VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Jiří Czipszer


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION


# VYHODNOCOVANÍ EFEKTIVITY VÝROBY
EVALUATING PRODUCTION EFECTIVITY

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Jiří Czipszer**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Jan Pásek, CSc.**
SUPERVISOR


**BRNO 2020**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

***Student:*** Bc. Jiří Czipszer ***ID:*** 174271
***Ročník:*** 2 ***Akademický rok:*** 2019/20

**NÁZEV TÉMATU:**

## Vyhodnocovaní efektivity výroby

**POKYNY PRO VYPRACOVÁNÍ:**

Cílem DP je návrh metody pro vyhodnocování efektivity výroby příkladové linky v programu APROL
1. Odborná rešerše sběru dat a vyhodnocovaní OEE
2. Optimalizace SQL databáze pro následné vyhodnocení ukazatelů OEE
3. Seznamte se s interním systémem APROL od společnosti B&R Automation
4. Popis řešení
5. Zobrazení dat ve formě OEE ukazatelů pomocí dashboardů

**DOPORUČENÁ LITERATURA:**

1. Charlotta Johnson,Combining Overall Equipment Efficiency (OEE) and Productivity Measures as Drivers for Production Improvements, on-line: http://portal.research.lu.se/portal/files/6184392/2224805.pdf

2. Peter Muchiri and Liliane Pintelon, Performance measurement using overall equipment effectiveness (OEE): literature review and practical application discussion, 2006

3. B&R Automation, APROL PDA Monitor and optimize performance online, on-line: https://download.br-automation.com/BRP4440000000000000384089/APROL%20PDA%20MM-BR-PA-PDA-EN-01.p-f?px-hash=b8e299ef0bd48485611c28e26a475b9d&px-time=1572001785

***Termín zadání:*** 3.2.2020 ***Termín odevzdání:*** 1.6.2020

***Vedoucí práce:*** Ing. Jan Pásek, CSc.
***Konzultant:*** Ing. Maroš Macej

**doc. Ing. Václav Jirsík, CSc.**
předseda oborové rady

## ABSTRAKT

Diplomová práce se zaobírá problematikou vyhodnocování efektivnosti a sběru dat linek ve výrobě, pomocí programu APROL od společnosti B&R automation. Cílem práce je vytvořit program splňující požadavek, který se následně porovná z již realizovanými projekty společnosti a bude sloužit k vytvoří standardu společnosti. První část se věnuje popisu teorii této problematiky a použitého softwaru. Ve druhé části je popsáno řešení simulátoru linky a celé koncepce sběru a vyhodnocování.

## KLÍČOVÁ SLOVA

PLC, MES, DCS, KPI, OEE, TEEP, PEE, OFE, MTTR, MTBF, Dashboard, Skorekarty, APROL, MariaDB, Jaspersoft.

## ABSTRACT

The master thesis deals with the issue of evaluation of efficiency and data collection of lines in production, using the APROL program from B&R automation. The aim of this work is to create a program completed requirement, which is going to be compared with already implemented projects of the company and will serve to create a company standard. The first part is devoted to the description of the theory of this issue and used software. The second part describes the solution of the line simulator and the whole concept of collection and evaluation.

## KEYWORDS

PLC, MES, DCS, KPI, OEE, TEEP, PEE, OFE, MTTR, MTBF, Dashboard, Scorecards, APROL, MariaDB, Jaspersoft.

CZIPSZER, Jiří. *Vyhodnocovaní efektivity výroby*. Brno, 2020, 81 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Jan Pásek, CSc.

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vyhodnocovaní efektivity výroby“ jsem vypracoval 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 uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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 1. 6. 2020 .....

podpis autora

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce, panu Ing. Janu Pásekovi, CSc., za odborné vedení, trpělivost a podmětné návrhy k diplomové práci. Také bych chtěl tímto poděkovat panu Ing. Zdeňkovi Švihálkovi, panu Ing. Marošovi Macejkovi a panu Ing. Janu Balcarovi ze společnosti B&R automation, za pomoc s vypsáním tématu práce, cenné rady a pomoc, které mi pomohly k dokončení této práce.

Brno 1. 6. 2020 .....

podpis autora

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů

# Úvod

V následujících stránkách textu předkládám čtenáři svou diplomovou práci na téma "Vyhodnocovaní efektivity výroby". To znamená, že celá tato práce se zabývá tématikou sběru a zpracování dat a formou klíčových ukazatelů neboli *Key Performance Indicators* (KPI). Předem musím říct, že před tím, než jsem začal uvažovat o tomto tématu, jsem pro svou diplomovou práci neměl žádné podvědomí o této problematice a své rozhodnutí jsem udělal po zběžném průzkumu, kde mě dané téma zaujalo a chtěl jsem se o něm dozvědět něco víc.

V poslední době dochází k velkému rozvoji konceptu Průmyslu 4.0, který zavádí do řady průmyslových odvětví nové technologie a přístupy. V současnosti se v průmyslu klade velký důraz na sběr a vyhodnocování dat. To je dáno hlavně požadavkem na zvýšení efektivity výroby a s tím související vyšší produktivitou s ohledem na zachování, resp. zvýšení kvality výroby a snížení výsledné výrobní ceny. Touto problematikou se, stejně jako mnoho dalších firem, zabývá také společnost B&R Automation.

Cílem této diplomové práce je vytvořit způsob sběru a vyhodnocování dat z výroby pomocí softwaru společnosti B&R Automation, především softwaru APROL. Společnost následně plánuje použít moji realizaci jako jeden z podkladů pro vytvoření firemního standardu pro tuto problematiku. Bohužel má diplomová práce není realizovaná na žádné reálné lince, takže data pro sběr a vyhodnocení efektivity si generuji sám ve mnou vytvořeném simulátoru. Tento simulátor jsem navrhl, aby dostatečně nahradil chod jedné linky, ale také, aby byl jednoduše duplikovatelný pro více linek a bylo možné vytvořit strukturu linek, jak bylo potřeba pro mé testovaní funkčnosti.

Na začátku této práce je probrána potřebná teorie, která napomáhá k lepšímu pochopení práce. V kapitole 1.1 je rozebrána funkčnost *Manufacturing Execution Systems* (MES) s vrstvami důležitými pro mou práci. Jelikož se většina mé práce provádí za pomocí *Distributed Control System* (DCS), je v kapitole 1.2 uvedena jeho definice. Následně v kapitole 1.3 se věnuji teorii sběru a ukládání dat z průmyslových systémů. V kapitole 2 se věnuji KPI, především *Overall Equipment Efficiency* (OEE) a jeho variacím. KPI nejsou jediné způsoby vyhodnocování dat, mezi často používané patří například analýzy nebo určitá pravidla; i ty jsou popsané také v této kapitole. Na závěr 2. kapitoly si uvedeme možné způsoby zobrazení právě vyhodnocených dat.

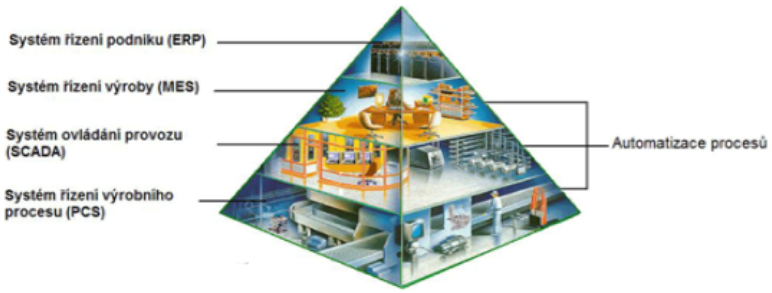
Další kapitoly této práce se věnují použitým programům a samotné realizaci této diplomové práce. V kapitole 3 je popsán všechen použitý *Software* (SW) pro realizaci práce. Kapitola 4 se zaobírá již zmíněným simulátorem linky, určeným ke generování dat. Zpracováním a rozdělením dat v jednotlivých databázích se zabývám v kapitole 5. Způsob zobrazení a vyhodnocení dat se věnuji v kapitole 6.

# 1 Teoretická časť práce

Tato kapitola se zabývá úseky teoretickými částmi, které jsou důležité pro moji diplomovou práci. I když se má práce věnuje rozšíření funkcionalit DCS je nutné pochopit právě provázanost tohoto systému s MES a jeho všemi částmi. V poslední části nastíním, jaké možnosti sběru a uložení dat v praxi máme a jak se od sebe rozlišují.

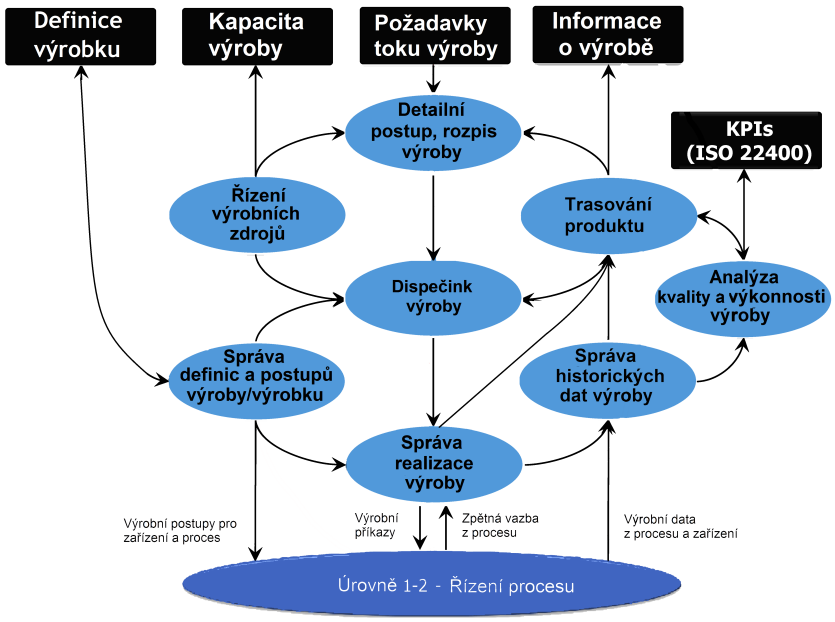
## 1.1 Manufacturing Execution Systems

Z názvu kapitoly vyplývá že se budeme zabývat MES, ale co si má normální člověk pod zkratkou MES představit? Jsou to systémy vyvinuté pro operativní plánování a řízení výroby, jejichž účelem je poskytovat informace pro okamžité řízení a optimalizaci výrobních procesů. Pomáhají výrobním manažerům lépe využívat informace pro spouštění výrobního plánu a jsou vhodné i pro nasazení do výroby, kde je již nasazený podnikový informační systém *Enterprise Resource Planning* (ERP), z důvodu zefektivnění řízení a optimalizace výrobních procesů podniku. Jedná se tedy o přímý integrovaný počítačový systém, který shromažďuje metody a nástroje potřebné ke zdokonalení výroby [1]. Vydaný standard ANSI/ISA-95 situuje MES systém do tzv. „úrovně 3“ mezi ERP na úrovni 4 a *Proces Control Systems* (PCS) na procesních úrovních, jak je tomu znázorněno na obrázku 1.1 [3].




Obr. 1.1: Pyramidové rozložení výrobního podniku [2]

Obecně MES pokrývá široké spektrum vlastností. Tyto výrobní informační systémy implementují, dle požadavků asociace *Manufacturing Enterprise Solutions Association* (MESA), 11 funkčních oblastí. Řada funkcí se ale při realizaci systému v konkrétních podmínkách může navzájem překrývat, a naopak některé funkce nemusí být v konečné verzi zahrnuty vůbec. Výsledné řešení je vždy závislé na potřebách a požadavcích zákazníka [1].




Obr. 1.2: Podrobnější rozepsání MES vrstvy [2]

Před vývojem MES byly nástroje pro řízení výroby náročné na zpracování. Složité ruční papírové zpracovávání informací a jejich částí jako jsou údaje ze statistického řízení procesu, PCS, doručování zpráv, nepřesné kontrolní zprávy a zprávy o zmetkovitosti apod [1]. MES systémy pracují s aktuálními daty v reálném čase, což jim umožňuje pružně reagovat jak na nestandardní stavy ve výrobě, tak i na okamžité požadavky obchodu a přizpůsobovat výrobní proces, aby byl co nejefektivnější.

Kde tato diplomová práce si bere některé z těchto funkcionalit MES a implementuje je o úroveň níž v pyramidovém rozložení podniku popsaném na obrázku 1.1:

### 1.1.1 Sběr a archivace dat

Sběr a archivace dat je základní stavební kámen každého MES systému. Zabezpečuje nepřetržitý sběr dat z výroby v reálném čase, jejich dlouhodobou archivaci a dostupnost pro další zpracování [1] [5].

### 1.1.2 Analýza výkonnosti

Sleduje a počítá klíčové výrobní ukazatele, porovnává výsledky aktuálně dosahované ve výrobě s jejich krátkodobou historií a predikuje odhady ekonomických výstupů [1]. Takto vyhodnocená data se následně mohou použít k regulaci dané výroby a nalezení optimálního chodu, podobně jak je tomu například u řízení dynamických systémů.

### 1.1.3 Řízení kvality

Zajišťuje v reálném čase analýzu dat snímaných z výrobního zařízení s cílem sledovat kvalitu vyráběného produktu a včas identifikovat nežádoucí odchylky. Využívá metody *Statistical Process Control* (SPC) průběžného statistického vyhledávání rozdílů mezi požadovanými „ideálními“ a skutečnými parametry procesu a vyhledávání příčin těchto rozdílů. Do řízení kvality bývají též často zahrnuty offline prováděné analýzy [1].

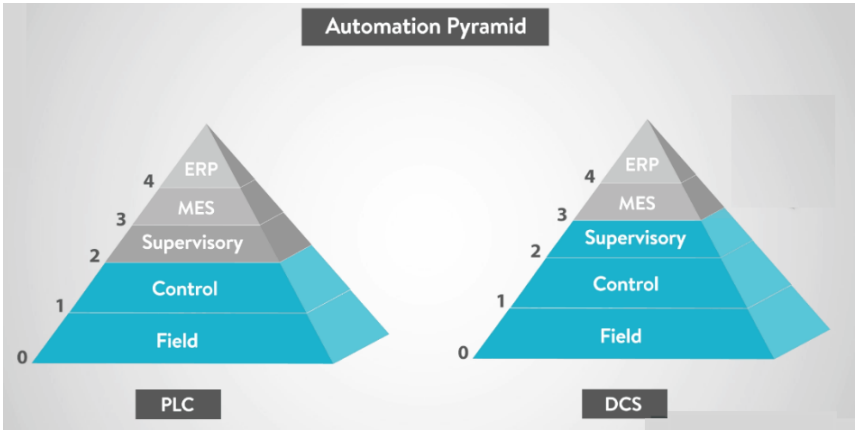
Mezi zbylé funční oblasti MES patří:

- Operativní plánování
- Řízení a přidělování zdrojů
- Dispečerské řízení
- Genealogie a trasování výroby
- Správa dokumentace
- Řízení údržby
- Správa lidských zdrojů
- Řízení laboratoře

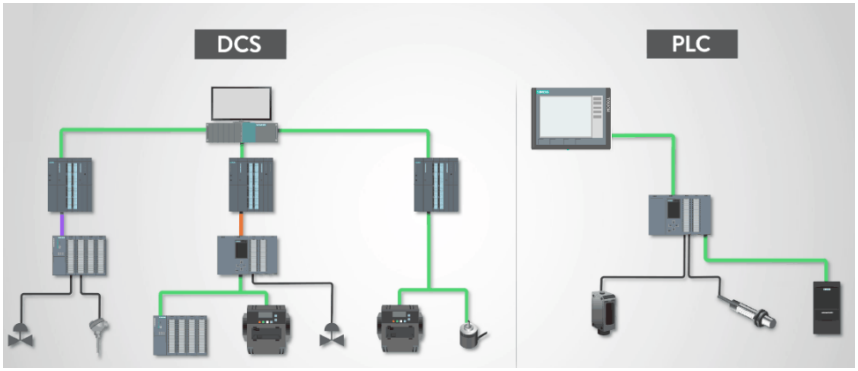
## 1.2 Distributed Control System (DCS)

DCS je počítačový řídicí systém pro proces nebo zařízení obvykle s mnoha vstupními/výstupními body a regulačními smyčkami. DCS se poprvé objevil ve velkých, vysoce hodnotných a kriticky důležitých procesních průmyslových odvětvích. Byl atraktivní, protože výrobce DCS dodal jak místní úroveň řízení, tak centrální dohledové zařízení jako integrovaný balíček, viz. obrázek 1.3, čímž se snížilo riziko integrace návrhu. Systém DCS má stále tendenci být používán ve velkých nepřetržitých procesních provozech, kde je důležitá vysoká spolehlivost a bezpečnost a řídící místnost není geograficky vzdálená [6].

