prostoje do `ErrorsDB.CENTRAL`.

Při zakládání prostojů se do kolonky `IN_STOP` vyplní, že prostoj je aktivní. To znamená, že doposud nebyl ukončen validní STOPP akcí. Zároveň se také prostoj rovnou pokusí provázat s balíkem chybových hlášení z *„Logfile“*, který pravděpodobně odpovídá příčině vzniku podle časů vzniku hlášení a prostoje. Nejdříve nahlédne do tabulky `ErrorsDB.ERRORS`, zda nenajde shodu. Pokud ne, aktualizuje hlášení z příslušného `Logfile` a poté do tabulky znovu nahlédne a najde shodu. Dále pak v databázi `CML` dohledáme identifikátor inženýra, který je aktuálně ke stanici přiřazen a ten pak uložíme do kolonky `ETC_ID`.

Poté, co vlákno dokončí zpracování nových událostí, přesune se na aktualizaci času aktivních prostojů v `ErrorsDB.CENTRAL`. To znamená, že dopočítá aktuální délku prostoje do doby, kdy probíhá tento proces. Při tomto dopočítávání se zároveň provádí ukončování prostoje příchodem validní STOPP akce a dopočítání konečného času prostoje.

5.3.3 Popis doplňujících postupů a procedur implementovaných v systému pro výpočet prostojů

Nyní si podrobněji popíšeme další funkce v tomto systému.

Volitelný čas aktivace: V systému je opět implementován princip, který nám umožňuje spustit proceduru výpočtu prostojů až při x–tém načtení webové stránky *Errors.aspx*.

Doplnění GRP_ID: Protože zapisování hlášení do *„Logfile“* od CPS pracuje někdy nespolehlivě, nemusela být vytvořena žádná hlášení při zastavení stanice v momentě, kdy již byl založen prostoj. Toto hlášení bývá dopsáno do *„Logfile“* později. Proto při x–tém načtení stránky nebo při zavolání, je také možné spustit proceduru, která pokusí nevyplněné GRP_ID znovu dohledat a vyplnit. Toto dohledání probíhá pouze u maximálně měsíc starých prostojů.

Hlášení o Interních chybách: Vypisování interních chyb a událostí zde funguje stejně jako v systému pro sběr hlášení.

5.4 Ostatní stanice

Naše diplomová práce i samotné zadání práce je založené na jednotném typu stanic, které generují *„Logfile.cps“* s hlášením o chybách. Oddělení ETC ale také vlastní některé další stanice, které jsou na jiné druhy zkoušek. Tyto stanice se pokusíme do našeho systému částečně také zahrnout.

U těchto stanic se žádné chybové hlášky sbírat nedají. Proto se pokusíme u těchto stanic pouze počítat denní prostoje. V databázi CML je tabulka `CML.LZ`, do které se

periodicky okolo šesté hodiny ráno ukládá záznam s reálnými naběhanými hodinami stanice pro každý den.

Z těchto záznamů nepoznáme, od kdy do kdy stanice stála, ale pouze kolik hodin v daném dnu fungovala a kolik hodin prostála. Rozhodli jsme, že tyto denní záznamy o prostojích z těchto stanic nebudeme míchat do stejné tabulky `ErrorsDB.CENTRAL`. Důvodem je, že data u těchto záznamů budou mírně odlišná a budou vytvářeny z jiné části systému, vytvoříme tabulku novou téměř identickou s výše zmíněnou tabulkou `ErrorsDB.CENTRAL` (5.2) s názvem **ErrorsDB.CENTRAL_DAILY**. Rozdílem je, že záznamy v této tabulce nebudou ukazovat na balík s hlášením o chybách a nebudou k záznamům přiřazováni zkušební inženýři.

Vytváření těchto denních záznamů bude popsáno v kapitole (6.3).

6 GRAFICKÉ ROZHRANÍ PRO EDITACI PROSTOJŮ A SPRÁVU SYSTÉMU

V této části práce nejdříve popíšeme jednoduchou vizualizaci a obsluhu systému pro sběr chybových hlášení a vytváření záznamů o prostojích. Dále pak popíšeme námi vytvořené grafické rozhraní, které mají zkušební inženýři z oddělení ETC k dispozici na editaci a správu prostojů. Zde se také zmíníme o významech jednotlivých kategorií, do kterých budou zkušební inženýři rozdělovat čas prostoje.

Při vytváření těchto grafických rozhraní byl kladen hlavní důraz na jednoduchost, přehlednost a jednotnost s webovým formulářovým systémem CML, který již na oddělení ETC používají.

6.1 Správa systému pro sběr hlášení

Celý proces sbírání chybových hlášek z CPS a vytváření nových záznamů o prostojích musí jít jednoduše deaktivovat. K tomu slouží právě tato kontrolní stránka, jejíž důležitá část je na obrázku (6.1).

Obr. 6.1: Správa chybových hlášek z CPS a vytváření nových záznamů o prostojích.

Jak bylo probráno v kapitolách výše, proces sběru chybových hlášení z CPS a proces vytváření záznamů o prostojích v případě spuštění běží každý v samostatném vlákně nezávisle na sobě. K indikaci, zda jsou jednotlivé procesy aktuálně v provozu, slouží druhý řádek odspoda (Waiting – čeká na aktivaci od scheduleru \ Running – aktuálně pracuje). Čas pod těmito indikátory značí čas posledního dokončeného vlákna s EVENTY, tedy procesu se zjišťováním nových prostojů a dopočítáváním času prostojů. Počítadla cyklů slouží k indikaci, kdy dojde k volitelně přednastaveným doplňkovým procedurám.

Hlavní vypínací tlačítko „Zastavit“ znemožní aktivaci jednotlivých vláken od Task Scheduleru, ale nechá již rozběhnutá vlákna doběhnout. Po stisknutí tlačítka „Zastavit“ se hlavní indikátor RUNNING změní na STOP a tlačítko na „Spustit“

Celý systém je vytvořen tak, aby při delším vypnutí nepřišel o žádná důležitá data. Při opětovném zapnutí systému a při první aktivaci od Task Scheduleru systém dohledá, dopočítá a aktualizuje všechny prostoje, které se odehrály v době vypnutí systému.

6.2 Grafické rozhraní pro zkušební inženýry

Na této stránce budou zkušební inženýři přiřazovat reálnou příčinu výpadku a rozdělovat prostátý čas stanice do jednotlivých kategorii.

6.2.1 Jednotlivé kategorie prostojů

Jednotlivé prostoje budou zkušební inženýři rozdělovat do následujících osmi kategorií z důvodu jiných účtovacích sazeb pro zákazníka. O tyto kategorie je nutné rozšířit SQL tabulku `ErrorsDB.CENTRAL` a `ErrorsDB.CENTRAL_DAILY`. Datovým typem těchto kategorii bude float.

Systém: Do této kategorie patří čas prostoje způsobený běžnou prací na projektu jako například montáž prvku zkoušeného systému, montáž čidel, měření na systému nebo ladění software.

Personál1: Do této kategorie patří čas prostoje, kdy nebyl čas a personál na provedení nutného úkonu v pracovní době.

Personál2: Tato kategorie je totožná s předcházející s tím rozdílem, že do této kategorie patří čas mimo pracovní dobu.

Bank: Do této kategorie patří čas, kdy stanice stála kvůli poruše na stanici.

Infrastruktura: Sem spadá čas prostoje z důvodu výpadku energií, chladící vody nebo ventilace.

Auftragmangel: Do této kategorie spadá čas, kdy pro zkušební stanici nebylo využití, tedy bylo málo zkoušek vhodných na danou stanici.

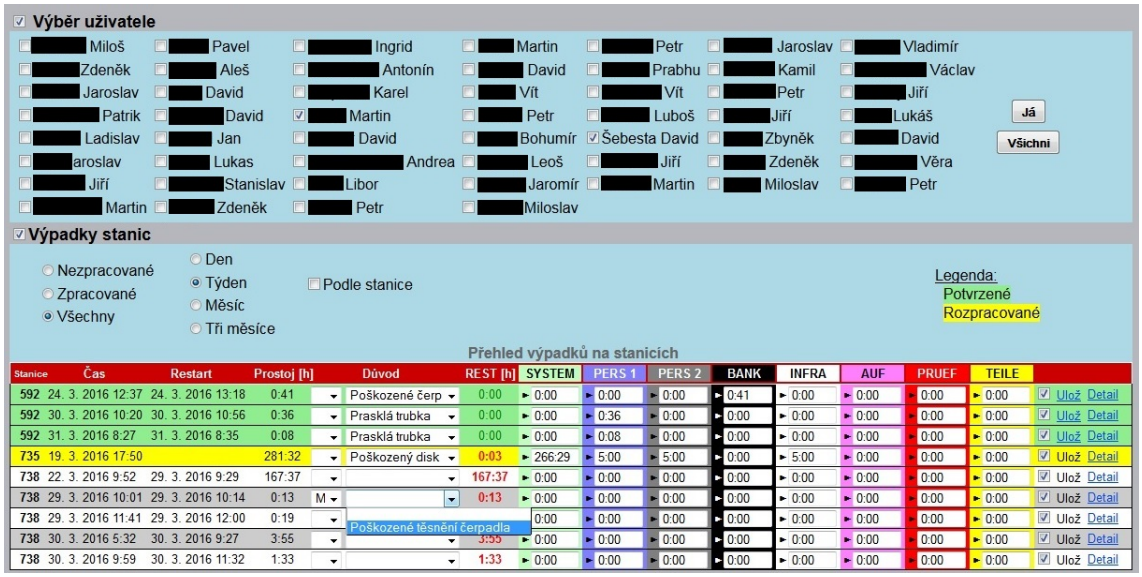
Prüfzeit: Zkoušené díly jsou odmontovány a jsou na mezizkoušce na jiném zařízení.

Teilemangel: Do této kategorie patří prostoj zapříčiněný zákazníkem, když pozdě dodá díl ke zkoušce.

6.2.2 Popis grafického rozhraní

Hlavním cílem bylo, aby proces editace prostojů zabral zkušebnímu inženýrovi co nejméně času. To znamená, aby provedl co nejméně kliknutí myší. Zároveň by měl

mít k dispozici další dostupné informace, když je bude potřebovat. Proto jsme navrhli takové rozhraní, aby vyváženě odpovídalo oběma těmto kriteriím. Výsledkem je webová stránka `\U_CHECK.aspx`, jejíž hlavní část je vidět na obrázku (6.2).