Pod pojmem riziko integrace návrhu u standardních řídících systémů, že když je přidán vstupně/výstupní bod včetně řídicí logiky, je nutné přidat nebo změnit element v *Human–Machine Interface* (HMI), v databázích historie a alarmů, a to z různých vývojových programovacích prostředí. Komplikace přinášejí také častější změny nastavení alarmů např. v řídicí smyčce, protože neexistuje automatické spojení mezi *Programmable Logic Controller* (PLC) a HMI. Systém správy alarmů a aplikační program HMI potom musí být po každé změně v PLC znovu aktualizovány. Systémy DCS umožňují jednoduše implementovat celý systém řízení procesní výroby integrováním všech databází a funkcí do jednotného systému, který je vytvářen, konfigurován a řízen ve stejném prostředí [6] [7].




Obr. 1.3: Pyramidové rozložení PLC a DCS [6]




Obr. 1.4: Rozdíl mezi PLC a DCS řízením [6]

Tradiční DCS je především určen pouze k účelu řízení procesu. Aby ale společnosti získali výhodu v současné konkurenční ekonomice, musí začít využívat nových technologii a inovací. Proto je na čase přehodnotit, co se očekává od DCS a tímto se zabývá tato diplomová práce.

## 1.3 Skladování dat

Pojem datové skladování v originále "data warhousing"je celkové řešení zahrnující návrh a implementaci nástrojů, procesů a služeb k zajištění doručení přesných srozumitelných, kompletních a včasných dat potřebných k tvorbě rozhodnutí [8].

Data běžných informační systémů bez datového skladu však data neuchovávají. Jakmile se stávají historickými, jsou maximálně přehrávána do archivu, kde je již jejich využití minimální. Toto se děje zejména z důvodu výkonnosti systému a také proto, že běžný uživatel takového provozního systému tato historická data s velkou pravděpodobností nepotřebuje. Pro rozhodování a analýzy jsou však tato data

nezbytná a s tímto přístupem se stávají nedostupnými. Pokud jsou tato historická data přece jen k dispozici, pak pravděpodobně provozní informační systém neposkytuje dostatečné techniky a nástroje pro zpracování těchto dat. Běžný provozní systém disponuje pouze sadou předem připravených sestav, které se spíše orientují na jednotlivé transakce než na globální pohled [9].

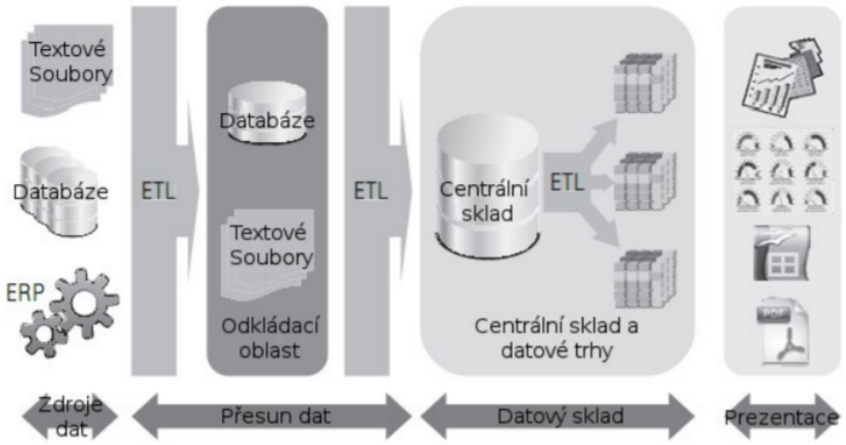
Důvody ke zbudování datového skladu z pohledu jeho uživatele jsou nejčastěji tyto:

- **Všechna data na jednom místě** – Není nutné procházet mnoho zdrojů dat k získání informací, není ani nutné kombinovat množství dat ručně, protože všechna potřebná data již jsou integrovaná.
- **Aktuální informace** – Data v datovém skladu jsou automaticky aktualizována, což znamená, že uživatel má vždy poslední informace.
- **Rychlý přístup** – Datový sklad je optimalizován pro rychlé čtení dat.
- **Bez omezení na množství dat** – Tabulkové procesory jsou schopné pojmout omezené množství dat a mnohdy musí být rozděleny na několik částí. Datový sklad pojme téměř neomezené množství záznamů v závislosti na použitém databázovém systému.
- **Dostupná historie** – Datové sklady neobsahuje pouze aktuální informace, ale informace získané během let. To znamená, že datový sklad je ideální k zjišťování různých trendů a porovnávání historických dat. Data z datového skladu nejsou téměř nikdy mazána.
- **Jednotné definice** – Všichni používají stejné definice, což vylučuje diskuse a problémy v pohledu na pochopení smyslu použitého výrazu.
- **Standardizovaná data** – Všechna data v datovém skladu jsou vždy daného typu [8].

Jelikož je proces datového skladování složen ze složitého komplexu komponent, je použití vhodné architektury jeden z předpokladů k jeho úspěšnému vybudování. Na obrázku 1.5 je ukázána logická strukturu a vztah jednotlivých komponent datového skladování [8] [9].

Část zdroje dat obrázku 1.5 znázorňuje několik různých zdrojů pro plnění *Data Storage* (DS). V části přesun dat dochází k extrakci, transformaci a nahrání dat ze zdrojů do DS. Často tato část obsahuje odkládací oblast, sloužící k dočasnému uskladnění, provedení počátečního vyčištění a transformaci dat před samotným přesunem do datového skladu. Oblast Datový sklad obsahuje centrální datový sklad a případné datové trhy. Část prezentace pak zahrnuje nástroje pro práci se sestavami, prezentacemi obsahu datového skladu, což představuje prezentační vrstvu datového skladování [8] [9].

Zdrojů dat pro datový sklad může být velké množství. Každý ze zdrojů může být na různých médiích v jiném formátu a v jiném kódování. Je tedy nutné zvolit




Obr. 1.5: Znázornění procesu datového skladování [8]

vhodné zdroje dat, ze kterých pak pomocí *Extract, Transform and Load* (ETL) procesu data získáme, následně je transformujeme a přesuneme do datového skladu [8] [10]. Zdrojová data můžeme rozdělit takto:

- Produkční data
- Interní data
- Archivní data
- Externí data

Striktní pravidla, jak by měl proces datového skladování přesně vypadat, neexistují, ale existuje několik architektur, dle kterých můžeme tento proces navrhnout. Jedno z možných členění je podle použitých komponent procesu datového skladování nebo podle struktury datového skladu:

- podle použitých komponent
  - **Jednovrstvá architektura** - Jednovrstvá architektura neobsahuje fyzický datový sklad, dotazy jsou prováděny přímo nad operačními daty.
  - **Dvouvrstvá architektura** - Již má fyzický datový sklad, ten je ale plněný přímo procesem ETL.
  - **Třívrstvá architektura** - Přidáním fyzické odkládací oblasti do dvouvrstvé architektury získáme architekturu třívrstvou.
- podle struktury datového skladu
  - **Independent Data Marts** - Každý datový trh je zbudován a plněn nezávisle na ostatních, jednotlivé trhy neví nic o ostatních datových trzích, takže neexistují společná metadata.
  - **Data Mart Bus** - Je přesným opakem architektury s oddělenými datovými trhy, využívá sdílených dimenzí a jednotlivé datové trhy obsahují atomická a sumarizovaná data.
  - **Hub and Spoke** - Obsahuje centrální datový sklad a z něj případně

plněné datové trhy. Normalizovaný relační sklad obsahuje atomická data. Volitelné datové trhy pak data atomická a sumarizovaná.

– **Centralized Data Warehouse** - Je v podstatě stejný jako Hub and Spoke, ale neobsahuje přidané datové trhy. Koncový uživatel je přímo odkazován na centrální datový sklad. Tento přístup je velice podobný architektuře Data Mart Bus, hlavní rozdíl je v normalizaci tabulek použitých v normalizovaném relačním skladu [8] [9].

# 2 Key Performance Indicators

V kapitole si řekneme co je KPI, jaké KPI a analýzy se v praxi nejčastěji používají. V závěru kapitoly si ukážeme i způsoby zobrazení dat v praxi a rozdíly mezi nimi.

KPI je velice rozšířený pojem a je rozdílně chápán každým odvětvím průmyslu. Může se tak stát, že i po čase kde vedete rozhovor s člověkem o tom samém tématu, každý ho chápete trochu jiným způsobem, proto v této práci pod pojmem KPI jsou myšlené vypočítané indexy z nasbíraných dat.

Následně je nutné si uvědomit, že k zavedení KPI není znovu žádná jednotná šablona, pokaždé toto zavedení začíná kompletní analýzou, diskuzí a až následně možným řešením. Bohužel toto řešení není vůbec levné a někdy je kvůli tomuto nutná kompletní inovace výrobní linky nebo linek, což je jeden z hlavních důvodů, proč je v českém průmyslu nástup KPI velice pozvolný. Základními vlastnostmi zavedení KPI do výroby jsou:

- získání přehledu o aktuálním stavu a efektivitě strojů,
- odhalení prostojů a ostatních činností, které stroj omezují,
- sledování nákladů na stroj,
- zlepšení nebo udržení stávající kvality výroby [11].

V praxi se můžeme setkat z mnoha KPI, proto byla provedena snaha většinu těchto KPI sepsat a standardizovat, tímto vznikl dokument 34 KPI, viz. tabulka 2.1, u kterých se odborníci shodli, že jsou nejčastěji využívány v průmyslu. Norma zabývající se tímto problémem můžete najít pod *International Organization for Standardization* (ISO) 22400 přesněji ISO 22400-2.

## 2.1 Rešerše v praxi využívaných indexů

Z již zmíněného standardizovaného dokumentu jsem vybral v praxi nejpoužívanější indexy. Tento výběr jsem si nadále ověřil u společností zabývající se vyhodnocováním výkonosti linek a zavádění MES do výroby, jako jsou Act-In, Unicorn a další.

### 2.1.1 Availability

Availability je to podíl plánovaného času výroby a provozního času výroby. Od provozního času jsou odečteny všechny prostoje, které ovlivnily zastavení plánované produkce. Zastavení je počítáno od několika minut tak, aby bylo zaznamenatelné. Prostoje, jako například poruchy zařízení, zpoždění dodávky materiálu a přestavba stroje, jsou zařazovány do faktoru Availability. Zbývající dostupný čas bez prostojů je nazýván časem výroby. Výpočet Availability je uveden ve vztahu [13] [14].

Tab. 2.1: KPI norma ISO 22400 [12]

| | | |
|---|---|---|
| Worker Efficiency | Production process ratio | Finished goods ratio |
| Allocation Ratio | Actual to planned scrap ratio | Integrated goods ratio |
| Throughput rate | First pass yield | Production loss ratio |
| Allocation efficiency | Scrap ratio | Storage and transportation loss ratio |
| Utilization efficiency | Rework ratio | Other loss ratio |
| *Overall equipment effectiveness index | Fall off ratio | Equipment load ratio |
| Net equipment effectiveness index | Machine capability index | *Mean operating time between failures |
| *Availability | Critical machine capability index | Mean time to failure |
| *Effectiveness | Process capability index | *Mean time to restoration |
| *Quality Ratio | Critical process capability index | Corrective maintenance ratio |
| Setup Rate | Comprehensive energy consumption | - |
| Technical efficiency | Inventory turns | - |

$$Availability = \frac{provozní\ čas\ výroby}{plánovaný\ čas\ výroby}\ [-] \tag{2.1}$$

### 2.1.2 Performance

Performance je dán podílem ideálního času stroje a celkového počtu vyrobených kusů během provozního času výroby. Do faktoru výkonu jsou započítány všechny ztráty rychlosti zařízení, které zabránily procesu, aby proběhl maximální možnou rychlostí bez přerušení. Ztráty rychlosti zařízení mohou být způsobeny například opotřebením stroje, použitím nestandardních materiálů, zaseknutými výrobky ve stroji. Odečtením ztrát rychlosti od provozního času výroby je dosaženo čistého provozního času výroby. Základní obecné vztahy pro výpočet výkonu jsou uvedeny níž. V praxi je tento vztah upraven dle požadavků a podmínek podniku [13] [14] [15].

$$Performance = \frac{ideální\ cyklus\ stroje * celkový\ počet\ kusů}{provozní\ čas\ výroby}\ [-] \tag{2.2}$$

$$Performance = \frac{ideální\ čas\ cyklu\ stroje}{\frac{provozní\ čas\ výroby}{celkový\ počet\ kusů}}\ [-] \tag{2.3}$$

$$Performance = \frac{\frac{celkový\ počet\ kusů}{provozní\ čas\ výroby}}{ideální\ výrobní\ čas\ výrobku}\ [-] \tag{2.4}$$

### 2.1.3 Quality

Při výpočtu faktoru Quality jsou dány do poměru kvalitní výrobky vyrobené napoprvé ku celkovému počtu vyrobených kusů. Nekvalitní výrobky jsou ty, které neodpovídají standardům kvality dle vztahu [13] [14].

$$Quality = \frac{celkový\ počet\ kvalitních\ kusů}{celkový\ počet\ kusů}\ [-] \tag{2.5}$$

### 2.1.4 Overall Equipment Effectiveness

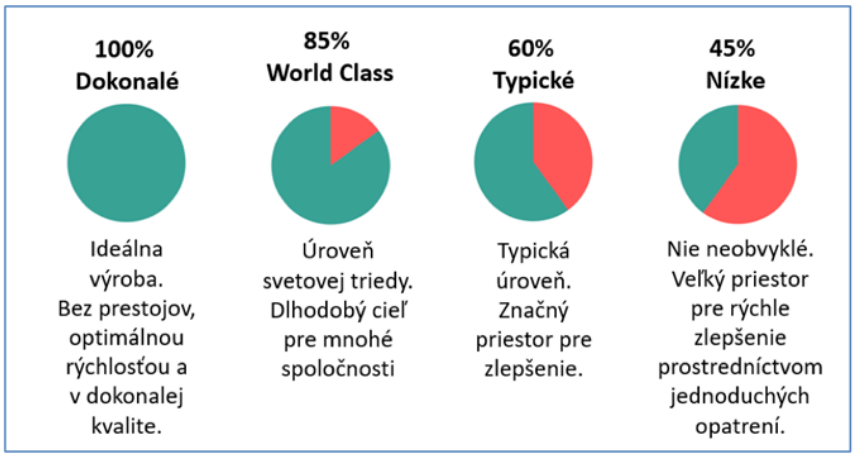
V případě OEE se jedná o jeden z nejznámějších a nejpoužívanějších KPI. Kdykoliv se v praxi narazí na pojem KPI, většinu lidí napadne právě OEE. Je to dáno také tím, že pod tímto indikátorem se skrývá hned několik dílčích indikátorů výroby a její efektivnosti. Ukazatel OEE vytvořil v 60. letech Seiichi Nakajima ze společnosti Nippon Denso. Na konci 80. let se tato metodika dostává do povědomí díky rozšíření *Total Productive Maintenance* (TPM), kde postupně metodiku přijala i ostatní odvětví průmyslové výroby [13].

$$OEE = Availability * Performence * Quality * 100\ [\%] \tag{2.6}$$

Takto získaná hodnota nám následně definuje efektivnost daného stroje nebo výroby, ale problém nastává když danému podniku vychází OEE rovno 120 %, což z reálného hlediska není možné [16]. Zobrazení průměrných a očekávaných výsledků můžete vidět na obrázku 2.1.

### 2.1.5 Total Equipment Effectiveness Performance

*Total Equipment Effectiveness Performance* (TEEP) je jedním z odvozených ukazatelů z OEE a stále v praxi velice používaným ukazatelem ve výrobě, který posuzuje efektivnost vzhledem ke kalendářnímu času (tedy 24 hodin denně, 7 dní v týdnu, 365 dní v roce), oproti OEE, které je posuzováno v rámci plánovaného času (plánované směně). Pokud vezmeme v úvahu případ, že by chod zařízení byl naplánován na 24




Obr. 2.1: Typické hodnoty OEE v praxi [17]

hodin denně, 7 dnů v týdnu a 365 dní v roce, pak by ukazatel TEEP odpovídal ukazateli OEE [13].

TEEP je využitelný pro analýzu podniku a důležitý pro maximalizaci produktivity před investicí do nového zařízení. Vhodnost ukazatele TEEP spočívá v posouzení celkového využití majetku a výrobních nákladů. Ukazatel TEEP je klíčovou informací pro obchodní oddělení, které se na jeho základě je schopno rozhodnout, zda přijmout další zakázky či nikoliv [14] [15].

I zde máme různé způsoby výpočtů, kde každý z daných vzorců ukazuje různé varianty pohledu na danou problematiku.

$$TEEP = \frac{provozní\ doba\ podniku - nevýrobní\ směny}{provozní\ doba\ podniku * počet\ kalendářních\ dní}\ [\%] \tag{2.7}$$

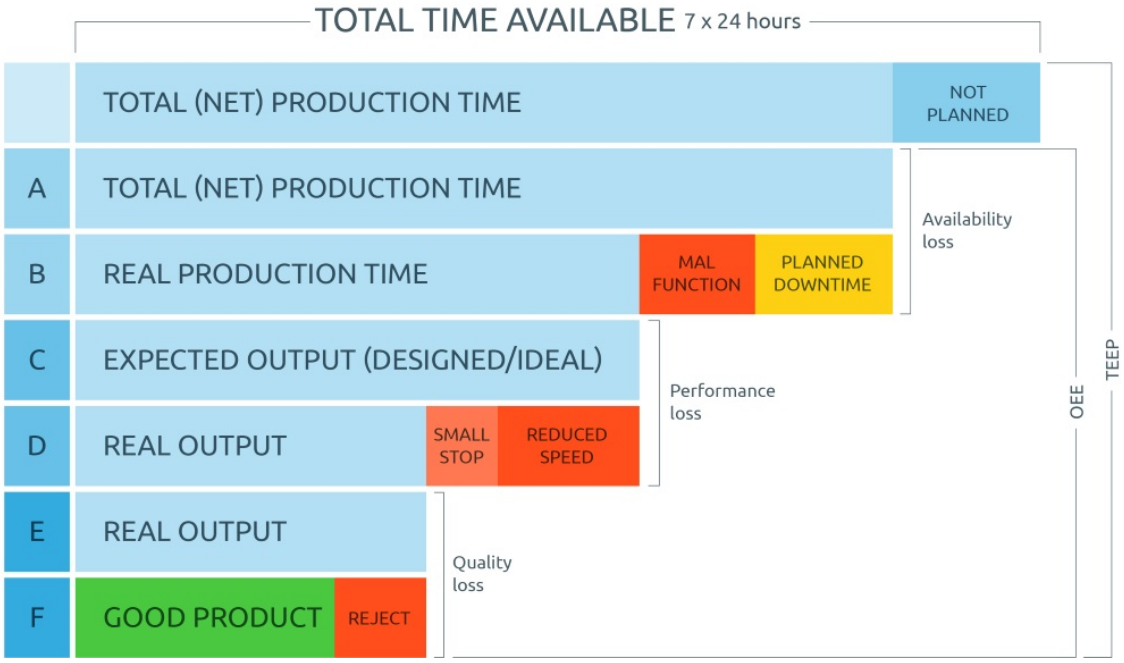
$$TEEP = \frac{užitečný\ čas\ zařízení}{kalendářní\ čas}\ [\%] \tag{2.8}$$

$$TEEP = Utilization * OEE = \frac{disponibilní\ čas}{kalendářní\ čas} * OEE\ [\%] \tag{2.9}$$

### 2.1.6 Production Equipment Efficiency

Další ukazatel, který je odvozen z ukazatele OEE je *Production Equipment Efficiency* (PEE). Jeho hlavní přidanou hodnotou je zavedení tzv. vah do výpočtu OEE, kde jsou naopak brány všechny dílčí výpočty se stejnou vahou [13] [18].

Váhy ukazatelů můžeme použít například, pokud při hodnocení personálu preferujeme některou ze složek OEE. Pokud mají různé skupiny pracovníků odlišný vliv na výsledky jednotlivých ukazatelů (operátoři na kvalitu, seřizovači na výkon,




Obr. 2.2: Grafické znázornění odečtení hodnot pro výpočet TEEP

vedoucí výroby na využití stojů apod.), můžeme pro hodnocení těchto skupin použít výpočty s odlišnými vahami [19]. Způsob výpočtu PEE se dále liší dle typu výroby:

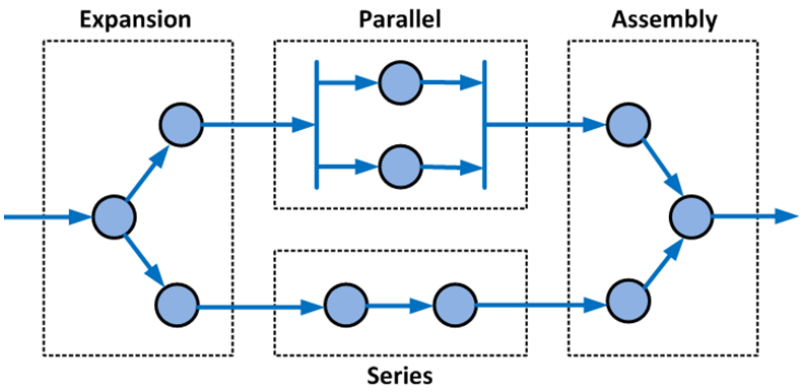
- diskrétní výroba,
- seriová výroba.

### 2.1.7 Overall Factory Effectiveness

Dalších ukazatel odvozených z OEE, který vám popíši je *Overall Factory Effectiveness* (OFE). Tento ukazatel byl odvozen pro vyjádření efektivnosti celého podniku, jako výrobního celku, jelikož, není možné na celopodnikové úrovni použít klasické OEE. Zatímco OEE se zaměřuje na efektivnost jednotlivých zařízení, OFE vyhodnocuje všechna zařízení dohromady. Vyhodnocujeme tedy všechna zařízení dohromady, i když probíhá více kroků na více výrobních zařízeních. Do výpočtů zahrnujeme vztahy a interakce mezi zařízeními a procesy, kde následně uvažujeme, že výrobní prostředí lze rozdělit do čtyř základních skupin [13] [18]:

- **Series** - sériová, v řadě
- **Parallel** - paralelní, souběžná
- **Assembly** - spojení, montáž
- **Expansion** - rozdělení, expanze

Pomocí těchto 4 skupin subsystémů, obrázek 2.3 lze namodelovat celý výrobní provoz. Efektivitu jednotlivých subsystémů pak zkoumají metodiky jako *Overall Throughput Effectiveness* (OTE) nebo *Cycle Time Effectiveness* (CTE) [13] [18].




Obr. 2.3: Názorná ukázka možného rozložení výrobního podniku [19]

Pokud se ve výrobě nevyskytují složité vztahy a interakce mezi různými výrobními zařízeními a procesy, můžeme se spokojit s některou elementární agregační metodou. Speciálním případem elementární agregační metody je pak tzv. Sdružené zařízení, které představuje skupinu výrobních zařízení, která posuzujeme jako jedno výrobní zařízení realizující paralelní výrobu [19].

### 2.1.8 Overall Throughput Effectiveness

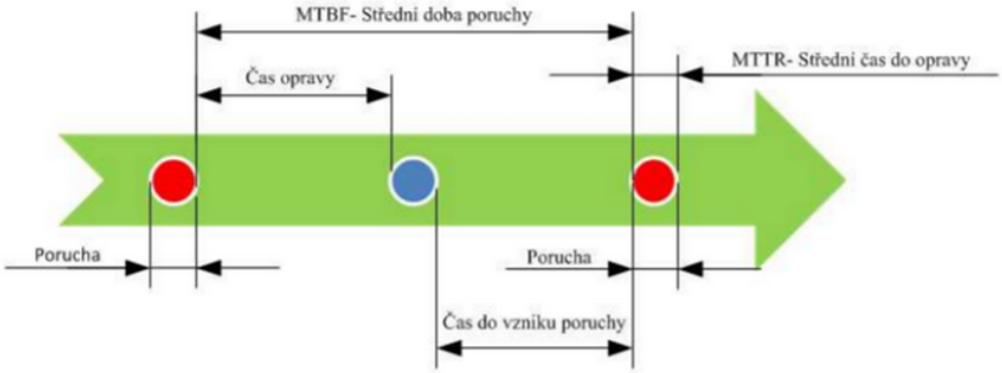
OTE vyjadřuje poměr mezi skutečným výstupem a teoreticky možným výstupem. Metodika je úzce spojena se simulačními technikami pro zvyšování produktivity výroby. Podobně jako OFE, tato metrika také uvažuje o různých konfiguracích výrobních systémů. Cíl OTE je měření výkonu založené na továrně a provádění diagnostických akcí (na úrovni výroby), jako je zjišťování vytížených míst a skrytí prostředků [13] [20].

### 2.1.9 Cycle Time Effectiveness

CTE je pak poměrem mezi teoretickou délkou cyklu a skutečnou délkou cyklu, kde metodika je také úzce spojena se simulačními technikami pro zvyšování produktivity výroby [19].

### 2.1.10 Mean Time To Restoration

V praxi se velmi často můžeme setkat s ukazateli středních dob, jeden z těchto ukazatelů je *Mean Time To Restoration* (MTTR), kde je často restoration zaměňováno za repair. MTTR je možné stanovit poměrem celkové doby prostojů za dobu provozuschopnosti ku počtu poruch za dané období. Je to čas, který je vyžadován na




Obr. 2.4: Ukázka ukazatelů MTTR a MTBF [21]

opravu zařízení [21]. Tento ukazatel je velice důležitý v případě hodnocení servisních techniků a často je kladen, aby daný čas byl co nejmenší [22].

$$MTTR = \frac{celková\ doba\ neplánovaných\ prostojů}{počet\ poruch}\ [s] \tag{2.10}$$

### 2.1.11 Mean Time Between Failure

Dalším ukazatelem vycházející ze střední doby je *Mean Time Between Failure* (MTBF), ten je dán celkovou dobou provozuschopnosti podělenou počtem poruch na daném zařízení. Vyjadřuje množství času od konce jedné poruchy do vzniku další [21]. Tento ukazatel, na rozdíl od předchozího, poukazuje na spolehlivost stroje, jak často se stroj dostává právě do tzv. neplánovaných prostojů, které nám následně ovlivňují jeho efektivitu a tím OEE [23]. Jednoduše je to střední doba mezi dvěma poruchami.

$$MTBF = \frac{celková\ doba\ provozu\ linky}{počet\ poruch\ za\ provozu\ linky}\ [min] \tag{2.11}$$

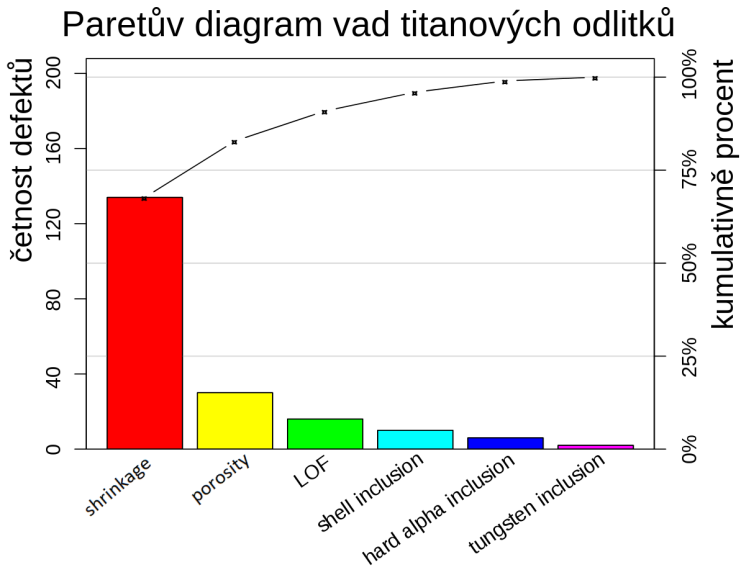
## 2.2 Další metody vyhodnocebí

Výše zmíněné indexy nejsou jediný způsob jak vyhodnocovat efektivnost nebo kvalitu výroby, existuje mnoho dalších řešení, i když indexy jsou nejběžnějším a nezjednoduším řešením. Jedním z dalším často zmiňovanou metodou je například OLAP kostka, která přizpůsobí uložení dat v databázi pro lepší přehlednost uživatele. Mezi další metodiky hlídající výkonost a efektivnost patří metoda Six Sigma, Lean Production nebo *Total quality management* (TQM). Když pomineme metody dostáváme se k jednodušším pravidlům, jako například Paretovo pravidlo, které využívám ve své diplomové práci a je popsané níže.

### 2.2.1 Paretova analýza

Paretova nebo také ABC analýza, či pravidlo 80/20 je velice jednoduchým, ale přesto efektivním nástrojem, který umožňuje společnostem se matematicky exaktně soustředit na to, co je pro ně skutečně důležité. Použít lze přitom na zákazníky, vlastní výrobky a služby či třeba na skladové zásoby. Jeho autorem je Joseph Moses Juran, rumunský rodák, který se po emigraci do USA věnoval problematice řízení kvality a který se v roce 1941 náhodou dostal k výsledkům práce italského mikro- a socioekonoma Vilfreda Frederica Damasa Pareta. Joseph M. Juran, tohle pravidlo aplikoval na oblast řízení kvality, kde zjistil, že například zhruba 80 % odstávek výroby je způsobeno 20 % zařízení továrny. Toto pozorování následně zobecnil na konstatování, že za 80 % problémů může 20 % příčin, čemuž se od roku 1941 říká Paretovo pravidlo [24].

Pro zobrazení Paretova pravidla, potažmo výsledků Paretovy analýzy se nezřídka využívá invertovaná Lorenzova křivka. Lorenzova křivka popisuje v případě zobrazeném na obrázku 2.5 nerovnoměrnost rozdělení vad na titanových odlitků.




Obr. 2.5: Paretův diagram [24]

V každé výrobě jsou nějaká klíčová místa, která vyžadují zvýšenou pozornost. Rozdělení této pozornosti je cílem Paretovy analýzy, resp. Lorencovy křivky. Může se jednat o vlastní výrobní segment, nebo o výrobní chyby vedoucí k nedokonalostem výrobků. Tam nám Lorencova křivka, jako výsledek analýzy ukáže, který segment výroby, nebo která výrobní operace je nejpodstatnější a tím vyžaduje největší pozornost.

## 2.3 Zobrazení vyhodnocených dat

Kapitola se zabývá různými druhy zobrazení vyhodnocených dat, prvně se koukneme na nejběžnější způsob zobrazení, a to pomocí dashboardu. Řekneme si definici a jak jej můžeme rozdělit a následně rozlišovat. Něco si řekneme i o druhém způsobu zobrazení, a to o scorecards, který v praxi není zas tak častý.