Obr. 6.2: Grafické rozhraní editace prostojů pro zkušební inženýry.

Výběrový filtr

Při načtení stránky se jako výchozí nastavení v horním panelu pro výběr uživatele, automaticky zaškrtne uživatel aktuálně přihlášený do operačního systému na daném počítači v přístupové síti, odkud právě přistupuje na tuto webovou stránku. A následovně se mu zobrazí všechny nezpracované záznamy o prostojích na jeho stanicích za poslední týden.

Uživatel si pak může jednoduše v horním panelu zaškrtnout i záznamy o prostojích patřící jiným uživatelům. Zároveň může v prostředním panelu filtrovat jaké prostoje ho zajímají na základě času nebo fáze zpracování záznamu o prostoji. Ke změně dat dochází ihned po kliknutí na dané kritérium.

V prostředním poli se také nachází zaškrtávací CheckBox, pomocí kterého zrušíme výběr záznamů na základě identifikace zkušebních inženýrů a aktivuje se výběr pomocí jména zkušebních stanic. Uživatel pak už jen vyplní jména stanic oddělené čárkou, které ho zajímají, do TextBoxu, který se objeví hned pod výše zmíněným CheckBoxem.

Data v tabulce

Nyní už k samostatným datům. Zleva je nejdříve sloupec s názvem stanice a poté s časovým razítkem začátku prostoje a koncem prostoje. Dále pak v přiblíženém výřezu z obrázku (6.2) na obrázku (6.3) zleva je celkový prostoj.

Obr. 6.3: Přiblížený výřez z obrázku (6.2).

Následuje část pro přiřazení reálné příčiny výpadku, kterou je nutné vyplnit, aby se dal daný řádek uložit. Předdefinovaných reálných příčin v SQL tabulce `ErrorsDB.ERROR_NAME` je velké množství. Uživatel by tedy musel vybírat z dlouhého seznamu příčin, než by našel tu kterou hledá. Proto první v řadě je úzký DropDownList, který blíže specifikuje okruh chyby kterou hledá. V úzkém DropDownListu je tedy na výběr z předvolených okruhů z tabulky `ErrorsDB.ERROR_NAME` (Elektrická, Hydraulická, Mechanická, Ostatní, SW, . . .). Po výběru okruhu je vidět jen začáteční písmeno zvoleného okruhu, jako je vidět v posledním řádku s daty na obrázku (6.3). Tato předvolba nám následovný výběr z druhého dlouhého DropDownListu s reálnou příčinou zúží jen na příčiny z příslušného okruhu.

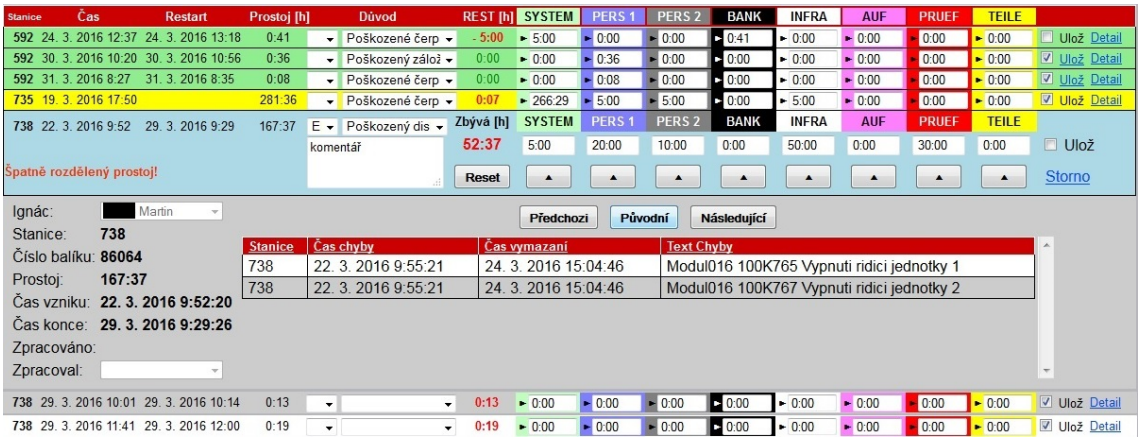
Následujících devět sloupců slouží k rozdělování času prostoje do výše zmíněných kategorií. Aby šel řádek s prostojem uložit, je nutné rozdělit celý čas prostoje mezi jednotlivé kategorie. Prostoj lze rozdělit do osmi TextBoxů v jednotlivých sloupcích. Barvy těchto sloupců bylo nutné dodržet podle všeobecného standardu dlouhodobých zkušeben zadavatelské společnosti. Po kliknutí na TextBox je rovnou celý text označen, aby šel snadno ihned přepsat. Hodnotu do TextBoxu lze zadat buď v časovém formátu *h:mm*, nebo v decimálním hodinovém formátu *1,25*, který se po opuštění TextBoxu ihned opět převede na časový formát *1:15*. Zároveň při opuštění TextBoxu se provede aktualizace zbylého prostoje v prvním sloupci REST. Takto lze prostoj rozdělovat do jednotlivých sloupečků, dokud nebude celý prostoj rozdělen, tedy ve sloupci REST bude hodnota *0:00*. Tohoto stavu lze také jednoduše dosáhnout přičtením celého zbytku prostoje (hodnota ve sloupci REST) do konkrétní kategorie. To se provádí trojúhelníkovým Buttonem před každým TextBoxem. Pak již lze, pokud je přidělená pravá příčina prostoje, daný řádek uložit.

Na konci řádku před tlačítkem „Ulož“ je CheckBox, po jehož odškrtnutí nebude tento prostoj započítáván do statistik.

Na obrázku (6.2) je vidět, že datové řádky se záznamy o prostojích jsou podbarvené zeleně, žlutě, bíle a šedě. Toto podbarvení opět slouží rychlejší orientaci zkušebních inženýrů ve vypsaných datech. Pokud je záznam doposud nikým needitován, střídá se pravidelně bílá barva se šedou pro lepší vizuální orientaci ve větším množství řádků. Pokud prostoj již někdo editoval, bude podbarvení řádku zelené. Pokud prostoj již někdo editoval, ale v době editace prostoj nebyl dokončený, to znamená, že stanice v době editace pořád ještě nebyla spuštěna, bude podbarvení žluté.

Detail

Pokud potřebuje uživatel více informací k danému prostoji, může na konci řádku kliknout na tlačítko „Detail“. To konkrétní řádek rozvine, jak je vidět na obrázku (6.4) a vypíše více detailů.



Obr. 6.4: Detail záznamu o prostoji.

Vrchní panel v otevřeném detailu obsahuje všechna data a funkce jako původní řádek. Navíc panel obsahuje TextBox pro případný komentář a tlačítko „Reset“ pro vynulování rozdělených prostojů v kategoriích.

Ve spodním panelu se pak vypíšou přehledněji informace o daném prostoji včetně jména zodpovědného inženýra za danou stanici v době výpadku a údaje o předchozím editorovi. Dále pak se ve spodním panelu nachází další tabulka, která obsahuje balík chybových hlášení z konkrétní stanice, který byl k danému prostoji přiřazen. Zkušební inženýr pak může snadno tlačítky „Předchozí“ a „Následující“ přepínat mezi předchozími a následujícími balíky chybových hlášení nasbíraných na dané stanici. Dále mezi těmito tlačítky je tlačítko „Původní“, které způsobí návrat na přiřazený balík hlášení k danému prostoji.

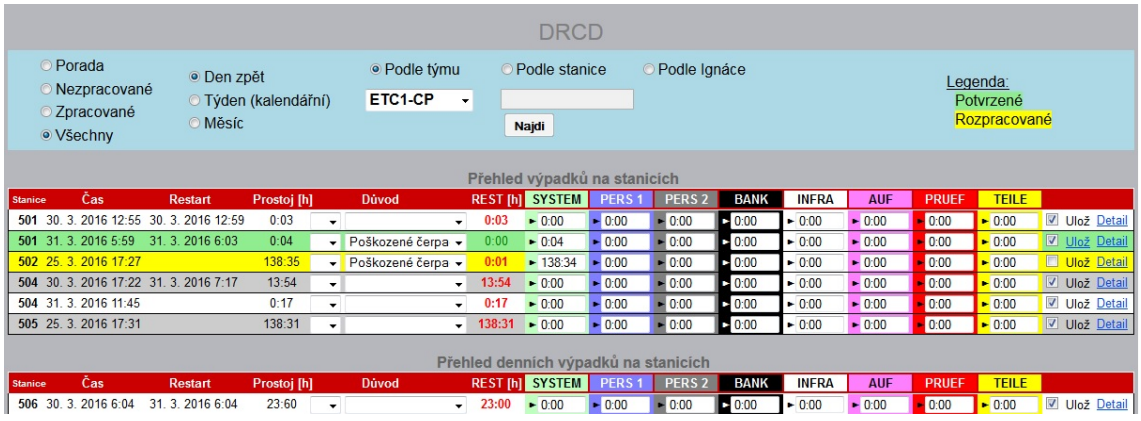
Uložení

Pro uložení zkušební inženýr vždy musí vyplnit skutečnou příčinu výpadku a rozdělit celý prostoj do kategorií. Při editaci prostojů v jednotlivých řádcích (bez otevřeného panelu s detailem) se provede automatické uložení záznamu v momentě, kdy jsou splněny podmínky pro možné uložení řádku. Přesněji se tak stane v situaci, kdy inženýr již zadal reálnou příčinu a klikne na jeden z doplňovacích trojúhelníčků zbytku nerozděleného prostoje do jednotlivých kategorií. Případně se tak stane v momentě, kdy inženýr již rozdělil všechen prostoj a změní nebo vybere reálnou příčinu. Takto jsme opět ušetřili čas, kdy by inženýr následovně musel klikat na tlačítko „Uložit“.

Při každém ukládání se zároveň uloží do příslušných kolonek u záznamu v databázi identifikace editora a čas, kdy došlo k editaci. Pro zachování historie o vícenásobné editaci se tato informace zároveň vždy uloží i do kolonky `NOTE` společně s případným komentářem editora.

6.3 Grafické rozhraní pro vedoucí týmů a porady

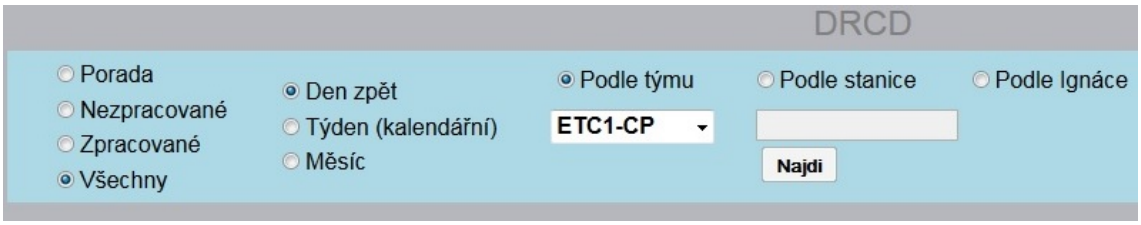
Důležité také bylo navrhnout grafické rozhraní na ranní společné porady a pro vedoucí týmů. Základ zůstane stejný jako na stránce pro zkušební inženýry. Rozdíl však bude ve výběrovém filtru a v přidané další tabulce s denními hlášeními o prostojích z SQL tabulky `ErrorsDB.CENTRAL_DAILY`. Hlavní část výsledného provedení z webové stránky `\DRCD.aspx` můžeme vidět na obrázku (6.5).



Obr. 6.5: Grafické rozhraní pro vedoucí týmů a porady.

Výběrový filtr

V detailu výběrového filtru na obrázku (6.6) je vidět, že do výběru podle stavu rozpracovanosti záznamu přibyl mód „Porada“ a do výběru podle konkrétních stanic přibyl výběr „Podle týmu“.



Obr. 6.6: Detail výběrového filtru.

Opět byl kladen velký důraz na rychlý přístup k datům. Jelikož se pracovní porady konají každý pracovní den ráno, tak při prvním načtení této stránky se předvolí mód „Porada“a výběr „Podle týmu“. Pak jen stačí v DropDownListu zvolit příslušný tým a data jsou ihned vypsána.

Mód Porada je nezávislý na volbě časového rozmezí. Při módu Porada jsou vypsány všechny doposud needitované prostoje (bílé a šedé podbarvení řádku). Dále pak všechny editované prostoje, které v době poslední editace stále nebyly dokončené (žluté podbarvení řádku). V neposlední řadě také ty, co byly editovány za poslední den (zelené podbarvení řádku), pokud je ale pondělí, budou to editované prostoje za poslední tři dny. To umožní mít vedoucímu týmu větší přehled nad správným rozdělováním prostoje a přiřazováním reálných příčin výpadku.

Dále pak má uživatel opět možnost zvolit si zobrazení prostojů jen z konkrétních stanic, nebo prostoje přiřazené konkrétním zkušebním inženýrům. Při volbě výběru podle zkušebních inženýrů se zobrazí nový panel se zaškrtávacím seznamem všech zkušebních inženýrů tak jako na obrázku (6.2).

Denní prostoje z ostatních stanic

V kapitole (5.4) jsme se zmiňovali o tom, že do našeho systému částečně zahrneme i některé ostatní stanice oddělení ETC. Pro každou takovou stanici tedy vždy budou vytvořeny denní záznamy z předchozích dnů, kolik za ten den stanice prostála času. Tyto záznamy o prostojích jsou pak vypsány v samostatné tabulce, jak je vidět ve spodní části obrázku (6.5). V databázi jsou tyto denní záznamy o prostojích uloženy v sql tabulce `ErrorsDB.CENTRAL_DAILY`.

O vytváření těchto záznamů se bude starat právě tato webová stránka `/DRCD.aspx`. Zjednodušeně řečeno se pokaždé při prvním načtení této stránky systém podívá, zda jsou všechny záznamy z předchozích dnů již v v sql tabulce `ErrorsDB.CENTRAL_DAILY` vytvořeny. Pokud nejsou, tak chybějící denní záznamy automaticky vytvoří. Uživatel tedy vždy bude mít aktuální data.

7 SMS UPOZORŇOVÁNÍ NA VÝPADEK ZKUŠEBNÍ STANICE

Jedním z bodů zadání diplomové práce také bylo, informovat příslušného zkušebního inženýra pomocí GSM modemu o výpadku zkušební stanice, za kterou aktuálně zodpovídá. Tato informace přijde formou SMS zprávy na jeho osobní nebo služební mobilní telefon. Včasné upozorňování zkušebních inženýrů na výpadek jejich stanice může značně zkrátit doby prostojů zkušebních stanic a tím zvýšit efektivnost zkušebního procesu.

Zároveň ale musí být toto upozorňování přizpůsobitelné na míru jednotlivým zkušebním inženýrům. Každý si tedy bude moci zvolit, kdy chce být na výpadky upozorňován.

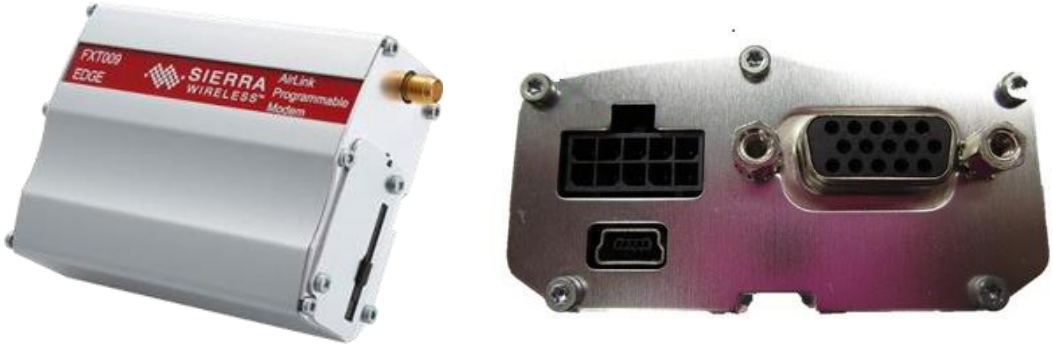
7.1 GSM modem

Požadavkem zadavatele bylo, aby měl celý proces odesílání SMS zpráv pod svou správou. Možnost odesílání SMS zpráv přes některou z internetových důvěryhodných placených SMS brán by nesplnilo toto kritérium. Zároveň stejně nevhodné by bylo využití SMS brány jiného oddělení ve společnosti Bosch Diesel s.r.o. v Jihlavě. Doposud ani žádné jiné oddělení nemá jakýkoli podobný SMS systém v provozu. Oddělení ETC tedy bude v tomto ohledu průkopníkem u zadavatelské společnosti v Jihlavě.

Proto jsme po konzultaci s příslušným IT oddělením společnosti Bosch zvolili možnost, kdy připojíme IT oddělením doporučený GSM modem Sierra Wireless Airlink Fastrack Xtend FXT009 (obrázek 7.1) přímo k serveru oddělení ETC. Takto budeme moci obsluhovat GSM modem přímo z naší ASP.NET webové aplikace.

7.1.1 Vlastnosti Sierra Wireless Airlink Fastrack FXT009

Jedná se o průmyslové řešení GSM modemu v hliníkovém obalu. Modem je velmi robustní a bezúdržbový. Jde o nástupce velmi populárních GSM modemů od společnosti Wavecom, jejímž vlastníkem je nyní právě Sierra Wireless. FXT009 je schopný odeslat až 500 sms zpráv o délce 160 znaků za 47 minut [12]. Předpoklad našeho využití je v řádech desítek SMS denně. Pro naše účely je tedy tento modem plně dostačující. Technické parametry jsou uvedeny v tabulce (7.1).



Obr. 7.1: Sierra Wireless Airlink Fastrack Xtend FXT009 [11].

Tab. 7.1: Technické parametry.

Síťové technologie:	GSM / GPRS / EDGE
Frekvenční pásma GSM:	850 / 900 / 1800 / 1900 MHz
Komunikační rozhraní:	RS232, USB 2.0
Rozměry:	89 x 60 x 30 mm
Hmotnost:	100 g
Napájecí napětí:	4,75 až 32 V
Provozní teplota:	-30 až +75°C

7.1.2 Instalace GSM modemu

Jak je vidět na obrázku (7.1), zařízení má na čelní straně slot na SIM kartu a konektor antény. Na zadní straně pak napájecí konektor a rozhraní Mini-B USB a RS232. Po vložení SIM karty, připojení antény a napájení je GSM modul ihned připraven k použití. V případě použití USB komunikace je ještě nutné stáhnou a nainstalovat USB ovladač [13].

Základní nastavení jsme provedli přes HyperTerminál. Pro komunikaci se zařízením je potřeba nastavit konfiguraci RS232 portu podle tabulky (7.2).

7.2 AT příkazy pro obsluhu GSM modemu

Dnes tyto AT příkazy používají téměř všechny GSM modemy. AT příkazový jazyk byl vymyšlený právě pro jednoduchou konfiguraci modemů. Každý příkaz začíná dvojicí znaků AT (z anglického ATtention). Každý příkaz se ukončuje znaky `<CR>`,

Tab. 7.2: Konfigurace RS232 portu.

Port:	COMx
Přenosová rychlost:	115200 bps
Datové bity:	8
Parita:	None
Stop bity:	1
Řízení toku:	Hardware

tedy stiskem klávesy Enter. V tabulce (7.3) uvedeme základní příkazy a odpovědi z GSM modulu, které jsme použili pro počáteční konfiguraci našeho modulu. Více informací o AT příkazech lze nalézt v dokumentech [14] nebo [15].

Tab. 7.3: AT příkazy.