### 2.3.1 Dashboard

Tento způsob zobrazení je v praxi právě nejčastějším a nejznámějším způsobem zobrazování KPI v praxi. Dashboard je v překladu do češtiny přeložen jako nástěnka. V praxi se lidé tímto překladem natolik inspirovali, že často zobrazují a vyhodnocovali KPI tímto způsobem tzv. firemních nástěnkou, jak je tomu vyobrazeno na obrázku 2.6. S tímto způsobem zobrazení se stále můžeme v praxi setkat, ale jedná se jen o informativní gesto jelikož i při veliké snaze tyto data nejsou zobrazena v real-time.

Obr. 2.6: Starší zobrazení Dashboardu

Příkladem dashboardu je kontrolní panel automobilu. Najdeme na něm různé budíky, ukazující aktuální rychlost, otáčky motoru, obsah palivové nádrže a další ukazatele, které nám zobrazují informace o stavu automobilu. Tyto informace musíme vyhodnocovat téměř neustále, ale zároveň se musíme věnovat řízení, a proto musí být informace lehce pochopitelné na první pohled. Pomocí tohoto jednoduchého přirovnání je dashboard vnímán jako uspořádání několika graficky znázorněných informací, které lze jednoduše vyhodnotit. Nicméně takové vnímání dashboardu není úplně dostačující pro jeho definici [25].

Odborná definice zase říká, že dashboard je vizuální zobrazení těch nejdůležitějších informací, potřebných k dosažení jednoho nebo více cílů, které jsou uspořádány na jediné obrazovce, aby se informace daly monitorovat hned na první pohled [26]

a následně správně navržený dashboard by měl splňovat všechny požadavky, které popisuje jeho definice.

Data jsou prezentována grafickým způsobem, protože tak mohou nejefektivněji a nejsnadněji komunikovat informace. Lidé totiž dokážou lépe zpracovat vizuální informace, než samotný text. Zpracovávají je spolu s kontextem paralelně a tím je dokážou pochopit téměř okamžitě. Na dashboard patří pouze informace, které uživatel potřebuje vědět ke splnění svých cílů. Prostoru pro zobrazení dat je na dashboardu velmi málo a musí se s ním pracovat efektivně. Každá zobrazená informace musí mít dobrý důvod tam být, jinak by se měla odstranit. Nepotřebné informace navíc berou pozornost od těch důležitých. Informace by měly být zobrazené pouze na jediné obrazovce, která je nejlépe kompletně zobrazena uživateli. V ideálním případě by uživatel nemusel scrollovat ani překlikávat, aby dostal kompletní obraz o současném stavu [26].

Dashboardy můžeme rozdělit podle způsobu, jakým ho budou jeho uživatelé používat a jaké problémy chtějí pomocí nich vyřešit. Podle toho potom můžeme dashboardy rozdělit do tří kategorií; na strategické, analytické a operační [27]. Výsledný dashboard může být i kombinací těchto přístupů. Důležité je, aby dashboard zobrazoval informace jasně a stručně.

### Strategické dashboardy

Hlavním cílem strategického dashboardu je poskytovat uživatelům potřebné informace k porozumění současného stavu organizace. Informace by měly zobrazovat, jestli je organizace v dobrém nebo špatném stavu. Dále by měly umožňovat identifikaci potenciálních příležitosti ke zlepšení a rozšíření [28]. Strategické dashboardy neposkytují všechny detailní informace potřebné k rozhodnutí, ale identifikuje příležitosti pro hlubší analýzu. Tento typ dashboardu uživatelé používají typicky jednou za měsíc nebo týden a primárně ho využívá vysoký managment při strategickém plánování.

### Analytické dashboardy

Analytické dashboardy mají za cíl poskytovat uživateli data, která potřebuje k pochopení trendů a příčin některých výsledků v organizaci. Uživatele navádí k podrobnějšímu prozkoumání určitých věcí. Analytické dashboardy často obsahují více informací než strategické a operativní. Informace jsou zde zobrazeny ve větším detailu a porovnány v širším kontextu. Tento typ dashboardu je většinou interaktivní. Uživatel si může vyfiltrovat data, o která se zajímá a která potom může důkladněji prozkoumat [27] [28]. Tento typ dashboardů především vedoucí výroby - ten je zodpovědný a musí mít neustále informace o výrobě, údržbáři a seřizovači pro hladký

chod výroby, tedy odstranění možných příčin ještě před jejich vznikem.

### Operační dashboardy

Cílem operačních dashboardů je monitorovat a podporovat každodenní výrobní aktivity určité oblasti organizace v reálném čase. Uživatele upozorňují na odchylky od normálních hodnot a poskytují jim specifické informace, které uživatelé potřebují znát k uvedení stavu organizace do normálu. Ve srovnání se strategickými a analytickými dashboardy obsahují méně informací, které jsou aktualizovány mnohem častěji, někdy i v reálném čase [29]. Operativní dashboardy si můžeme prohlédnout na obrázku 2.7, kde je jasně patrná jejich jednoduchost a na první pohled přehlednost. Tento typ dashboardů využívají především vedoucí výroby a operátoři.


Obr. 2.7: Příklady Dashboardů v praxi

## 2.3.2 Scorecards

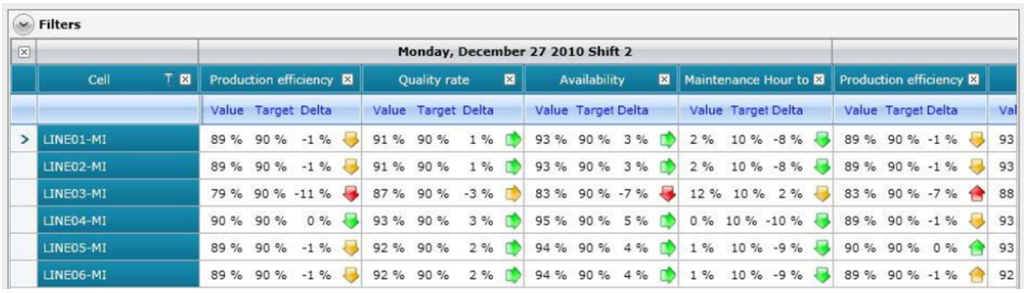
Scorecards je v současné době jednou z méně používanou metodou zobrazování KPI v praxi na operátorské úrovni. Je to hlavně způsobeno tím že Scorecards nebo taky *Balanced Scorecard* (BSC) je metoda v managementu, která vytváří vazbu mezi strategií a operativními činnostmi s důrazem na měření výkonu [30]. U scorecard je vždy stanoven cíl, který se má dosáhnout, skóre nám ukazuje vztah s aktuálním stavem a stavem cílovým. Tím se liší od dashboardu.

Definice říká, že Balanced Scorecard je metoda pro strategické hodnocení výkonnosti firmy finanční i nefinanční. Využívá se pro tvorbu, zavedení i aktualizaci strategie, vyvážení požadavků zájmových skupin, pro rychlý a jasný přehled o podnikání. K hodnocení se využívají výkonnostní ukazatele KPI. Jedná se vlastně o měřitelný způsob dosahování strategie při dodržení principu příčina-následek [30].

Pokud bychom se drželi analogie, kterou jsem nastínili u dashboardů můžeme scorecards přirovnat k GPS, která poskytuje podnikům informace o aktuální pozici

a době, kterou je třeba, aby se dosáhlo cíle. Ve většině případů, když jedete z města A do města B, používá pouze GPS, ale pokud se něco stane s vozem, první místo, na které se podíváte, je palubní deska (dashboard) [2].

Scorecards je spíše tabulkové zobrazení, kde je nutné vždy zvolit požadovaný cíl neboli cílovou hodnotu které chceme dosáhnout. Následně tuto cílovou hodnotu porovnáváme s aktuální a vyhodnocujeme. Příklad scorecard můžete vidět na obrázku 2.8.




Obr. 2.8: Příklad Scorecard

V praxi se scorecards využívají společně s metodou *Management by Objectives* (MBO) v češtině řízení podle cílů, mohou takto navrhnuté scorecards sloužit jako motivující prvek právě operátorům pro splnění dílčí cílů dané výroby. Tato metoda je založena na stanovení a vzájemném odsouhlasení cílů a vyhodnocování úspěšnosti jejich dosahování.

# 3 Použitý software

Kapitola 2 poskytuje popis všech programů se kterými jsem při plnění své práce pracoval. Většina těchto programů je od společnosti B&R automation a proto mi bylo umožněno projít komerčními firemními školeními na tyto programy. Prvně si něco řekneme o Automation Studiu, ve kterém jsem vytvořil simulátor výrobní linky na generování dat. Následně se přesuneme v programu APROL, který slouží pro sběr a předzpracování dat. Dále se dostaneme k programu MariaDB, který naopak uchovává a znovu zpracovává data. A jako poslední se dostaneme k SW od společnosti Jaspersofr, který slouží k zobrazení zpracovaných dat.

## 3.1 Automation Studio 4.6

Automation Studio je konfigurační prostředí používané pro komponenty společnosti B&R automation. To zahrnuje PLC, motion control components, safety moduly a aplikace HMI. Automation Studio nabízí ideální prostředí pro vytváření různých variant projektu a jejich úhlednou správu. Projekty mohou být jasně strukturovány a tímto způsobem usnadňovat týmovou práci [31].


Obr. 3.1: Funkce Automation Studia [31]

Uživatelé si mohou vybrat ze široké škály programovacích jazyků, diagnostických nástrojů a editorů, kterým mohou pomoci v každé fázi projektu. Rozsáhlé možnosti simulace usnadňují konfiguraci a testování aplikací nezávisle na hardwaru [31].

Modulární architektura a struktura programovacího prostředí podporuje každodenní pracovní postup programování. Program má integrované a standardizované jazyky IEC 61131-3, perfektní integrací ANSI C. Automation Studio nabízí další

integrované funkce, jako jsou Smart Edit, C++ a záložky editorů, usnadňující práci a současně zvyšuje produktivitu [32].

## 3.2 Automation Runtime

Automation Runtime je deterministický operační systém v reálném čase, který slouží jako softwarová platforma pro celou škálu produktů B&R automation. Automation Runtime poskytuje služby pro volnou konfiguraci a troubleshooting cílového systému tak i pro provádění programů [33].

Automation Runtime je plně integrován do cílových systémů B&R. Poskytuje uživateli deterministický, na hardwaru nezávislý, multitaskingový nástroj pro vytváření aplikací, umožňuje konfiguraci 8 různých tříd a časů cyklů. Spravuje hardwarové a softwarové prostředky a nabízí kompletní diagnostiku. Může běžet na mnoha různých hardwarových platformách. Jako PLC typu X20, Power Panels a Automation PC a hardwarové platformy založené na PC [33].

## 3.3 APROL

APROL je PCS, pod který spadá několik PLC, obvykle značky B&R, k nimž jsou připojena různé druhy počítačových systémů (řídicí počítače). Tyto počítače provádějí širokou škálu úkolů. Mohou například sledovat všechna data, která jsou důležitá pro provádění automatizovaných úloh (proces) nebo pomocí databází je sbírat. Ostatní počítače poskytují uživateli grafická uživatelská rozhraní, která lze použít ke sledování běhu procesu a analýzu všechna shromážděných dat (operátorská stanice) [34].

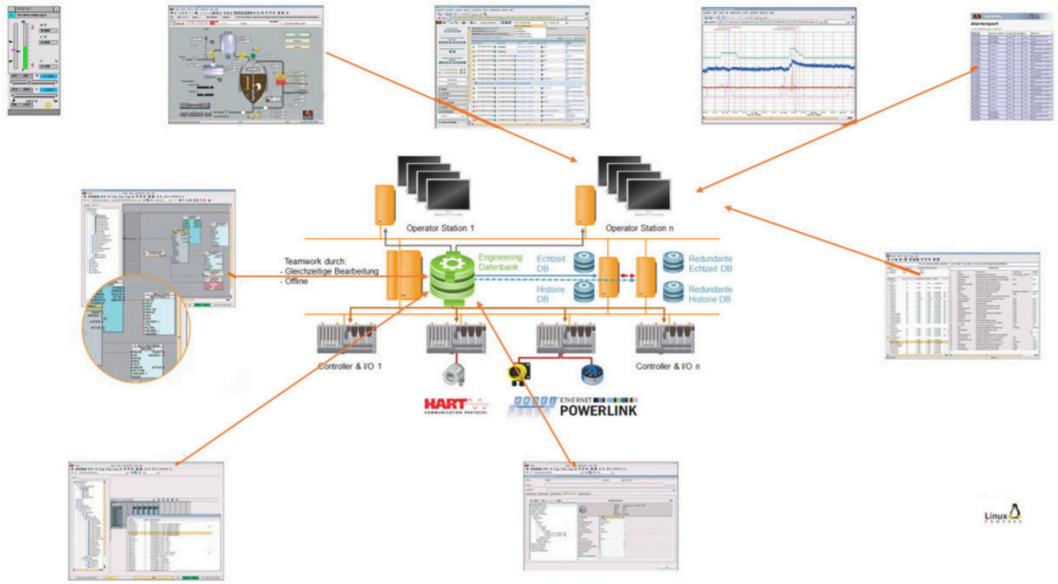
APROL je založený na operačním systému Linux. Všechny programy APROL si vyměňují svá data pomocí komunikačního protokolu TCP/IP. Programy lze rozdělit na programy uživatelského rozhraní a programy na pozadí [34].

Engineering server se používá k úplné konfiguraci systému řízení procesů až na úroveň řízení. Kompletní distribuce systému je řešena v Engineering serveru (stahování úkolů do PLC, stahování Operator system, stahování Runtime systému (procesy na pozadí)) [34].

Na základě údajů Engineering systému tvoří Runtime systém jádro process control systemu. Tento systém monitoruje, shromažďuje a distribuuje procesní data nakonfigurovaná v Engineering systému pro další zpracování [34].

Operator system tvoří rozhraní HMI. Tento program se používají ke sledování procesů na jednom nebo více monitorech operátorských stanic a lze z nich provádět operace[34].

Engineering, Runtime a Operator systémy tvoří systém APROL a lze je instalovat a používat samostatně nebo v kombinaci mezi sebou na jednom počítači [34].




Obr. 3.2: APROL systém [34]

## 3.4 MariaDB

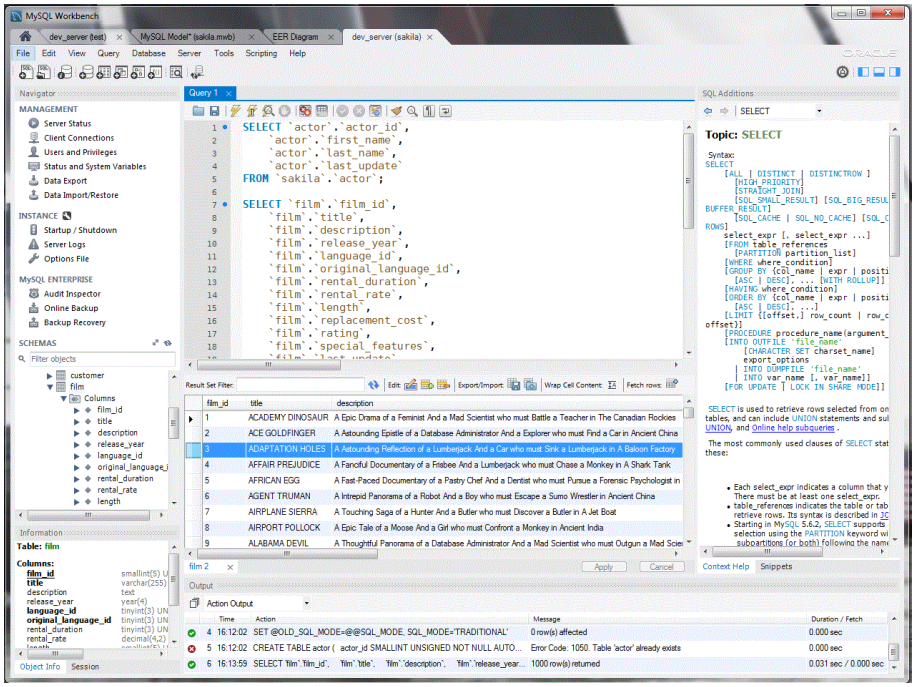
MariaDB Server je jednou z nejpopulárnějších open source relačních databází. Je vyroben původními vývojáři MySQL kteří garantují, že zůstane open source. Je součástí většiny cloudových nabídek a výchozí ve většině distribucí operačního systému Linux [35].