AT příkaz	Popis příkazu	Odpověď GSM modemu
AT	Otestuje komunikaci.	OK
AT+CSQ	Otestuje sílu přijmaného signálu.	+CSQ: signal, chybovost (11-31) uspokojivý signál
AT+CREG?	Ověření registrace do sítě.	AT+CREG: 0, 0 (neregistrováno) AT+CREG: 0, 1 (registrováno)
AT+CMGF=0	Přepnutí mezi textovým a PDU formátem zprávy. (0 = PDU)	OK
AT&W	Uložení aktuální konfiguraci do EEPROM.	OK

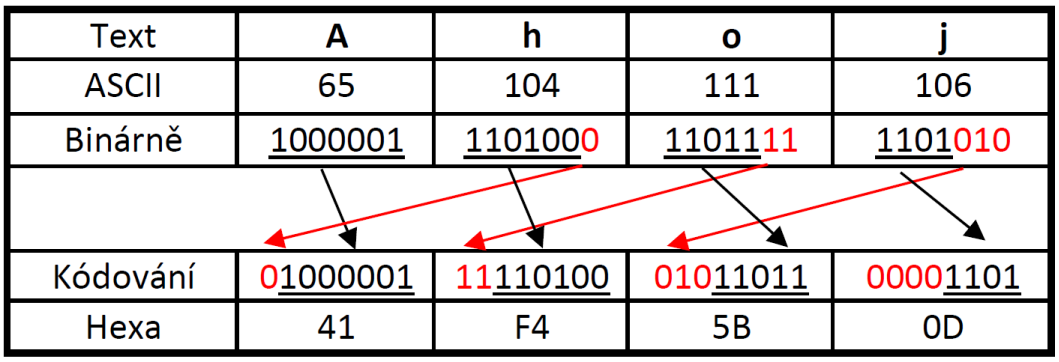
7.2.1 PDU formát zprávy

Pro odesílání SMS zpráv lze v GSM modemu zvolit dva módy. Textový mód není podporován všemi mobilními zařízeními [17] a zároveň nad odesíláním nemáme takovou kontrolu jako v PDU módu. Z těchto důvodu budeme odesílat SMS v módu PDU.

Tento formát zvládne přenést až 160 znaků v jedné zprávě, ovšem nebude přenesena diakritika. Pro odeslání zprávy s maximální délkou 160 znaků se používá 7-bitové kódování [16].

Převod textu zprávy do PDU formátu

V ASCII tabulce je každé písmeno reprezentováno číselným kódem. Základní ASCII tabulka obsahuje 128 znaků. To znamená, že v binárním vyjádření bit s nejvyšší hodnotou u každého jednotlivého znaku bude mít hodnotu 0. Proto lze tyto bity u všech znaků vypustit a každý znak vyjadřovat jen 7-bity. Toho bylo využito v kódování PDU, kdy 8 znaků se vejde do 7 oktetů. Postup zakódování zprávy „Ahoj“ do PDU formátu je znázorněn na obrázku (7.2) [18].



Obr. 7.2: Ukázka kódování zprávy „Ahoj“.

Nejdříve se ASCII hodnoty jednotlivých znaků převedou na 7-bitové binární čísla. První oktet PDU formátu vznikne tak, že se na nejvyšší pozici oktetu k sedmi bitům prvního znaku „A“ přidá bit s nejnižší hodnotou z druhého znaku „h“. Ze znaku „h“ nám zbývá už jen 6 bitů, k nim tedy musíme opět na nejvyšší pozici oktetu přidat dva nejnižší bity ze znaku „o“, abychom opět vytvořili kompletní oktet. Takto pokračujeme s vytvářením oktetů, dokud nám nedojdou znaky v původní zprávě. Až se tak stane a poslední oktet není kompletní, doplníme jeho nejvyšší hodnoty nulami, jak je vidět na obrázku (7.2).

Složení PDU rámce

Nyní se podíváme na složení PDU rámce, který pak následovně odesíláme do GSM modemu. Složení je vidět v tabulce (7.4).Více informací lze nalézt v dokumentu [18].

Tab. 7.4: Složení PDU rámce

TYP	OKTET	Popis
SMSC	00	Hodnota značí délku následujícího telefoního čísla SMS centra v oktetech. Pokud je zde zadaná hodnota „00“, jako je tak v našem případě, bude číslo SMS centra doplněno GSM modemem.
TYP	11	Identifikátor typu PDU zprávy. V našem případě jde o typ SMS-SUBMIT. Více informací naleznete v dokumentu [19].
MR	00	Referenční číslo zprávy poslané z GSM modemu do SMS centra. Hodnota „00“ značí, že si číslo doplní sám GSM modem.
DA	0C	Počet číslic cílového telefoního čísla. Například číslo „420 123 456 789“ má 12 číslic, to je hodnota „0C“ hexa.
	91	Identifikace formátu telefoního čísla příjemce. Hodnota „91“ značí mezinárodní formát.
	24 10 32 54 76 98	Telefoní číslo cíle. Číslo se zadává tak, že dvojice znaků je mezi sebou vždy prohozena. V našem případě jde tedy o číslo „420 123 456 789“.
PID	00	Informace pro SMS centrum s přídavnými parametry pro doručení zprávy.
DCS	00	Informace o kódování textu zprávy. Hodnota „00“ znamená 7-bitové kódování.
VP	0B	Časový limit na doručení zprávy. Hodnota „0B“ znamená jednu hodinu.
UDL	04	Počet oktetů přenášené již zakódované zprávy.
UD	41 F4 5B 0D	Přenášená zakódovaná zpráva. V naší ukázce jde o zprávu „Ahoj“.

Odeslání PDU rámce GSM modemu

Pokud hodnoty z tabulky (7.4) poskládáme postupně za sebe, dostaneme připravený PDU rámec, který můžeme odeslat připojenému a přednastavenému GSM modemu.

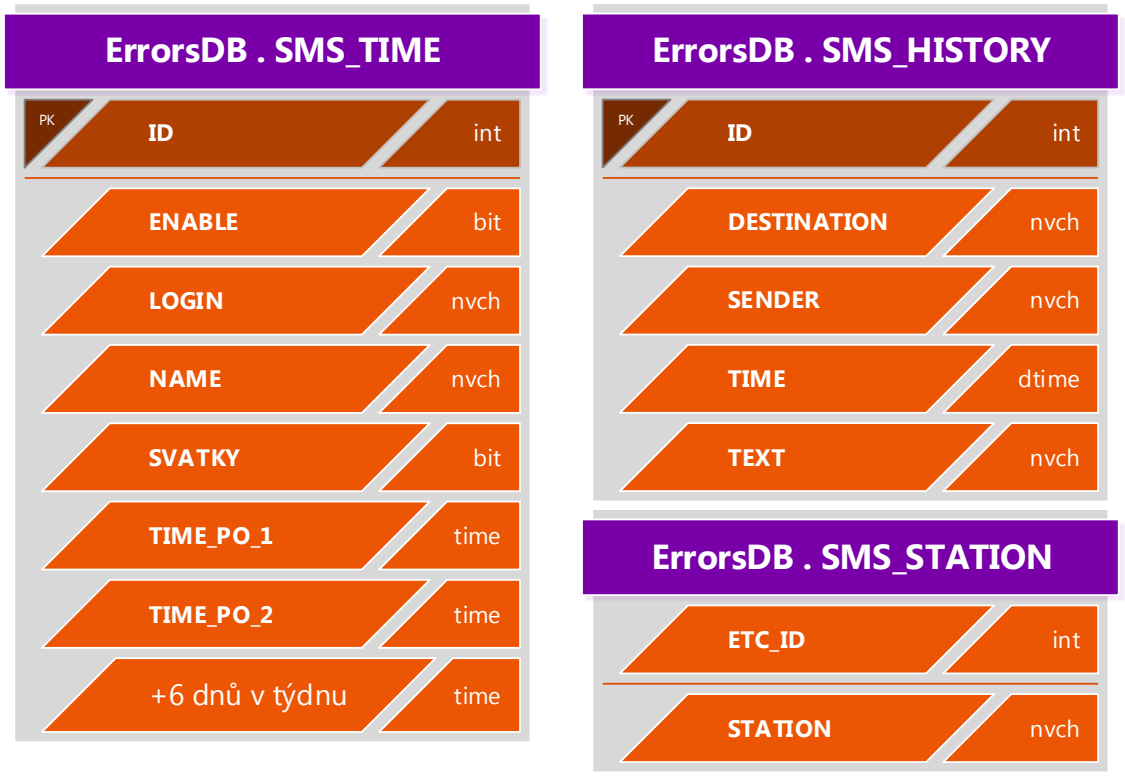
Nejdříve je nutné zaslat příkaz pro odeslání SMS zprávy „AT+CMGS“. Jako parametr zadáme počet oktetů celého rámce –1. Odečítáme jeden oktet, protože se do této hodnoty nezahrnují dvě počáteční nuly. Odeslání potvrdíme znaky `<CR>`. Následně obdržíme od GSM modemu znak „>“, který značí, že modem čeká na

zadání zprávy. Odešleme vytvořený PDU rámec a potvrdíme „<Ctrl+Z>“ což je ASCII znak 26. Následně obdržíme od GSM modemu informaci o referenčním čísle zprávy (MR), kterou přidělil zprávě GSM modem. Zároveň také potvrzení zda byla zpráva odeslána „OK“ nebo došlo k chybě „ERROR“.

```
AT+CMGS= 18 <CR>
> 0011000C91241032547698000000B0441F45B0D <Ctrl+Z>
+CMGS: 198
OK
```

7.3 Datová struktura pro odesílání SMS

Na obrázku (7.3) je ukázána SQL datová struktura, kterou využíváme pro automatický proces odesílání SMS zkušebním inženýrům.



Obr. 7.3: Struktura databáze pro odesílání SMS.

Tabulka **ErrorsDB.SMS_TIME** sloužící pro ukládání individuálních nastavení zkušebních inženýrů pro odesílání SMS:

ID: Jednoznačný identifikátor jednotlivých prostojů. Číslo se s nově vznikajícími prostoji postupně inkrementuje.

ENABLE: Bit určující zda daný zkušební inženýr chce automaticky generované SMS zasílat.

LOGIN: Přihlašovací jméno zkušebního inženýra do počítačů v přístupové síti.

NAME: Jméno zkušebního inženýra.

SVATKY: Bit rozhodující o přijímání či nepřijímání v den státního svátku.

TIME_PO_1: Čas od kdy chce v pondělí zkušební inženýr přijímat SMS.

TIME_PO_2: Čas do kdy chce v pondělí zkušební inženýr přijímat SMS.

+6 dnů v týdnu: Stejné dvojice časů jako v předchozích dvou sloupcích. Jsou vytvořeny i pro následující dny v týdnu.

Tabulka **ErrorsDB.SMS_HISTORY** sloužící pro ukládání historie všech odesílaných SMS zpráv:

ID: Jednoznačný identifikátor jednotlivých prostojů. Číslo se s nově vznikajícími prostoji postupně inkrementuje.

DESTINATION: Login cíle kterému je SMS zpráva odesílána.

SENDER: Identifikátor odesílatele.

TIME: Čas kdy byla SMS zpráva odeslána.

TEXT: Text odesílané zprávy.

Tabulka **ErrorsDB.SMS_STATION** sloužící pro ukládání z jakých zkušebních stanic, kromě jemu přiřazených, si přeje zkušební inženýr také přijímat upozornění na výpadek:

ETC_ID: Identifikační číslo zkušebního inženýra.

STATION: Název stanice.

7.4 Implementace systému pro odesílání SMS

Do našeho systému jsme tedy museli naimplementovat grafické prostředí, kde si zkušební inženýr navolí individuální konfiguraci časů pro přijímání automaticky generovaných SMS zpráv. Pro obsluhu GSM modulu jsme naimplementovali neinstancovatelný Module s názvem `smsModule`. Dále pak webovou stránku, která bude přijímat žádosti na odeslání SMS zprávy z jiných aplikací na oddělení ETC.

7.4.1 Grafické rozhraní pro konfiguraci přijímání SMS

Na stránce `/SMStime.aspx`, jejíž hlavní část je na obrázku (7.4), si může zkušební inženýr nastavit kdy a zda vůbec chce přijímat automaticky generované SMS zprávy.

Obr. 7.4: Konfigurace doručování SMS zprávy.

Při úplně prvním navštívení stránky se uživateli nejprve zobrazí jen nezaškrtnutý CheckBox s řádkem „Celkové povolení / zakázání odesílaní SMS“. Při zaškrtnutí zmíněného CheckBoxu se uživateli vytvoří záznam v databázi a zobrazí zbytek stránky. Tímto CheckBoxem si pak může uživatel kdykoli rychle zakázat či povolit přijímání SMS zpráv.

V nově otevřeném panelu si pak může uživatel kdykoli nastavovat časy doručování v jednotlivých dnech. Zároveň je zde také TextBox pro zadání telefonního čísla. Dále pak CheckBox, zda chce daný uživatel dostávat upozornění i v den státního svátku. Pro přehlednost, čí konfigurace je editována, je v panelu velkým písmem napsaný přihlašovací jméno uživatele. Konfigurace se uloží stisknutím tlačítka „Uložit“.

Pokud si uživatel přeje být upozorňován i ze stanic, ke kterým není přiřazen, může si v panelu na obrázku (7.5) zvolit, ze kterých dalších stanic chce dostávat upozornění. Tento panel se nachází hned pod panelem pro nastavování časů a čísla na obrázku (7.4).

Pro rychlejší volbu stanic je přidán DropDownList, který po vybrání týmu způsobí zaškrtnutí pouze těch stanic, které přísluší danému týmu. Dále jsou tam i tlačítka pro zaškrtnutí všech nebo žádné stanice. Po stisknutí tlačítka „Uložit“, se do

Obr. 7.5: Konfigurace přídavných upozornění ze stanic.

tabulky `ErrorsDB.SMS_STATION` vždy uloží datový pár identifikační číslo inženýra a jméno stanice.

7.4.2 Modul na obsluhu GSM modemu

Neinstancovatelný modul `smsModule.vb` má na starosti obsluhu GSM modemu přes COM port, frontu SMS zpráv a vytváření PDU rámce.

Do modulu se požadavek na odeslání zprávy dostane zavoláním public funkce `OdesliSMS(cisloPrijemce, textZpravy)`. V modulu se požadavek zařadí do fronty a spustí se vlákno na odesílaní SMS zpráv z fronty, pokud již vlákno není spuštěné.

Vlákno na odesílání SMS zpráv si načte požadavek z fronty s číslem a textem zprávy. Dále se pak zkontroluje validita čísla, odstraní se diakritika a případné nadbytečné znaky ve zprávě. Následovně dojde k vytvoření PDU rámce a odeslání přes COM do GSM modemu. Pokud mezitím nepřibyly ve frontě další požadavky na odeslání zprávy, vlákno se ukončí.

7.4.3 Automatické generování SMS zpráv při výpadku

Pro ověření, zda daný uživatel chce v příslušný moment výpadku přijímat SMS zprávy, jsme na straně SQL serveru vytvořili procedury. Pokud vybranou proceduru zavoláme příslušným SQL dotazem, databázový server nám v output parametru `@tel` vrátí buď hodnotu „0“ jako indikaci, že zprávu v tuto chvíli přijímat nechce nebo přímo telefonní číslo což znamená, že si uživatel přeje v tento moment přijímat zprávy.

Pokaždé, když se bude zakládat nový záznam o prostoji v `/Errors.aspx`, zavoláme příslušnou funkci, která pomocí SQL procedury, ověří zda chce uživatel zprávu

přijmout. Pokud ano, uloží záznam o SMS do tabulky `ErrorsDB.SMS_HISTORY` s historií a odešle požadavek na odeslání zprávy do modulu na odeslání SMS zpráv zavoláním funkce `OdesliSMS(cisloPrijemce, textZpravy)`.

Obsahem zprávy je informace o názvu stanice, časovým razítkem, kdy došlo k výpadku a textem prvního chybového hlášení z přiřazeného balíku chybových hlášení ze stanice. Z textu chybového hlášení už inženýr může poznat, co se s danou stanicí děje a příslušně reagovat, ať už je kdekoli.

7.4.4 Přijímání žádostí na odeslání SMS zprávy z jiných aplikací

Aby se dal námi vytvořený systém na odesílání SMS zpráv použít i v jiných aplikacích oddělení ETC, museli jsme vytvořit přijímací bránu, která bude požadavky z ostatních aplikací zpracovávat. S výhodou jsme opět vytvořili prázdnou webovou stránku, na kterou budou pomocí QueryStringu obsaženém v URL adrese zasílat požadavky na odeslání zprávy. My tyto požadavky pak budeme zpracovávat v Code behind stránky. Hodnoty QueryStringu jsou přenášené v kódování Base64, aby nebyl na první pohled zřejmý obsah požadavku.

Požadavky se na příslušnou adresu mohou zasílat v těchto třech formátech QueryStringů:

```
.aspx ? tel=123456789 & sender=odesílatel & text=zpráva
```

```
.aspx ? login=login & sender=odesílatel & text=zpráva
```

```
.aspx ? etc_id=identifikačníČíslo & sender=odesílatel & text=zpráva
```

V druhém a třetím případě kdy je v QueryStringu přenášeno `ETC_ID` nebo `login` dojde opět k ověření, zda chce daný uživatel v tento moment SMS zprávy přijímat. V případě telefonního čísla v QueryStringu k ověření nedochází a požadavek na zprávu je odeslán přímo.

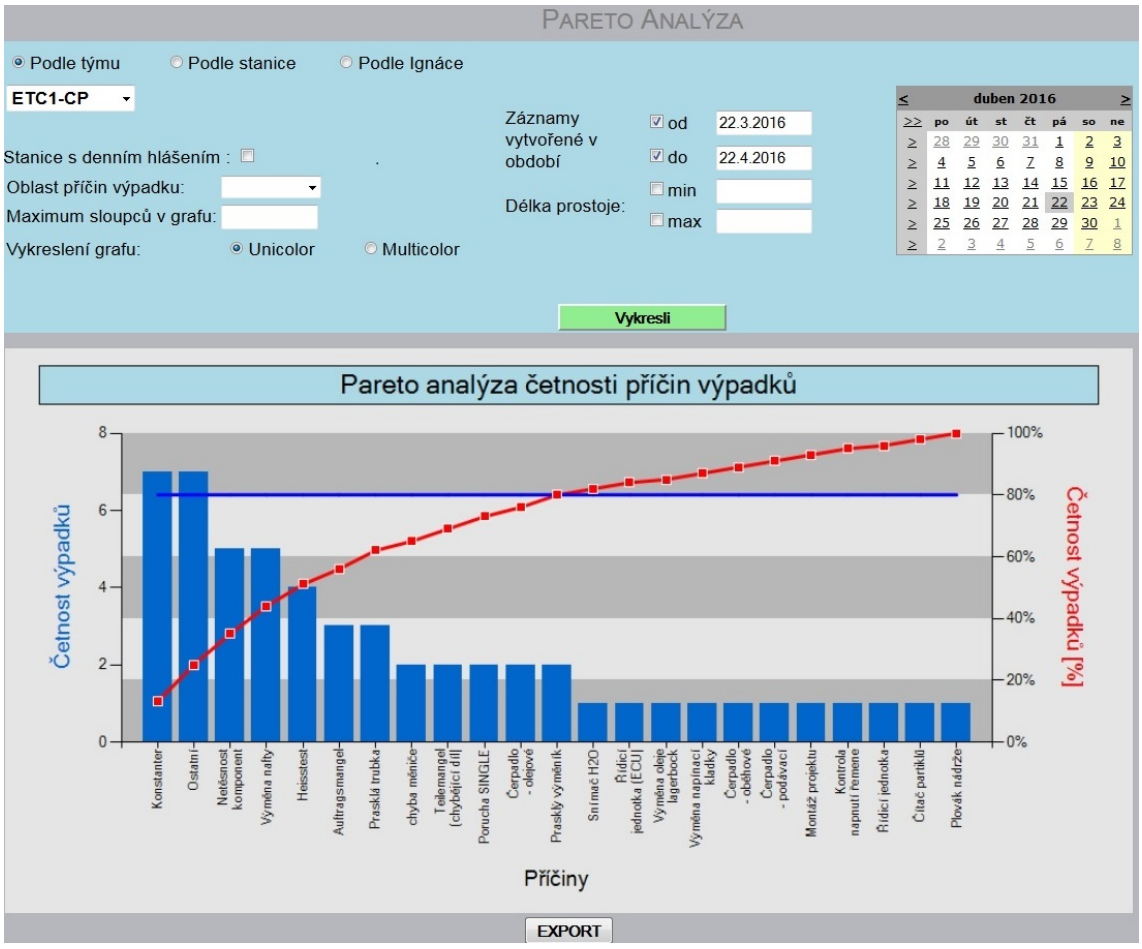
Při odesílání požadavku na odeslání zprávy se zároveň opět uloží záznam do tabulky `ErrorsDB.SMS_HISTORY` s historií.