MariaDB mění data na strukturované informace v široké škále aplikací. MariaDB, původně navržený jako vylepšená verze MySQL, je používaný z důvodu rychlosti, škálovatelnosti a robustnosti. Díky bohatému ekosystému úložných enginů, zásuvných modulů a mnoha dalších nástrojů je velmi univerzální pro širokou škálu případů použití. MariaDB také obsahuje funkce GIS a JSON [36].

### 3.4.1 MySQL Workbench

Díky společné provázanosti MariaDB a MySQL, jsem si pro práci s MariaDB databází vybral MySQL Workbench, což je sjednocený vizuální nástroj pro tvorbu databázové architekty a vývojáře. MySQL Workbench poskytuje prostředí pro modelování dat, vytváření, provádění a optimalizaci dotazů SQL a komplexní administrační nástroje pro konfiguraci serveru, správu uživatelů, zálohování a mnoho dalšího

[37]. Kde na obrázku 3.3 je ukázán jeden z nástrojů MySQL Workbench, v tomto případě editor.



Obr. 3.3: MySQL Workbench SQL editor [37]

## 3.5 TIBCO Jaspersoft

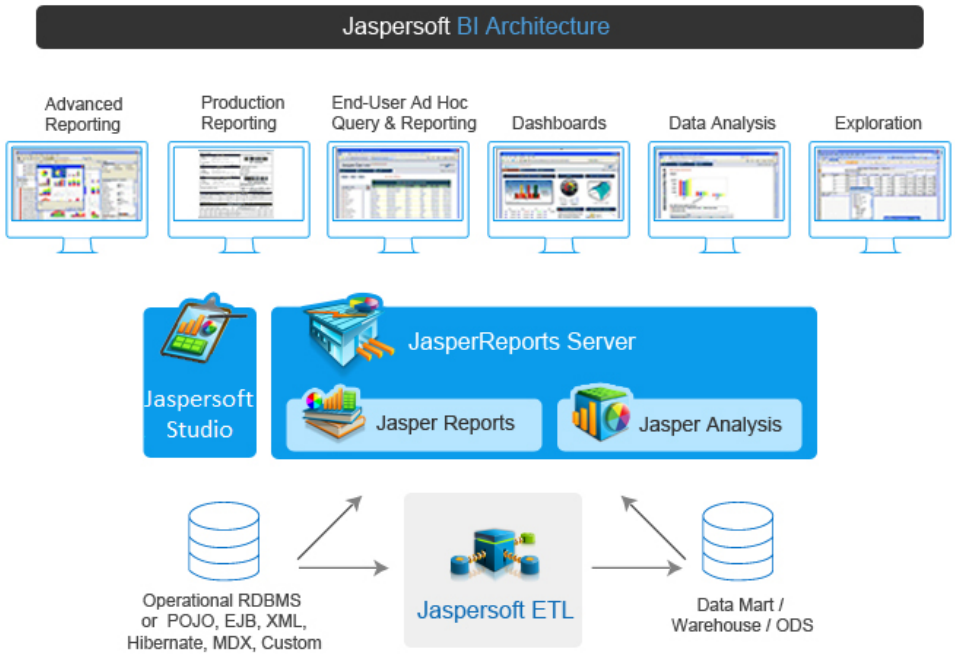
Pro zobrazení zpracovaných dat jsem ve své práci použil SW produkty společnosti TIBCO Jaspersoft, který si berou data z jakéhokoli zdroje a převede je do grafické reprezentace, se kterou lze během prezentace manipulovat za běhu. Výstupné zobrazení lze následně exportovat do několika v dnešní době běžných formátů. Ve své práci jsem především využil tyto dva produkty z této rodiny.

### 3.5.1 JasperReports Server

JasperReports Server je samostatná nebo integrovatelná platforma *Business Intelligence* (BI) umožňující zobrazování plánované nebo v reálném čase dodávané informace pro potřebné uživatele. Jedná se o centrum BI, které poskytuje širokou škálu funkcí pro navrhování, distribuci a bezpečnou správu reportů, dashboardů a vizualizací [38]. Klientem v tomto případě se stávají zařízení s webovým prohlížečem v dané síti.

### 3.5.2 Jaspersoft Studio

Jaspersoft Studio je výkonný návrhářský SW pro vývoj vizualizací a plnohodnotných reportů. Díky velice pokrokovému návrhovému prostředí vám umožňuje vytvářet vysoce formátované reporty a vizualizace v perfektním rozlišení [39].




Obr. 3.4: Struktura Jaspersofr BI suit [39]

Struktura zobrazená na obrázku 3.4 je názornou ukázkou propojení jednotlivých programů společnosti TIBCO Jaspersoft s výpisem jednotlivých funkcí, které poskytuje.

# 4 Simulátor linky

Před tím, než se dostaneme vůbec vyhodnocování a zobrazování dat je potřeba si uvědomit že stavebním kamenem této diplomové práce jsou právě tato statická data a jejich sběr. Jak už tomu bylo zmíněno v úvodu této práce, bohužel se mi nepodařilo pracovat na reálném příkladu, tudíž jsem byl nucen si vytvořit vlastní simulovanou linku, a to nejlépe nejvíc podobné té reálné.

V této kapitole se podíváme blíže právě na tento simulátor, a to přesněji na jeho jednotlivé stavy, jak mezi nimi přechází a čeho všeho je schopný. Také se podíváme jak tento simulátor je propojen s *Automation PC* (APC) a systémem APROL, kde samozřejmě popíši i datovou strukturu pomocí které vygenerované data předává.

## 4.1 Popis simulátoru

Při rozhodování, jak by měl simulátor fungovat jsem se rozhodl pro výrobu v prostředí lisoven se kterýma jsem se prakticky v minulosti setkal a mam alespoň minimální zkušenosti s jejich výrobním postupem. Přesněji o výrobní závod obsahující několik jednoúčelových strojů, vyrábějící zakázkově platové komponenty pro autoprůmysl. Kde tyto stroje jsou plně přenastavytelné na několik druhů výrobků.

Simulátor linky běží uvnitř PLC systému od společnosti B&R, přesněji na typu X20 CP3583. V tomto PLC běží právě dva spolupracující programy, první který simuluje výrobní linku a druhý simulující jednotlivé denní směny a povinné přestávky.

Díky tomu kódu pracuje simulátor kompletně automaticky a není nutný žádný zásah z venčí pro generování dat, samozřejmě se dá kdykoliv přepnout do manuálního režimu. Čímž se přestanou poslouchat signály z části pro denní směny. Manuální režim se v kódu nachází z důvodu možnosti ovlivnění dat a pozorování chování systému na tyto okamžité změny. Myšlenka byla taková že toto ovládání bude vyvedeno na HMI nebo přímo v APC, kde bude simulátor možno jednoduše ovládat. Bohužel od této myšlenky jsem ustoupil a pokud je potřeba se přepnout do manuálního režimu je nutné se napřímo připojit na PLC.

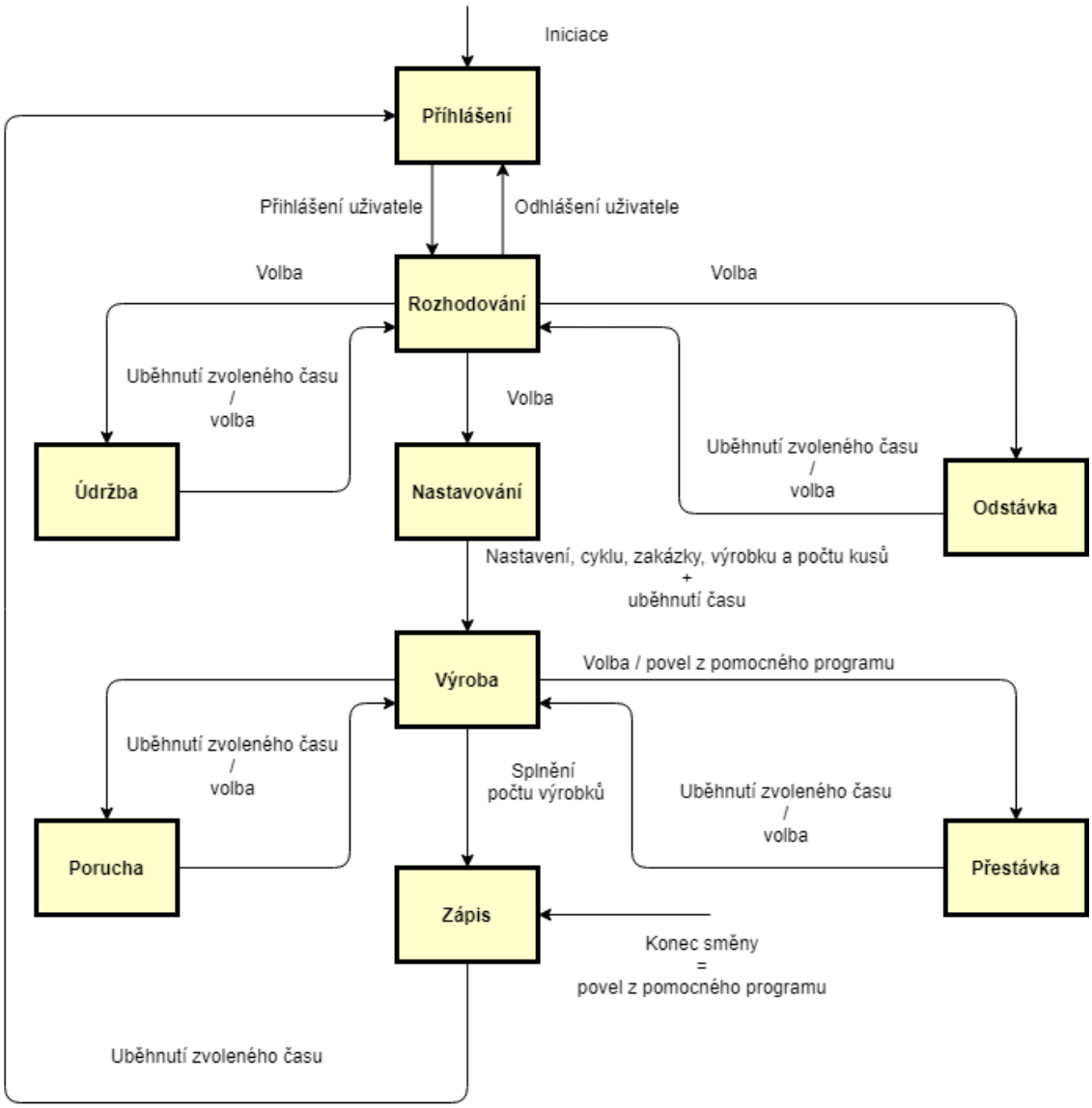
### 4.1.1 Denní směny

Program simulující denní směny jsem přidal do simulátoru z důvodu potřeby otestování chování celého systému záznamu dat v různých pracovních hodinách. Aktuálně je simulátor chopen nasimulovat osmi, dvanácti a dvaceti čtyř hodinový provoz výroby. Tyto provozy mají vliv na počet plánovaných přestávek ve výrobě a možnost změny operátora.

### 4.1.2 Výrobní linka

Kód výrobní linky, který je jádro celého simulátoru, běží pomocí stavového automatu na obrázku 4.1, který jsem uzpůsobil, aby co nejlépe simuloval reálný výrobní proces ve zmíněném závodu. Kde kódem simuluji stavy dané linky/stroje, kterými si při výrobě prochází, jejich časů a informací normálně generující tyto stavy.

Je potřebné říct že i když v této kapitole mluvíme v jednotném čísle, tak v simulátoru tento kód běží dvanáctkrát, tudíž simuluje dvanáct rozdílných linek. Toto naklonování linek je docíleno vektorem níže popsaných datových struktur v programu. Kde každou takovou linku jsem následném umístění kódu do rozdílných taskových tříd. Po zbytek práce budu ale vše popisovat pro první linku v automatickém režimu.




Obr. 4.1: Stavový automat výrobní linky

Než se dostanu k popisu jednotlivých stavů je potřeba si říct, jak dochází u většině stavů k výběru potřebných dat a následnému přechodu da následujícího stavu. Aby linka byla plně automatickou a nejlépe unikátní pro každý den, vytvořil jsem uvnitř kódu random funkci, viz. výpis 4.1. Tato random funkce je volána před každým rozhodnutím (jak dlouho daný stav bude trvat, jméno operátora, počet kusů na výrobu, ...), kde na výstupní hodnotě dané funkce je vybrána odpovídající hodnota. Pokud je v daném cyklu tato funkce volána více jak jednou je změněn dělitel funkce, aby se docílilo rozdílného výsledku.

Výpis 4.1: Random funkce simulátoru

```
// Volání funkce
Random(Delitel := 7);

// Funkce
FUNCTION Random
  Cas := clock_ms();
  Cislo := Cas / Delitel;
  Random := Cislo MOD 100;
END_FUNCTION
```

Funkce pracuje na konstantním milisekundovým čítači, kde výstupem je datový typ TIME. Toto číslo je následně děleno námi zvoleným dělitelem a přes modulo 100 nám vrací hodnotu od 0-99.

### Přihlašování

Iniciační stav, do kterého se simulátor dostává při spuštění linky, skončení směny nebo v případě odhlášení operátora ve stavu rozhodování. V tomto stavu se linka tváří jako kdyby byla odstavená.

### Rozhodování

Další stav, ve kterém se linka tváří jako kdyby byla odstavená, slouží k rozhodnutí, co se s linkou bude aktuálně dít.

### Údržba

Ze stavu Rozhodování je 10% šance že na lince je nutné provést údržbu. Pokud tento stav nastaven je zvolen čas, jak dlouho údržba se bude provádět a po uběhnutí tohoto času se vrací do stavu Rozhodování.

### Odstávka

Obdobně jako při Údržbě, tak i zde je procentní šance na dostání se do tohoto stavu. V tomto případě tato šance činní 12%. Po uběhnutí tohoto určeného času se linka vrací zpět do stavu Rozhodování.

Stav Údržba a Odstávka je identický, rozdílnost mezi nimi je jen v délkách času a následně rozdílnost těchto stavů má dopad na dostupnost této linky.

### Nastavonání

Pokud ze stavu Rozhodování přejde linka do stavu Nastavování, jsou nastaveny podstatné informace pro výrobu. Jako je číslo zakázky, typ výrobku, počet kusů na vyrobení, ideální takt stroje a délka nastavování stroje. Po uplynutí této délky automaticky přechází do stavu Výroba.

### Výroba

Výroba je jeden z dalších rozvětvovacích stavů linky. V tomto stavu se nastaví čas, po který linka vyrábí, v tomto čase linka zůstává v tomto stavu a inkrementuje vyrobené kusy výrobku. Po každém vyrobeném kusu, ale dojde k přepočtu taktu stroje za pomocí ideálního taktu a random funkce. Jakmile nastavený čas doběhne provede se opět random funkce, která určí, do jakého z možných stavů se linka dostane (Porucha, Přestávka nebo opět Výroba), také je tu šance že operátor zapíše NoK (nekvalitní výrobek). Jakmile linka vyrobí požadovaný počet výrobků přechází do stavu Zápis.

### Zápis

Zápis je přidaný stav pro 100% jistotu zapsání potřebných dat do APROL systému z důvodu rychlého přechodu jednotlivých stavů simulátoru a přepisem potřebných dat. To někdy v lince způsobovalo zápisy se špatnými daty a tím zkreslování výsledku. Později tento stav se využívá i k přepočtu dat v databázi. Linka se do něj dostává dvojím způsobem buď ze stavu Výroba, a to tak že byl vyroben požadovaný počet kusů nebo okamžitě v případě že skončila směna.

### Porucha

Pokud se linka dostane do tohoto stavu ze stavu Výroba, kde šance činní cca 20% je nastaven čas trvání poruchy a typ poruchy. Jakmile dojde k vypršení tohoto času typ poruchy se nuluje a linka přechází opět do stavu Výroba.

#### Přestávka

Stav Přestávka může nastat dvěma způsoby. Pokud se jedná o plánovanou přestávku, kterou nám hlídá program Denní směny nebo v případě neplánované přestávky ze stavu Výroba s 10% šancí. Po uplynutí nastaveného času přestávku se vrací linka do stavu Výroba. Po čas přestávku je stroj zastaven.

## 4.2 Spojení PLC a APROL

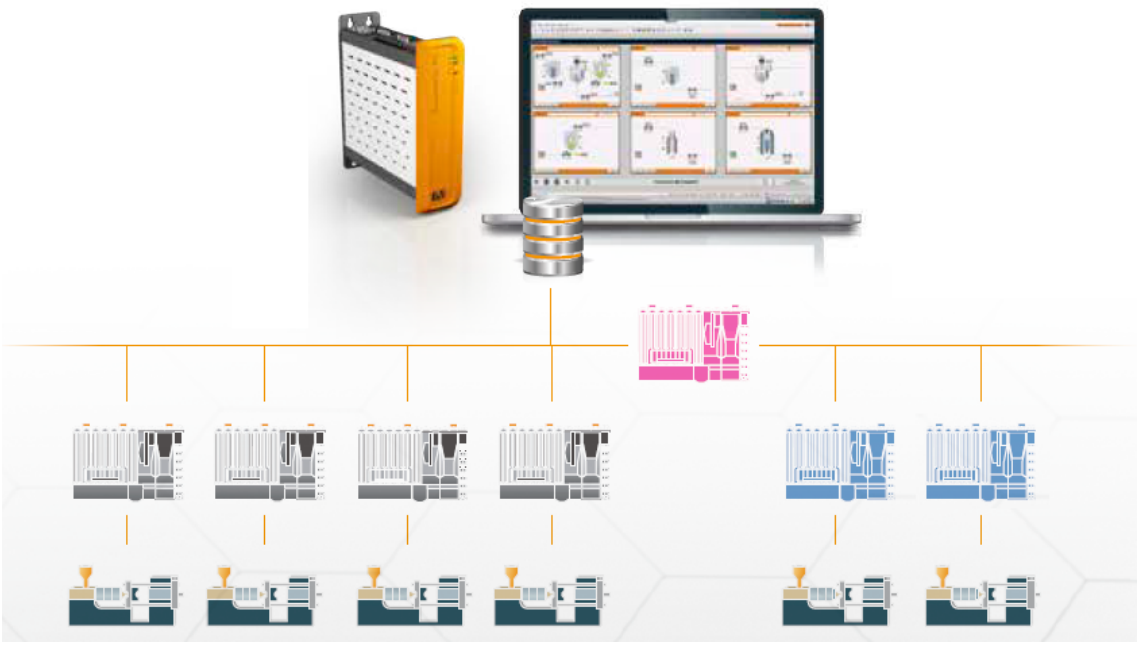
APROL systém nabízí různé způsoby navázání komunikace s požadovanými PLC. Jedním způsobem je nakonfigurovat PLC přímo v prostředí APROL a tím umožnit jejího programování v tomto prostředí, pomocí *Continuous Function Chart* (CFC). Další způsob zase umožnuje připojení PLC různých výrobců nebo umožní připojit PLC do kaskády tím že všechny PLC v síti budou moci mezi sebou komunikovat. Ale pro mou diplomovou práci jsem si vypral způsob, který umožnuje nakonfigurovat a naprogramovat PLC v prostředí Automation Studio a následně připojení k APROL systému pomocí interního driveru ANSL. Kde tato komunikace se jmenuje **PDA connection**.

Tento způsob komunikace je v daném případě nejefektivnější a umožnuje rychlé připojení na potřebné PLC a provést potřebné změny bez sebemenší znalosti APROL systému. Můj případ uvažuje že mnou simulovaná linka není řízena PLC od společnosti B&R, ale jiného výrobcem. PLC od společnosti B&R je připojený k jednomu nebo více řídícím PLC jiného výrobce formou tzv. ostrůvkem pro sběr dat a kontrolu dat. Tohle je ukázáno na obrázku 4.2, kde šedé PLC jsou napřímo připojená PLC značky B&R. Modrá PLC jsou na druhou stranu jiného výrobce a právě růžové je zmíněný ostrůvek, pro jejich propojení s APROL systémem.

S takto spojeným PLC se systémem APROL se v systému vyberou a nakonfigurují požadované proměnné nebo struktury, jinak řečeno body, potřebné k následnému zpracování. Takto nakonfigurovaná komunikace mezi PLC a systémem APROL může být jednosměrná nebo obousměrná a v případě konfigurace může dojít k drobnému přetypování proměnných (STRING -> SSTRING nebo LSTRING).

## 4.3 Komunikovaná datová struktůra

V rámci PDA connection může mezi PLC a APROL systémem komunikovat libovolný počet proměnných nakonfigurovaných právě v PLC. Bohužel s větším počtem komunikujících proměnných se konfigurace v systému APROL stěžuje a stává se zbytečně zdlouhavá. Proto je důležité si určit jaké proměnné potřebujeme v APROL



Obr. 4.2: Ukázka možného zapojení pomocí PDA Connection

systému pro následné zpracování a tyto data zapouzdřit do datové struktury. Tím se následná konfigurace zjednoduší a zefektivní.

V následující kapitole popíšu právě tuto mnou vytvořenou strukturu viz. obrázek 4.3. Jak je možno vidět celá struktura se skládá ze dvou podstruktur, a to pro posílání a příjem dat z a do APROL systému. Zbylé datové struktury celého simulátoru je možno si prohlédnout v příloze A.

| Name | Type | & Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Aprol_receive_typ | | | ☑ | | Struktura pro prijem z APROLu |
| Nok | BOOL | ☐ | ☑ | | Potvrzeni prijenti signalu NOK kusu |
| Aprol_send_typ | | | ☑ | | Struktura pro odeslani do APROLu |
| DevName | STRING[65] | ☐ | ☑ | | Nazev/kod stroje |
| User | STRING[65] | ☐ | ☑ | | Jmeno cloveka, ktery stroj obsluhuje |
| Contract | STRING[65] | ☐ | ☑ | | Kod zakazky |
| Product | STRING[65] | ☐ | ☑ | | Kod produktu |
| DevState | USINT | ☐ | ☑ | | Zakodovany stav stroje |
| ProgState | USINT | ☐ | ☑ | | Zakodovaný stav programu |
| NokNumber | USINT | ☐ | ☑ | | Pocet vadnych kusu, ktere se odvedli kdyz prisel Nok (BOOL) |
| PartCount | UINT | ☐ | ☑ | | Pocet vyrobenych kusu |
| ErrType | UINT | ☐ | ☑ | | Zakodovany kod poruchy na stroji |
| NokType | UINT | ☐ | ☑ | | Zakodovany kod vady kusu |
| Nok | BOOL | ☐ | ☑ | | Zadani NOK kusu na hranu |
| Aprol_typ | | | ☑ | | Struktura pro APROL |
| Send | Aprol_send_typ | ☐ | ☑ | | Odesilana data |
| Receive | Aprol_receive_typ | ☐ | ☑ | | Prijmana data |

Obr. 4.3: Datová struktůra komunikující s APROL systémem

- **DevName** - Stringová proměnná *DevName* je jedna z mála proměnných které zůstávají statické po čas celé doby, ale na druhou stranu jedna nejdůležitějších pro následnou filtraci dat v databázích. Tato proměnná udává název stroje.
- **User** - Proměnná *User* udává přímé jméno operátora nebo jeho kód. Myšlenka je taková že při začátku směny se operátor přihlásí a všechny akce se budou

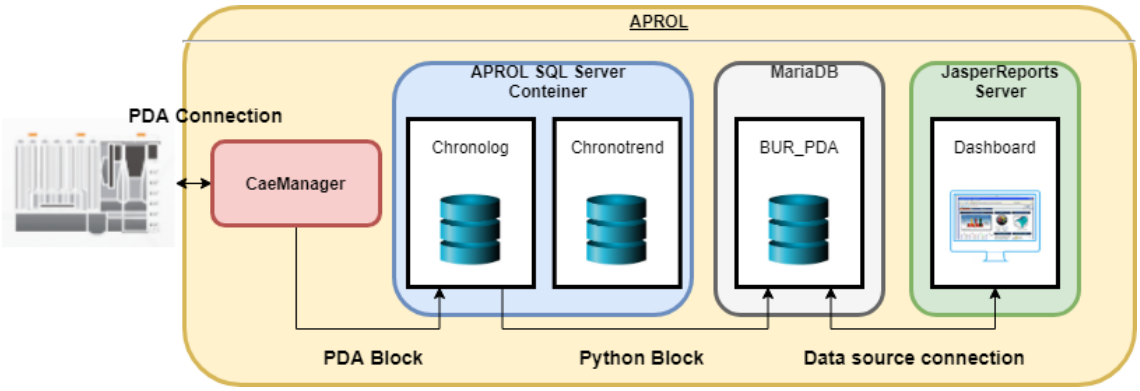
logovat pod jeho jménem.

- **Contract** - *Contract* je stringová proměnná udávající číslo nebo kód zakázky pro daný stroj. V praxi ty to mělo fungovat, že tato proměnná by se načítala do stroje přes připojené RFID nebo čtečku BAR kódu.
- **Product** - Podobně jako v případě proměnné *Contract*, důvodem rozdělení je možnost více produktů v jedné zakázce.
- **DevState** - Číselný stav stroje korespondující se stavy stavového automatu na obrázku 4.1. Jedná se o mechanické stavy stroje.
- **ProgState** - Stav programu stoje. Stroje může pracovat v jednotlivých mechanických stavech, kde každý tento stav může výt následně rozdělen na programové stavy. Například jako v mém jednoduchém případě AUT/MAN režim.
- **Nok** - Jediná proměnná pro příjem i odeslání z APROL systému. Jakmile APROL obdrží log. 1 z přijímaného *Nok* přepíše hodnotu do odesílaného.
- **NokNumber** - Proměnná *NokNumber* udává počet nekvalitních kusů na náběžnou hranu proměnné Nok. Jakmile PLC zaznamená log. 1 z APROL systému jako potvrzení přijetí proměnné se hodnota vynuluje.
- **NokType** - *NokType* je číselná porměnná udávající kód důvodu nekvalitního kusu. Podobně jako u *NokNumber* se po obdržení log. 1 z APROL systému nuluje.
- **PartCount** - Inkrementující číselná proměnná značící počet vyrobených kusů produktu. Pro každou výrobu se začíná inkrementovat od začátku.
- **ErrType** - Číselný kód nastalé poruchy stroje. Tato proměnná má v normálním stavu hodnotu 0.

# 5 Sběr a vyhodnocení dat

V předchozí kapitole jsem si řekli že PLC a APROL systém mezi sebou komunikují pomocí vytvořené struktury dat a PDA Connection, zde si řekneme co vše s těmito daty děláme v APROL systému a jak se následně dostanou do MariaDB databáze.

Než se ale dostaneme k jednotlivým podkapitolám, je nutné si objasnit celkovou návaznost jednotlivých celků. Tohle objasnění následně pomůže k rychlejšímu a jasnějšímu pochopení celého projektu. Jak můžeme vidět na obrázku 5.1 data z PLC se dostávaní do interního programu CeaManager ve kterém probíhá jejich předzpracování a následné uložení do APROL SQL Serveru, neboli Containeru. Conteiner obsahuje mnoho menších subdatabází celého APROL systém jako alarmy, audittrail, shiftlog, syslog nebo mnou používaný chronolog a mnoho dalších. Chonolog je interní databází APROL systému, do které je umožněno uživatelům ukládat jimi potřebná data. Tato subdatabáze má určitá omezení, ke kterým se dostaneme v pozdější části práce. Do Chronologu se následně přistupuje pomocí Python scriptu, který kopíruje potřebná data do MariaDB databáze pro lepší a efektivnější správu. Tato databáze je následně propojena s JasperReports Serverem, který zprostředkovává Dashboardy.




Obr. 5.1: Blokové schéma projektu

Tímto byla prezentována koncepce diplomové práce, nyní budou jednotlivé části popsány podrobněji.

## 5.1 Předzpracování dat a uložení do Chronologu

Za účelem předzpracování dat a následného uložení do interní APROL SQL databáze Chonologu jsem v projektu pomocí CeaManageru vytvořil knihovnu, jejíž struktura je zobrazena na obrázku 5.2. Tato knihovna posléze umožnuje jednoduché přidávání

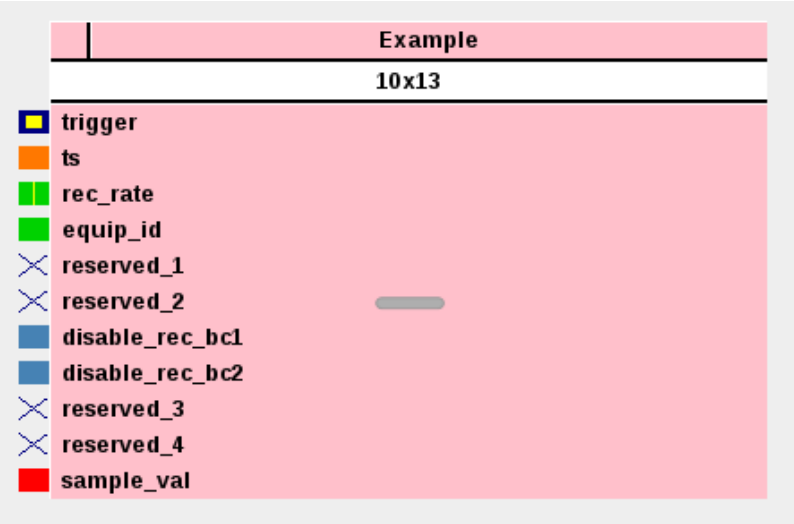
dalších výrobních linek do systému s kompletním ukládáním všech dat nebo jen vybráním potřebných informací.


Obr. 5.2: Struktura knihovny v CeaManageru

Důležitým prvkem této knihovny je nakonfigurování PDA bloků pro ukládání dat do Chronologu. Příklad nenakonfigurovaného PDA bloku můžete vidět na obrázku 5.3. Tyto bloky jsou klíčové pro ukládání dat, kde uložení probíhá vždy na náběžnou hranu signálu *trigger*. Pokud není přivedena vlastní časová stopa do signálu *ts* je použit systémový čas APROL-u. Proměnná *rec_rate* zase umožnuje rozdělit ukládaná data do dvou podčástí Chronologu, tyto podčásti jsou *Fast* a *Slow*. *Equip_id* umožňuje přidání pomocného identifikátoru pro následnou filtraci dat. Disable proměnné jsou v tomto bloku pro potřebu ignorování nějaké námi nakonfigurované proměnné při daném cyklu zápisu, tuhle funkcionalitu v mé práci ale nepoužívám. Signál *sample_val* nahrazujeme námi zvolenými proměnnými.

Z důvodu komunikačních prodlev ve spojení a automatického nastavování vykonávání bloků v CFC zabralo odladění použitého kódu delší množství času, než bylo plánováno. I přes první náznak funkčnosti se některé chyby projevili až při dlouhodobém testování nebo při zprovoznění pro více linek. Každý takto vytvořený blok vytváří tabulku v Chronologu databázi s námi nakonfigurovaným počtem sloupců pro data plus dalšími systémovými sloupci. Proto je potřeba optimalizovat rozložení tabulek již v této části práce. Přesný vzhled a rozložení tabulek si popíšeme v kapitole 5.2.2.