8 PARETO ANALÝZY A STATISTIKY

Posledním zbývající bodem diplomové práce bylo navrhnout grafické rozhraní pro generování statistik a Pareto analýz z nasbíraných dat. Za tímto účelem jsme navrhli dvě stránky `/graph.aspx` a `/graph_solo.aspx`. Na obou stránkách si může zkušební inženýr ve výběrového filtru nejdříve zvolit, jaká data chce do statistik zahrnout. Po stisknutí tlačítka „Vykresli“ se mu požadované grafy vykreslí.

8.1 Pareto analýza

Na stránce `/graph.aspx` si může uživatel vykreslit Pareto analýzy četnosti příčin výpadků a prostátých hodin. Výběrový filtr s prvním grafem na stránce je na obrázku (8.1).



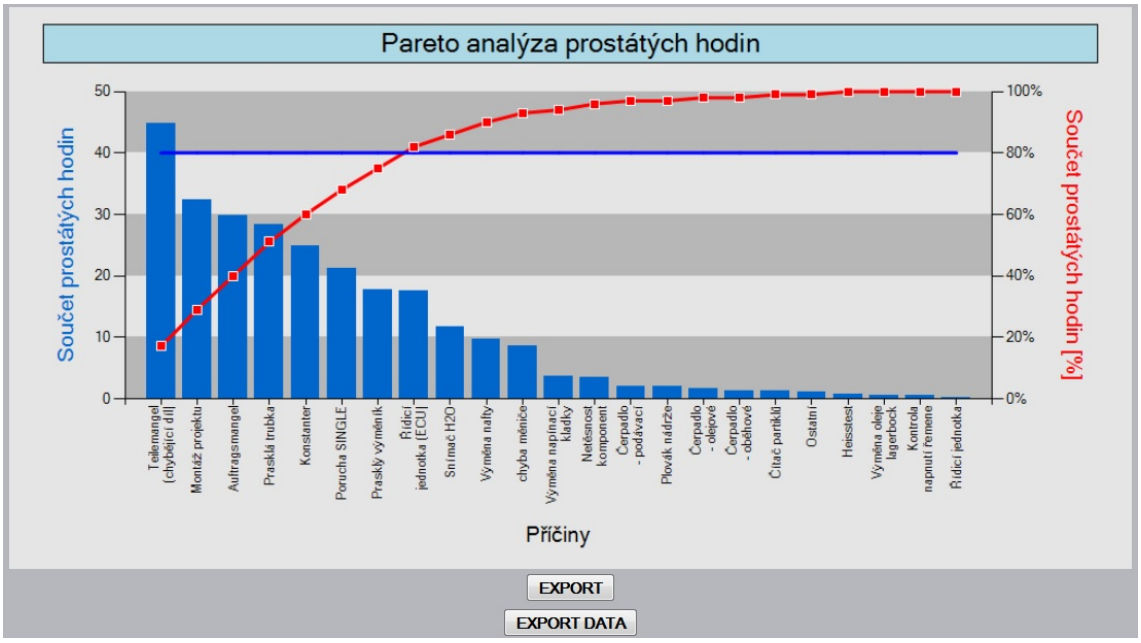
Obr. 8.1: Pareto analýza četnosti příčin výpadků.

Ve vrchní části obrázku (8.1) je vidět výběrový filtr, ve kterém si uživatel může zvolit základní volbu, na základě čeho chce dané záznamy o prostoji vybrat. Volit

může z výběru podle týmu, podle konkrétních stanic nebo podle konkrétních zkušebních inženýrů. Pokud vybere podle týmu a následně žádný konkrétní tým nevybere, vykreslí se do grafu všechny záznamy o prostoji. Pokud zvolí podle stanice nebo podle zkušebních inženýrů, zobrazí se další panel se zaškrtávacím seznamem stanic nebo zkušebních inženýrů.

Dále může uživatel upřesnit oblast příčin výpadku (Elektrická, Mechanická, Hydraulická, ...) a zda chce do statistik zahrnout prostoje z ostatních stanic generované denně v tabulce `ErrorsDB.CENTRAL_DAILY`. Vybrat si také může jen ty prostoje s konkrétní časovou délkou. Při velkém množství dat si může uživatel v grafu zobrazit pouze několik prvních sloupců, přičemž Paretova křivka bude počítána ze všech původních sloupců.

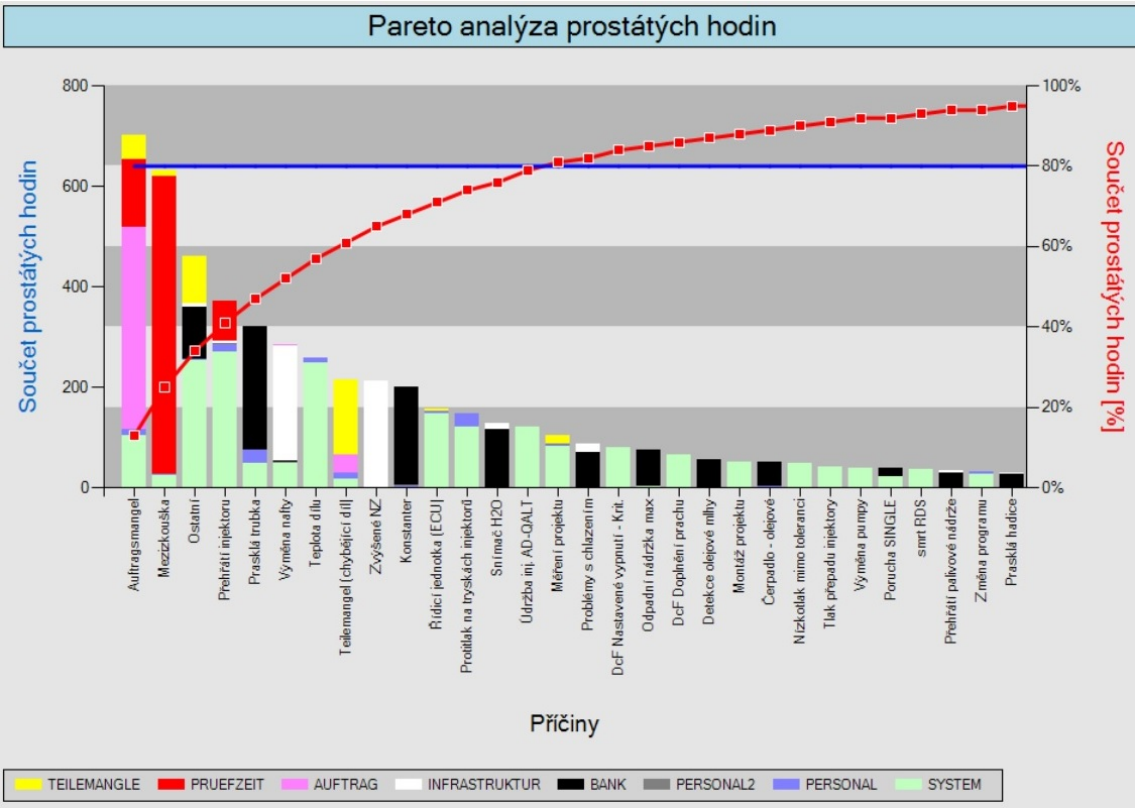
Vybrané data pak může uživatel dále zúžit, vybráním časového rozmezí (v kalendáři nebo dopsáním do příslušných TextBoxů), které prostoje od kdy do kdy chce do statistik zahrnout. Na základě této volby nejsou zobrazeny jen ty prostoje, které vznikly a zároveň byly ukončeny ve vybraném rozmezí, ale i ty, které přechází přes zvolené časové hranice. Pro Pareto analýzu prostátých hodin v druhém grafu na stránce na obrázku (8.2) je u těchto prostojů přecházejících přes vybranou časovou hranici, do statistik započítána jen ta část konkrétního prostoje, která spadá mezi vybrané časové hranice. Díky tomu nejsou statistiky zkresleny nadměrně dlouhými prostoji.



Obr. 8.2: Pareto analýza prostátých hodin „Unicolor“.

Uživatel si také může ve výběrovém filtru zvolit „Multicolor“ vykreslení Paretovy analýzy s prostátými hodinami. Ukázka „Multicolor“ módu je vidět na ob-

rázku (8.3). Barvy odpovídají kategoriím do kterých zkušební inženýři rozdělovali prostoje u jednotlivých výpadků. Uživatel si dále může nechat vykreslit data pouze z jedné či více kategorií, kterého právě zajímají. Tato možnost volby se zpřístupní ve výběrovém filtru po zvolení módu „Multicolor“.



Obr. 8.3: Pareto analýza prostátých hodin „Multicolor“.

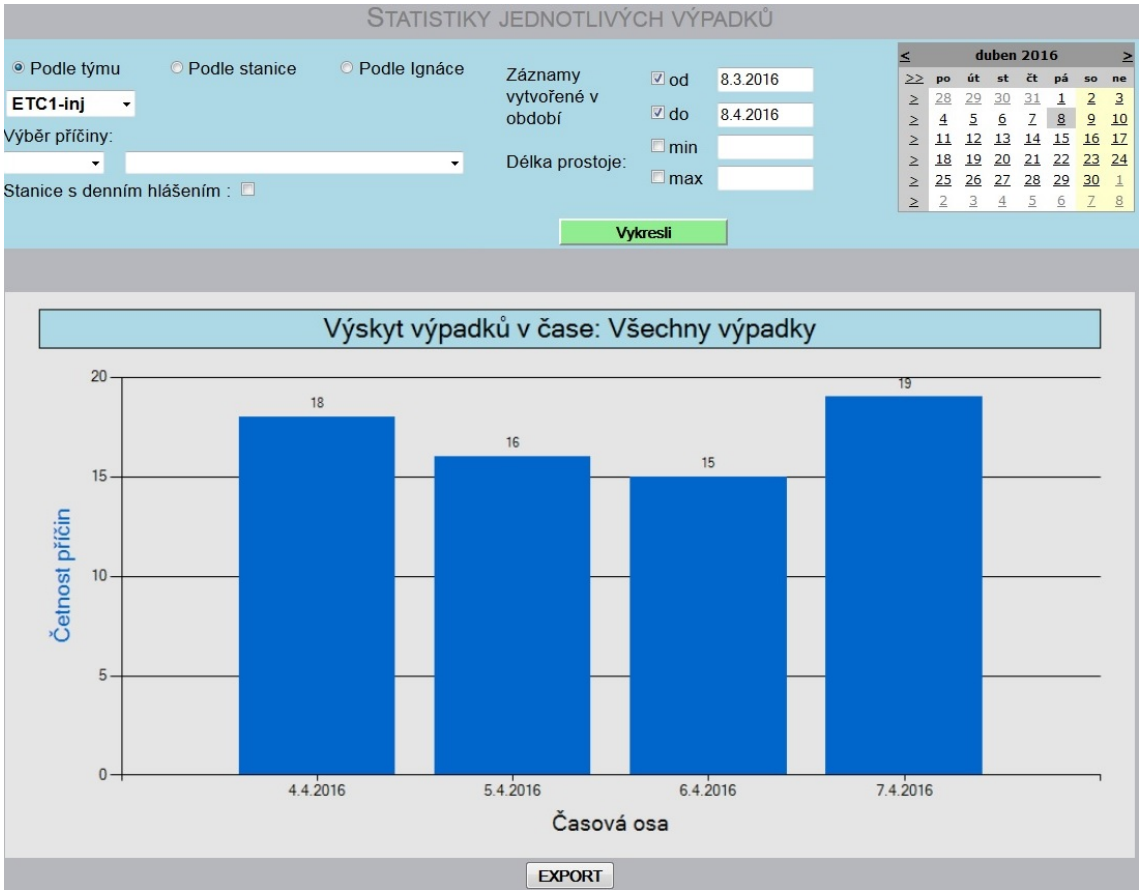
Pokud se oddělení ETC například rozhodne snížit prostáté hodiny, které byly přiřazeny do kategorie `SYSTEM`, ve výběrovém filtru si tedy zvolí vykreslit pouze prostáté hodiny příslušné kategorie. Ve výsledné Paretově analýze pak hned vidí příčiny prostojů, které způsobují největší prostoje právě v této kategorii a mohou přijmout příslušná opatření.

Po kliknutí na tlačítka „EXPORT“ pod vykreslenými grafy, si může daný uživatel vyexportovat výsledné statistiky ve formě JPEG obrázku. Po kliknutí se mu zobrazí klasický dialog, zda chce obrázek rovnou otevřít nebo uložit. Uživatel si také může nechat vyexportovat tabulku s daty, na základě kterých byly vykresleny příslušné zobrazené grafy na stránce. Tabulky budou vyexportovány ve formě souboru Microsoft Office Excel.

8.2 Statistiky jednotlivých příčin výpadků

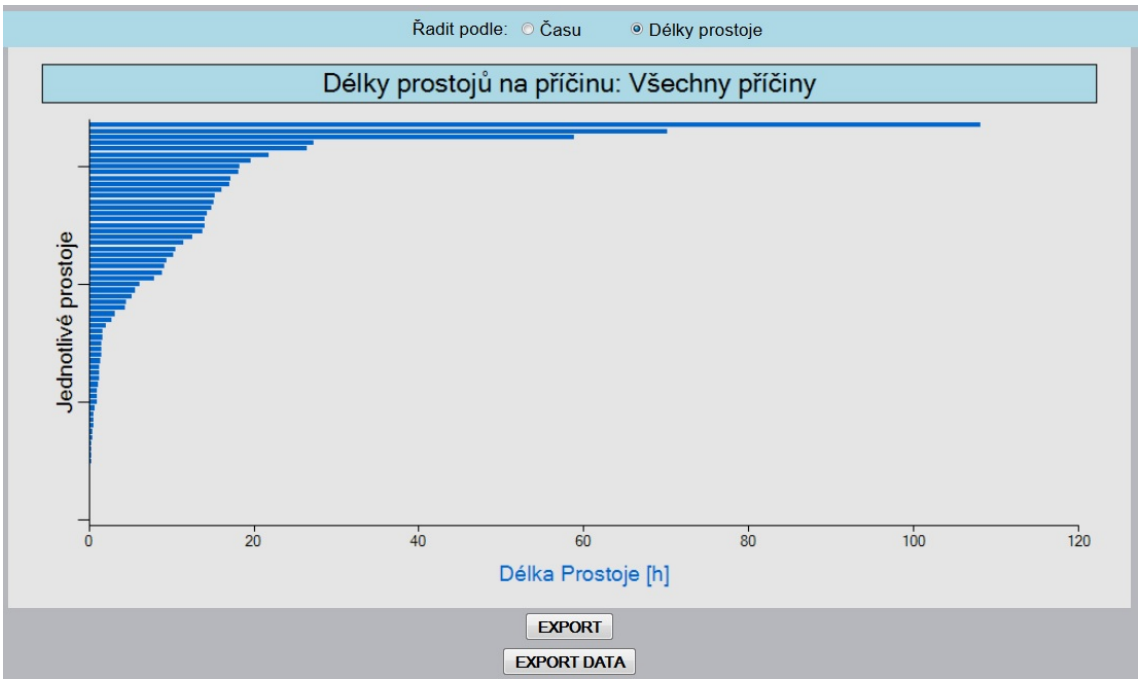
Statistiky jednotlivých příčin výpadků si uživatel může vykreslit na webové stránce `/graph_solo.aspx`. Výběrový filtr je částečně podobný jako u stránky `/graph.aspx`. Uživatel si opět může zvolit na základě jakého identifikátoru (tým, stanice, zkušební inženýr) chce nechat vybrat data do grafu. Data může blíže specifikovat časovým rozmezím nebo délkou jednotlivých prostojů. Novou možností je volba konkrétní příčiny výpadku. Pokud uživatel žádnou nezvolí, budou opět vykresleny grafy zahrnující prostoje se všemi příčinami výpadku.

V prvním grafu na obrázku (8.4) se vykreslí sloupcoví graf s údaji o tom, ke kolika výpadkům došlo v daný den na konkrétní zvolenou reálnou příčinu výpadku.



Obr. 8.4: Výskyt výpadků v čase.

Ve druhém grafu na obrázku (8.5) je vykreslen pruhový graf s délkami jednotlivých prostojů na zvolenou konkrétní příčinu výpadku. Data lze v tomto grafu seřadit na základě délky prostoje nebo času vzniku prostoje. Na základě těchto seřazení uživatel může sledovat, jak se v čase vyvíjí rychlost opětovného spuštění stanice po výpadku na konkrétní příčinu nebo průměrnou dobu prostojů na konkrétní příčinu výpadku.



Obr. 8.5: Délky prostojů na konkrétní zvolenou příčinu výpadku.

Stisknutím exportovacích tlačítek lze grafy a data opět exportovat podobně jako na stránce pro Pareto analýzu.

9 ZÁVĚR

Zadáním diplomové práce je vytvořit monitorovací systém výpadků na zkušebních stanicích společnosti Bosch Diesel s.r.o. v Jihlavě a to konkrétně na oddělení dlouhodobé zkušebny ETC.

Na začátku práce jsme se krátce seznámili s vnitřní strukturou tohoto oddělení a zkušebními stanicemi, se kterými bude náš systém pracovat. Také jsme popsali průběh u dlouhodobých zkoušek a nové trendy.

V další části práce jsme probrali všeobecné monitorovací systémy SCADA a ukázali princip Pareto analýzy, pro kterou budou výstupní data z našeho systému sloužit jako podklady.

Následně jsme provedli podrobný rozbor požadavků zadavatele a na základě tohoto rozboru jsme navrhli základní hlavní funkce našeho systému. Celý náš monitorovací systém poběží na serveru oddělení ETC. V této části je také podrobně popsáno rozvržení monitorovacího systému a nastínění technologie ASP.NET, kterou budeme používat pro realizaci monitorovacího systému.

V další části této práce se již zabýváme konkrétním sběrem chybových hlášení z CPS u zkušebních stanic. Tato hlášení jsou důležitá, protože uživatel pomocí těchto hlášení může určit skutečnou příčinu zastavení měřící stanice. Nejdříve jsme provedli rozbor *„Logfile.cps“*, ve kterém jsou hlášení na CPS ukládána a následně navrhli jakým způsobem a kam se budou ukládat. Hlášení o chybách se ukládá do serverové databáze MS SQL v balících podle času vymazání.

Samotnou implementaci sběru hlášení jsme provedli jako webovou aplikaci využívající samostatná vlákna na pozadí, periodicky aktivovanou pomocí Windows Task Scheduleru. Implementované řešení je velice robustní a bezproblémové. Díky aktivace pomocí Task Scheduleru není systém nikterak náchylný na restart serveru a to činí tuto část monitorovacího systému prakticky bezúdržbovou. Po dobu jednoho měsíce tento systém nasbíral ze zkušebních stanic okolo 110 000 hlášení o chybách.

Následně jsme se v práci zabývali výpočty prostojů zkušebních stanic. Výpočet těchto prostojů jsme nemohli provést na základě chybových hlášení od CPS, protože zapisování těchto hlášení zkušební stanicí je poměrně nespolehlivé. Zvolili jsme tedy alternativní možnost výpočtu nezávislou na hlášení z těchto stanic. Výpočty provádíme pomocí hlášení o Startu či Stopu zkušební stanice. Pokud stanice stále stojí, uživatele vždy zajímá aktuální doba prostoje, proto je nutné provádět periodicky opakující se dopočet.

Implementaci jsme provedli podobně jako u sběru chybových hlášení. Je spouštěna také Task Schedulerem a využívá vlákna na pozadí. Záznamy o prostojích jsou vedeny v serverové databázi. Záznam o prostoji se automaticky prováže s balíkem

chybových hlášení, který by měl odpovídat příčině tohoto prostoje. Tato část systému opět funguje bez větších potíží.

Dále bylo nutné vytvořit grafické rozhraní, kde budou moci zkušební inženýři každodenně editovat záznamy o prostojích. Úkolem zkušebního inženýra je vždy u každého prostoje přiřadit reálnou příčinu výpadku a rozdělit prostátý čas do jednotlivých kategorií. Proces editace jsme navrhli tak, aby zbytečně zkušebního inženýra nezatěžoval a délka provádění tohoto úkonu byla co možná nejkratší. Také jsme navrhli speciální grafické rozhraní sloužící pro každodenní týmové porady.