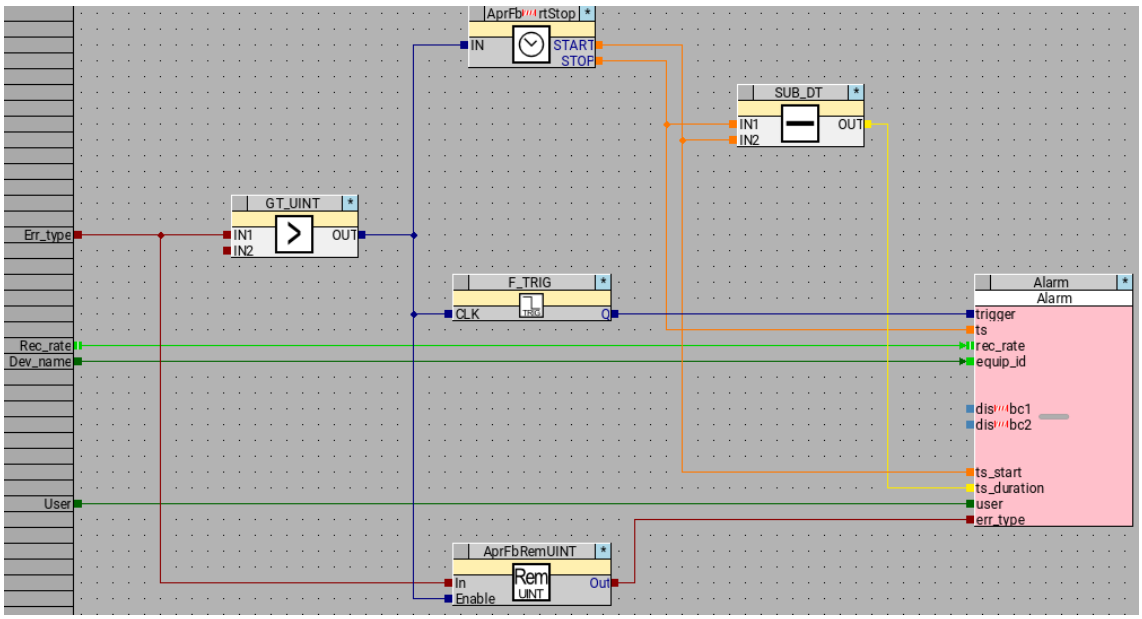
Pro svou práci jsem se rozhodl vytvořit pět specifických tabulek, kde pro každou z těchto tabulek předzpracovávám obdržená data, aby měli co největší samostatnou vypovídající hodnotu. Každá z těchto tabulek má následně svůj díl ve vyhodnocování výroby.




Obr. 5.3: Ukázka PDA bloku

### 5.1.1 AlarmPDA

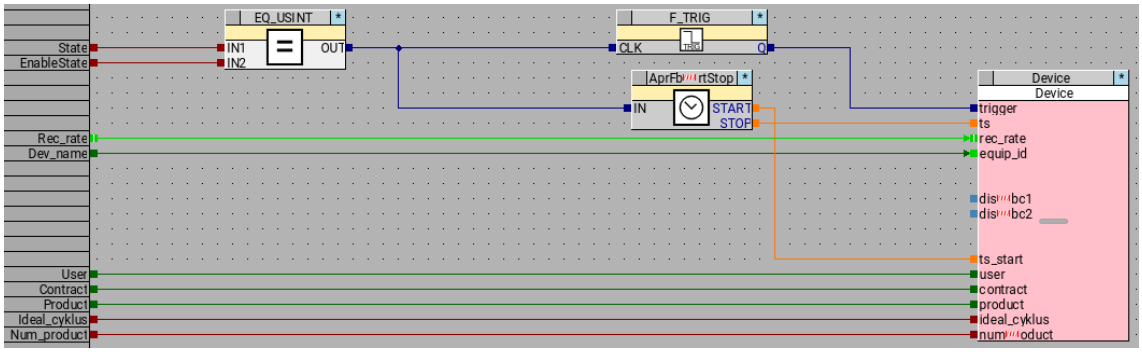
První z vytvořených tabulek je tabulka pro záznam alarmů, viz. obrázek 5.4. Do tabulky zaznamenávám, na jakém zařízení porucha nastala, o jakou porucha se jedná, ale k tomu i časový údaj kdy daná porucha nastal a skončil i s vypočítanou délkou trvání. Tyto údaje nejsou fatální pro výpočet OEE ale velice důležité například pro Paretovu analýzu daného stroje, vyhodnocení MTBF nebo MTTR.




Obr. 5.4: Předzpracovaní dat pro zápis alarmů

### 5.1.2 DevicePDA

Tabulka Device je příkladem jednoduššího předzpracování, ale na druhou stranu hraje velkou roli v následném vyhodnocování dat v MariaDB databázi, v kapitole 5.3. S ukončením nastavování linky se s názvem linky do databáze zapíše i čas, zakázka a produkt pro který stroj byl nastavován. K tomu jsou přidány i informace kolik kusů se mělo vyrobit s jakým ideálním taktem stroje a samozřejmě kdo měl danou výrobu na starosti. Zápisu do tabulky Device se týká obrázek 5.5.




Obr. 5.5: Předzpracovaní dat pro zápis nastavení strojů

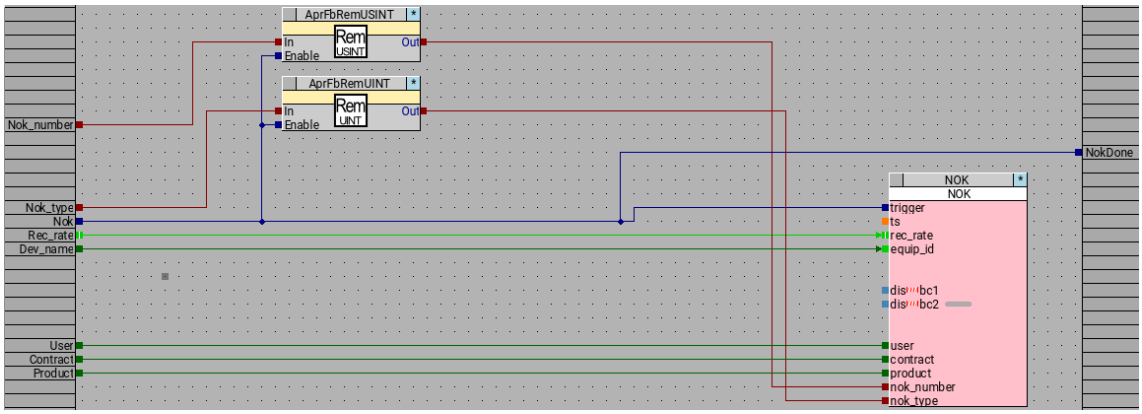
### 5.1.3 NokPDA

Zápis nekvalitních kusů z výroby je jediný blok, obrázek 5.6, u kterého funguje obousměrná komunikace. Jedná se o potvrzení přijetí těchto dat a tím zabránění případného nezapsání v případě krátkodobého výpadku komunikace. Při potvrzení nekvalitního kusu signálem Nok se do databáze zapíše jméno daného zařízení, systémový čas, počet nekvalitních kusů a jejich typ. Zápis také obsahuje informaci, o jaký produkt se jedná, u jaké zakázky byl nekvalitní kus pořízen a identifikační údaj daného operátora. Údaje z této tabulky jsou potřebné pro následný výpočet indexu Quality ale je možné použít i pro další vyhodnocení nebo informativní zobrazení v reportech.

### 5.1.4 ProdukcePDA

Pro záznam produkce dané výroby jsem se rozhodl nepoužívat přímo hodnotu inkrementující proměnné z PLC, z důvodu obtížného zpracování v databázi, ale rozdíl vyrobených kusů za daný časový úsek. Tento časový úsek je nastavitelný, takže každý stroj jej může mít individuální. Celé zapojení je možno vidět na obrázku 5.7.

Záznam výroby se provádí cyklicky jedině pokud je stroj ve stavu Výroby, pokud jen na okamžik stroj přejde do jiného stavu například Poruchy nebo Přestávku

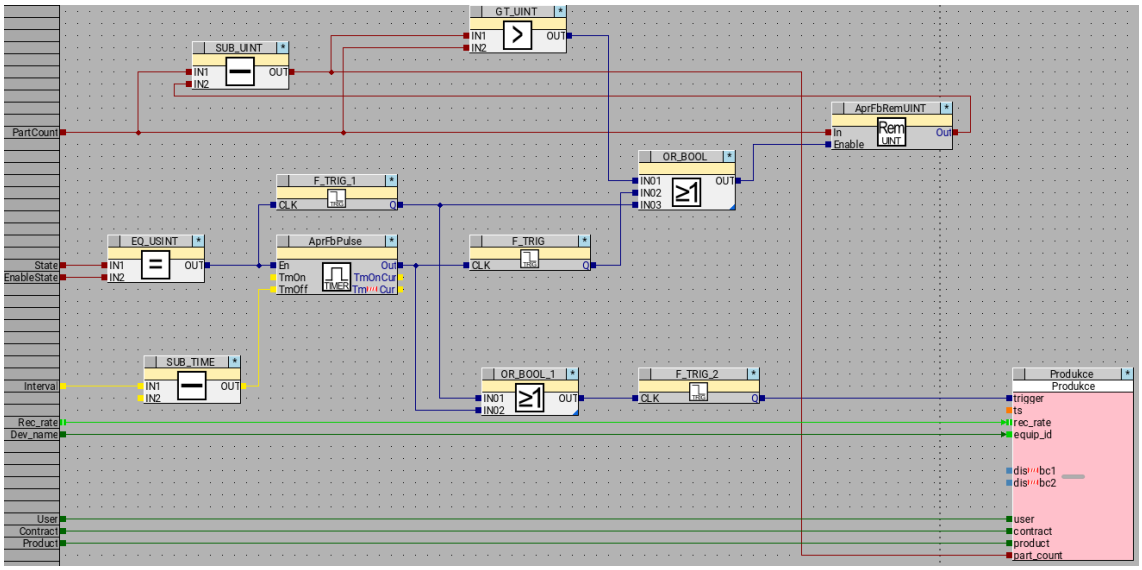



Obr. 5.6: Předzpracovaní dat pro zápis nekvalitních kusů

aktuální rozdíl vyrobených kusů se zapíše do databáze i přes nenaplnění podmínky cyklu. Při opětovném vrácení do stavu Výroby začíná celý cyklus znovu.

S rozdílem produkce za daný časový cyklus se zapisuje do tabulky i název stroje, název zakázky a produktu se jménem operátora zodpovědného za danou výrobu.

Při tomto bloku jsem se potkal s nejvíce problémy v rámci testování. Prvním problémem byl pravidelně se opakující záznamy v databázi se nereálnou produkcí, což bylo docílené vždy prvním záznamem pro novou výrobu a jeho špatným rozdílem. Po lokalizovaní tohoto problému byl opraven přidáním jednoduchého porovnávání hodnot a následným resetováním uložené proměnné. Druhým problémem bylo občasné mizení jednoho vyrobeného kusu při zápisu. Tento problém nastal díky již zmíněnému automatickému nastavování vykonávání bloků v CFC, problém jsem vyřešil přidáním bloků hlídající sestupnou hranu signálu.

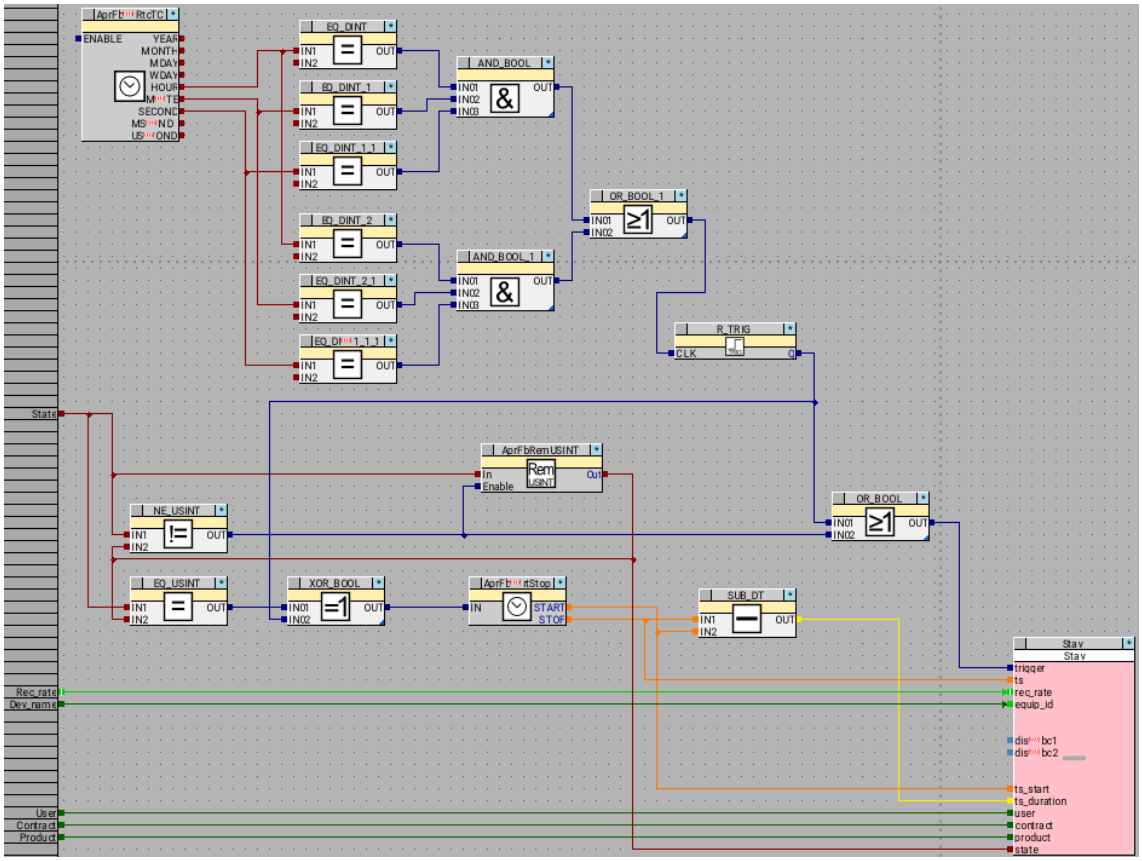



Obr. 5.7: Předzpracovaní dat pro zápis produkce

### 5.1.5 StavPDA

Poslední tabulkou ukládanou do Chronologu je tabulka pro záznam stavů stroje, viz. obrázek 5.8. Jakákoliv změna stavu na stroji je zaznamenána do databáze společně s názvem stroje, časem začátku a konce tohoto stavu, délkou trvání s milisekundách a pro návaznost na další tabulky i zde je zaznamenáno jméno zakázky, produktu a obsluhy.

Z důvodu častých požadavků od zákazníka na zobrazení těchto dat formou Gantova diagramu nebo denních reportů, jsem musel do kódu přidat bloky pro zajištění zapsání dat na konci dne a znovu druhý den na jeho začátku. Tímto jsem zajistil že data společná pro daný den jsou samostatná a nemusíme se uchylovat ke složitým SQL dotazům v databázi pro jejich rozdělení.




Obr. 5.8: Předzpracovaní dat pro zápis stavů stroje

## 5.2 Přenos dat do MariaDB

Jakmile jsou data uležena v Chronologu můžeme přejít na další krok korelující i se zadáním diplomové práce, a to je přesunutí těchto dat do MariaDB. Důvodem

tohoto přesunutí je nedostatečnost Chronologu. Conteiner, i když se jedná o velice robustní platformu SQL Server, již nepodporuje všechny aktuální funkce, jelikož využívání dotazovacího jazyka SQL- 92 v případě složitějších, nebo rozsáhlejších SQL dotazů na jejich vykonání zabírá delší čas. Jedním z dalších důvodů je absolutní nemožnost editace dat v databázi, ale hlavním problémem této databáze je omezenost její kapacity. Velikost Conteineru i Chronologu je omezena APROL systémem a jakmile dojde k jejich naplnění automaticky se začnou přepisovat nejstarší data v těchto databázích.