Jednou ze stěžejních částí této práce bylo navrhnout a implementovat automatické upozorňování zkušebních inženýrů na výpadek zkušební stanice pomocí SMS zpráv. Automatické SMS zprávy odesíláme pomocí GSM modemu *Sierra Wireless Airlink Fastrack FXT009* obsluhovaným přes AT příkazy. Samotné SMS zprávy jsou odesílány v PDU formátu se 7-bitovým kódováním. Uživatel si může individuálně zvolit časy, kdy chce tyto SMS upozornění přijímat a z jakých dalších zkušebních stanic kromě těch, které jsou přiřazeny přímo jemu.

Součástí naší implementace je také brána, která přijímá žádosti na odeslání SMS i z jiných aplikací na oddělení ETC. Toto včasné upozorňování na výpadek vedlo k rychlejším reakcím zkušebních inženýrů, a tím tedy ke zkrácení celkového prostátého času zkušebních stanic.

Pro vykreslování výsledných statistik jsme vytvořili dvě stránky, na kterých si uživatel může ve výběrovém filtru zvolit, které záznamy chce do statistik zahrnout a následně si grafy vykreslit. Na první stránce si uživatel může nechat vykreslit Pareto analýzu četnosti příčin výpadků a Pareto analýzu prostátých hodin na jednotlivé příčiny výpadku. Na druhé stránce si pak uživatel může nechat vykreslit sloupcový graf počtu výpadků na konkrétní příčinu v čase. Dále se vykreslí pruhový graf s délkou prostojů na konkrétní příčinu výpadku. V tomto grafu si může uživatel zvolit řazení dat podle času nebo délky prostoje.

Na základě těchto grafů se může oddělení ETC pružně zaměřit na řešení těch příčin výpadku, které vedou k nejčastějším nebo nejdelším prostojům a zvýšit tak efektivitu celého zkušebního procesu. Posléze také postupem času z grafu budou moci vyčíst, zda mělo konkrétní přijaté opatření pro odstranění příčiny výpadku dostatečný účinek nebo je třeba přijmout další opatření.

Po konzultaci se zadavatelem jsme ponechali systém sběru hlášení ze zkušebních stanic, výpočet prostojů zkušebních stanic a obsluhou GSM modemu v samostatném systému *CPS_Errors*, ve kterém jsme tento systém celou dobu vyvíjeli. Tento systém poběží paralelně k formulářovému systému *CML*, do kterého jsme implementovali pouze grafické rozhraní pro editaci prostojů, vykreslování statistik a individuální volbu přijímání automaticky generovaných SMS. Oba systémy jsme nechali běžet paralelně z toho důvodu, že formulářový systém *CML* se neustále rozšiřuje a upravuje

a to by narušovalo hladký a ničím nerušený chod sběru hlášení, výpočtu prostojů zkušebních stanic a odesílání SMS zpráv.

V době psaní této práce byl systém sledování výpadků v ostrém provozu necelé dva týdny. Už takto krátké období stačilo na to, aby ze statistik začaly vycházet smysluplné a zajímavé informace. Samotní zkušební inženýři byli statistikou za prvních několik dnů překvapeni a sami začali aktivně pracovat na odstranění příčin některých výpadků. K tomu dopomáhá i systém zlepšovacích návrhů, který je ve firmě zaveden a jehož cílem je finančně motivovat zaměstnance ke zlepšování procesů a snižování nákladů. Naše Pareto analýzy tak na jedné straně slouží jako vodítko pro identifikaci problémů, na straně druhé pak zpětné jako nástroj pro rychlé a zároveň objektivní vyčíslení úspor po zavedení opatření. Webová aplikace, předkládaná v této diplomové práci, tak prakticky okamžitě po spuštění začala u zadavatele měřitelným způsobem přispívat k optimalizaci procesů a snižování nákladů.

LITERATURA

[1] ZIKMUND, M. *Paretova (ABC) analýza – mocný nástroj v logistice, marketingu i obchodu* [online]. 2011, [cit. 10. 7. 2015]. Dostupné z URL: <http://www.businessvize.cz/rizeni-a-optimalizace/paretova-abc-analyza-mocny-nastroj-v-logistice-marketingu-i-obchodu>.

[2] STŘELEC, J. *Pareto analýza* [online]. 2012, [cit. 10. 7. 2015]. Dostupné z URL: <http://www.vlastnicesta.cz/metody/pareto-analyza/>.

[3] PÁSEK, J. *Programovatelné automaty v řízení technologických procesů* [skripta]. Vysoké učení technické, Brno 2007, [cit. 10. 24. 2015].

[4] BALDA, P. *Informační a řídicí systémy I. SCADA a HMI systémy* [online]. 2012, [cit. 10. 24. 2015]. Dostupné z URL: <http://vendulka.zcu.cz/Download/Free/IRS1/IRS1-08_SCADA_HMI.pdf>.

[5] CORAL *SCADA/HMI systém TIRSWeb* [online]. 2015, [cit. 10. 24. 2015]. Dostupné z URL: <http://www.coral.cz/scada-hmi-systemy-tirs/scadahmi-system-tirsweb/>.

[6] BALÁŽIK, M. *Kyberútoky na kritickou infrastrukturu* [online]. únor 2015, [cit. 11. 2. 2015]. Dostupné z URL: <http://www.corpus.cz/tiskove-centrum/kyberutoky-na-kritickou-infrastrukturu>.

[7] ŠENOVSKÝ, P. *Bezpečnostní informatika III* [online]. 2007, [cit. 11. 2. 2015]. Dostupné z URL: <http://homen.vsb.cz/~sen76/inform/bi3.pdf>.

[8] KUŽEL, S. *Kybernetická kriminalita V: Cyberwar už není Sci-Fi* [online]. 2012, [cit. 11. 2. 2015]. Dostupné z URL: <http://www.businessit.cz/cz/kyberneticka-kriminalita-v-cyberwar-uz-neni-sci-fi.php>.

[9] ČAPKA, D. *Úvod do ASP.NET* [online]. 2012, [cit. 11. 29. 2015]. Dostupné z URL: <http://http://www.itnetwork.cz/csharp/asp-net/tutorial-uvod-do-asp-dot-net>.

[10] HERCEG, T. *Úvod do ASP.NET 2.0* [online]. 2007, [cit. 11. 29. 2015]. Dostupné z URL: <http://www.dotnetportal.cz/clanek/44/Uvod-do-ASP-NET-2-0>.

[11] SIERRA WIRELESS. *AirLink FX Series User Guide*. Rev 9.0. [online] 2014, [cit. 03. 03. 2016]. Dostupné z URL: <http://source.sierrawireless.com/resources/airlink/hardware_reference_docs/airlink_fxt_series_user_guide/#>.

[12] DIAFAAN COMMUNICATION SOFTWARE. *Sierra Wireless Airlink Fastrack Xtend GSM modem.* [online] [cit. 03. 04. 2016]. Dostupné z URL: <https://www.diafaan.com/reviews/sierra-wireless-airlink-fastrack-xtend-gsm-modem/>.

[13] SIERRA WIRELESS. *AirPrime – USB driver for Windows 7 & 8.1.* Version 3.8.9.0. [online] [cit. 03. 03. 2016]. Dostupné z URL: <http://source.sierrawireless.com/resources/airprime/tools/usb-driver-installer/>.

[14] SIERRA WIRELESS. *AT Commands Interface Guide for Firmware 7.52* . Version 6.0. [online] 2015, [cit. 03. 04. 2016]. Dostupné z URL: <http://source.sierrawireless.com/resources/airprime/software/>.

[15] AGNIHOTRI, N. *AT Commands, GSM AT command set* [online]. 2012, [[cit. 03. 04. 2016]. Dostupné z URL: <http://www.engineersgarage.com/tutorials/at-commands?page=1>.

[16] DH SERVIS. *AT příkazy mobilních telefonů.* [online] [cit. 03. 04. 2016]. Dostupné z URL: <http://www.dhservis.cz/dalsi/at_prikazy.htm>.

[17] EMBEDDED SYSTEMS. *Difference between PDU mode and Text mode for SMS.* [online] 2011, [cit. 03. 05. 2016]. Dostupné z URL: <http://electrotech99.blogspot.cz/2011/02/difference-between-pdu-mode-and-text.html>.

[18] NOVOTNÝ, V. *Odesílání SMS zpráv v PDU formátu* [návod k laboratorní úloze předmětu MKPM]. Vysoké učení technické, Brno 2015, [cit. 03. 05. 2016].

[19] STRAKA, J. *PDU formát SMS zpráv* [Semestrální práce]. České vysoké učení technické v Praze, Praha 2006, [cit. 03. 05. 2016]. Dostupné z URL: <http://radio.feld.cvut.cz/personal/mikulak/MK/MK06_semestralky/PDUformatSMSzprav_StrakaJ.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DS/ETC–Jh Diesel Systems / Engineering Testing Center v Jihlavě

ETC	Engineering Testing Center
CPS	Control Panel System
SCADA	Supervisory Control and Data Acquisition
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
MES	Manufacturing Execution System
CML	Centrální Mozek Lidstva
ASP	Active Server Pages
CR	Common Rail
WIS	Windows Integrated Security
MS	Microsoft
SQL	Structured Query Language
IIS	Internet Information Services
SMS	Short Message Service
UTC	Coordinated Universal Time
DST	Daylight Saving Time
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
GSM	Global System for Mobile communications
GPRS	General Packet Radio Service
EDGE	Enhanced Data Rates for GSM Evolution
USB	Universal Serial Bus

SIM	Subscriber Information Module
COM	Component Object Model
AT	ATtention
EEPROM	Electrically Erasable Programmable Read–Only Memory
PDU	Protocol Description Unit
ASCII	American Standard Code for Information Interchange
JPEG	Joint Photographic Expert Group
SMSC	Short Message Service Center
MR	Message Reference
DA	Destination Adress
PID	Protocol IDentifier
DCS	Data Coding Scheme
VP	Validity Period
UDL	User Data Length
UD	User Data

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

V přiloženém CD se nachází následující přílohy:

xsebes15.pdf Elektronická kopie diplomové práce ve formátu PDF.
readme.txt Textový soubor s popisem přiloženého zdrojového kódu.
CPS_Errors Zdrojový kód projektu.