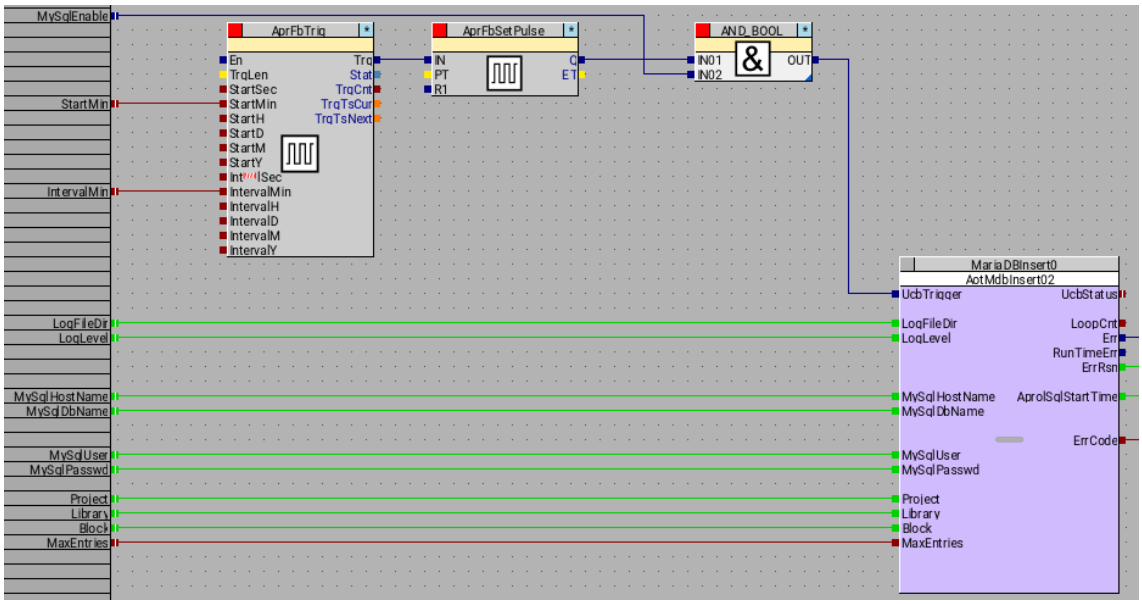
Využívání externí databázové platformy, bohužel není také dokonalé. Nastává tu možnost výpadku komunikace mezi APROL systémem a Serverem, především pokud se Server nenachází přímo v APC. To vede ke ztrátě dat a zkreslování výsledků. Proto jsem zvolil variantu využití dvou SQL serverů. Interního pro krátkodobé uložení dat a externího pro dlouhodobé uložení a lepší pracovní vlastnosti.

### 5.2.1 Použitý prostředek pro přenos dat

Pro tento přenos využívám jeden z bloků ze speciální knihovny společnosti B&R Automation. Tato knihovna je běžně dostupná na vyžádání a nenachází se v instalačním balíčku APROL systému. Jedná se o UCB blok, v němž běží Python script, tento script se provede vždy jakmile konektor *UcbTrigger* obdrží náběžnou hranu signálu. Zbylé bloky zobrazené na obrázku 5.9 slouží k cyklickému volání UCB bloku v požadovaných cyklech. Pro každou tabulku Chronologu je nutné použít nový UCB blok. Samozřejmě je nutné si uvědomit že v případě spuštění celého systému na novo je potřeba UCB bloky připojovat k databázi po jednom, proto je tu možnost zpozdit start bloku o nastavený čas, ale i tak je potřeba být na pozoru, aby se přenosy nepřekrývali.

UCB blok je také potřeba nakonfigurovat dalšími potřebnými parametry pro navázání spojení s MariaDB. Mezi tyto údaje patří název projektu v CeaManageru, knihovna, kde se nachází PDA blok a název tohoto bloku. Tím určíme, jakou tabulku Chronologu se jedná. Následně je také potřeba zadat adresu MariaDB Serveru, název databáze a přihlašovací údaje, pomocí kterých se připojíme k externí databázi.

Jakmile Python script naváže spojení s MariaDB databází provede se dotaz na nejnovější časovou značku v dané tabulce MatiaDB. Pomocí této značky následovně provádí dotaz v Chronologu, jestli existují nějaká data s novější časovou značkou, pokud ano, tak se tato data ukládají do pole. V opačném případě se nic neděje. Toto pole je následně použito jako vstupní proměnná funkce pro insert nových dat do MariaDB. Je naprosto nutné, aby počet a rozložení sloupců obou databází byli naprosto identické. Jakmile je daný přenos hotov, script se odpojí od MariaDB a je možné se připojit s další tabulkou. Pokud v jakékoliv části kódu nastane chyba




Obr. 5.9: Výřez bloku pro transfér dat do MariaDB

vypíše se v alarmovém logu příslušná hláška.

### Nastalé problémy s přenosem

Již při testování funkčnosti UCB bloku pro jednu linku se vyskytl problém v přenosu, kvůli kterým jsem musel částečně upravit daný script. Další problém jsem objevil při rozšíření na sběru dat na více linek.

Jednalo se o zaznamenávání duplicitních řádků posledních hodnot v Chronologu do MariaDB databáze. Nejvíce patrné to bylo v případě tabulek, do kterých se tak pravidelně nezapisovaly hodnoty. Každým cyklem zavolání UCB bloku se duplikovala poslední hodnota tabulky. Pro prvním pokusu vyřešit tento problém jsem chybu identifikoval v porovnávání hodnot, kde jsem pro vyřešení upravil script. Bohužel to problém vyřešilo jen částečně a stále se někdy objevovali duplicitní řádky.

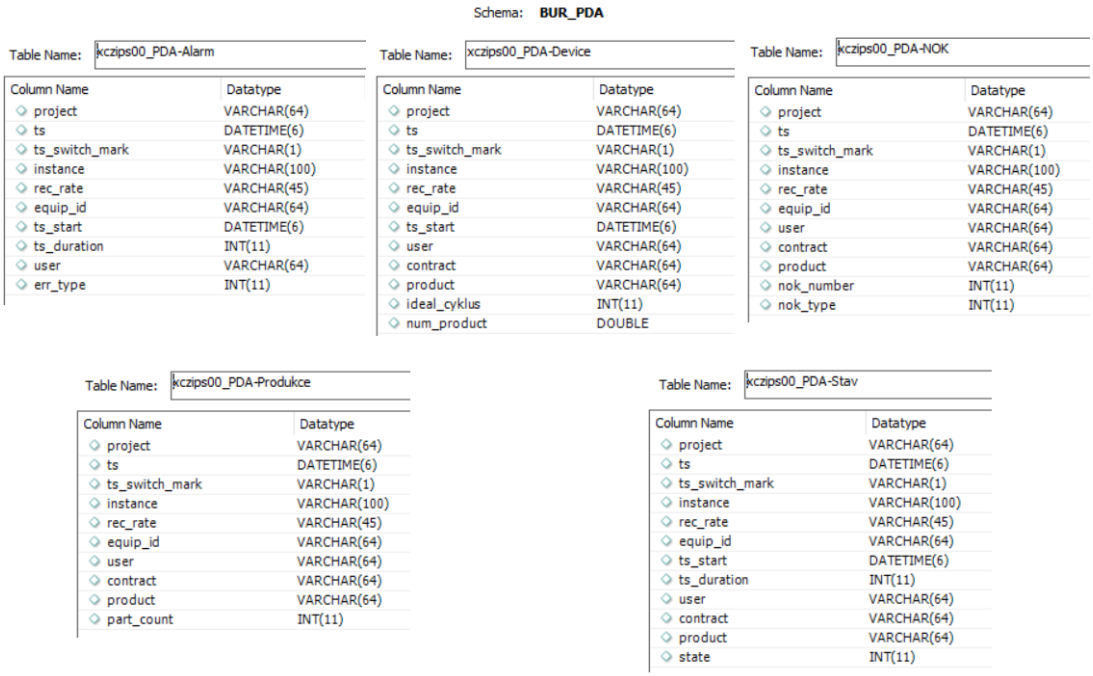
Při druhém pokusu vyřešit tento problém jsem zjistil že problém je způsoben zaokrouhlováním časové stopy. Pokaždé když se porovnávala získaná časová stopa z MariaDB s hodnotami ve Chronologu vyšlo v rámci zaokrouhlení, že některá hodnota má rozdílkou časovou stopu, i když se jednalo o stejný řádek. Důvodem byl počet desetinných míst časové stopy z MariaDB. Ta měla milisekundy na tři desetinná místa, zatímco časová stopa z Chronologu na šest desetinných míst. Tento problém se vyřešil nastavením správného počtu desetinných míst v deklaraci tabulek MariaDB a další úpravou scriptu.

Poslední problém při přenosu jsem zaznamenal, jakmile jsem rozšířil kód na více linek. Stávalo se hlavně ze začátku směny, když některé data se nepřenesly do

MariaDB. To bylo způsobeno špatnou volbou pořadí časových stop při zápisu do Chronologu. Jejich změnou se vše vyřešilo.

### 5.2.2 Přenesené tabulky

Jak už bylo řečeno dříve z důvodu použití scriptu v UCB bloku je velice nutné, aby počet sloupců a jejich rozložení bylo stejné pro obě databáze. Nevýhodou využití této přenosové metody jsou přebytečné systémové sloupce obsažené v Chronologu. Tyto sloupce již nemají zásadní vliv pro funkčnost v Chronologu a jsou pozůstatkem starších funkcionalit.




Obr. 5.10: Deklarace všech přenesených tabulek v MySQL Workbench

Z obrázku 5.10 je vidět, že členění tabulek se od sebe velice moc neliší. V každé tabulce nalezneme systémovou hlavičku a následně potřebná data na zpracování. Pro snížení nevýhody systémových sloupců se snažím využít co nejvíc proměnných pro uložení dat generovaných simulátorem. Jako například časová značka *ts* nebo *equip_id*. Pokud se koukneme přímo na data v tabulce, viz obrázek 5.11, můžeme vidět, že některé tyto proměnné nejsou vůbec využity a zbylé nemají žádnou informační hodnotu.

Všechny tabulky jsou spolu provázané hodnotou *equip_id*, což je stringová proměnná udávající označení stroje. Pomocí této proměnné můžeme data spojovat, takže při správném dotazu zjistíme, kolik daná linka vyrobí za dané období kusu a při tom kolik z těchto kůsu bylo nekvalitních. *Equip_id* není jediná hodnota pomocí

| project | ts | ts_switch_mark | instance | rec_rate | equip_id | user | contract | product | part_count |
|---|---|---|---|---|---|---|---|---|---|
| PDADip | 2020-04-21 10:47:48.044944 | | ToChronolog_DataToChronologLine3_ProdunkcePDA_Produkce | NULL | NX-38 | Uzivatel 02 | Zakazka 251 | Produkt 03 | 34 |
| PDADip | 2020-04-21 10:47:57.553924 | | ToChronolog_DataToChronologLine2_ProdunkcePDA_Produkce | NULL | NX-37 | Uzivatel 01 | Zakazka 250 | Produkt 01 | 97 |
| PDADip | 2020-04-21 10:48:13.068810 | | ToChronolog_DataToChronologLine7_ProdunkcePDA_Produkce | NULL | NX-42 | Uzivatel 02 | Zakazka 386 | Produkt 03 | 26 |
| PDADip | 2020-04-21 10:48:31.587288 | | ToChronolog_DataToChronologLine5_ProdunkcePDA_Produkce | NULL | NX-40 | Uzivatel 02 | Zakazka 385 | Produkt 03 | 25 |
| PDADip | 2020-04-21 10:49:37.650824 | | ToChronolog_DataToChronologLine11_ProdunkcePDA_Produkce | NULL | NX-51 | Uzivatel 04 | Zakazka 210 | Produkt 02 | 52 |
| PDADip | 2020-04-21 10:52:35.850414 | | ToChronolog_DataToChronologLine1_ProdunkcePDA_Produkce | NULL | NX-36 | Uzivatel 03 | Zakazka 249 | Produkt 03 | 31 |
| PDADip | 2020-04-21 10:53:08.879798 | | ToChronolog_DataToChronologLine3_ProdunkcePDA_Produkce | NULL | NX-38 | Uzivatel 02 | Zakazka 251 | Produkt 03 | 33 |

Obr. 5.11: Ukázka zobrazení dat po přenosu do MariaDB v MySQL Workbench

které můžeme jednotlivé tabulky provázat. Mezi další varianty patří využití hodnot *contract* a *product*. V mém projektu ale využívám primárně *equip_id* a zbylé zmíněné spíše jako upřesnění zařazení.

Takto vytvořené tabulky obsahují všechny podstatná data pro vyhodnocení efektivnosti výroby a jsou dostatečně vhodná pro práci v programech od společnosti Jaspersoft. Nicméně kvůli složeným a rozsáhlých SQL dotazů nutných pro práci jsem se rozhodl pro další optimalizaci.

## 5.3 Optimalizace MariaDB databáze

Tato optimalizace spočívá v redukci složených a rozsáhlých SQL dotazů potřebných pro výpočet KPI indexů v programech Jaspersoft. To jsem docílil vytvořením dalších tabulek v databázi. Přesněji třech relačních tabulek pro informační účely a dvou výpočtových tabulek.

Zmíněné tabulky pro informační účely jsou jednoduché dvousloupcové tabulky, které doplňují slovní popis číselným datům z výroby (stav stroje, typ nekvalitního kusu a porucha stoje), všechny tyto tabulky můžete vidět na obrázku 5.12. Jedná se o relační tabulky v databázi tudíž hodnota v prvním sloupci je nastavena jako *Primary Key* (PK) a je unikátní v celé tabulce. Díky tomuto klíči jsou tabulky spojeny s přenesenými tabulkami Alarm, NOK a Stav. Takto v případě dotazu se nemusí již vypisovat zapsané číslo, ale přímo daná informace. Tyto tabulky přidávají kosmetickou hodnotu, ale v případě SQL dotazů nesnižují složitost spíše naopak, ale jejich použití je vhodné.

Na druhou stranu výpočtové tabulky tuto složitost v určitém ohledu snižují. Tím je myšleno že funkčnost zůstává stejná a díky těmto tabulkám je možné jednotlivé dotazy rozdělit na dílčí a data předzpracovat do formy se kterou se bude následně jednodušeji pracovat a nepracovat nárazově. Deklaraci obou výpočtových tabulek můžete vidět na obrázku 5.13. Obě tabulky se zabývají výpočtem OEE ale každá z jiného pohledu. První z tabulek se zabývá výpočtem OEE pro simulátorem generované zakázky a produkty u daného stroje, tento přístup je nutný především z důvodu rozdílných ideálních výrobních taktů stroje u jednotlivých výrobků, tedy pro výpočet Performance indexu. Následně výpočet celkového OEE má vypovídající

| err_type | description |
|---|---|
| 2241 | Porucha 8 |
| 2314 | Porucha 3 |
| 2412 | Porucha 6 |
| 2475 | Porucha 1 |
| 2651 | Porucha 2 |
| 2753 | Porucha 9 |
| 2852 | Porucha 10 |
| 2875 | Porucha 4 |
| 2951 | Porucha 5 |
| 2985 | Porucha 7 |
| NULL | NULL |

| nok_type | description |
|---|---|
| 3321 | Vada 02 |
| 3452 | Vada 05 |
| 3512 | Vada 01 |
| 3622 | Vada 04 |
| 3845 | Vada 03 |
| 3999 | Vada 06 |
| NULL | NULL |

| state | description |
|---|---|
| 0 | Zápis dat |
| 10 | Odstávka |
| 20 | Údržba |
| 30 | Nastavování |
| 40 | Výroba |
| 50 | Porucha |
| 60 | Přestávka |
| NULL | NULL |

Obr. 5.12: Přidané tabulky s informacemi

hodnotu pro důležitost operativního plánování ve výrobě. Druhá tabulka je naopak výpočtem OEE pro daný stroj za uběhlých dvacet čtyři hodin, což je v praxi nejpožadovanější informace.

Table Name: OEE_contract

| Column Name | Datatype |
|---|---|
| ts | DATETIME |
| equip_id | VARCHAR(64) |
| contract | VARCHAR(64) |
| product | VARCHAR(64) |
| part | INT(11) |
| nok | INT(11) |
| ts_complete | INT(11) |
| ts_production | INT(11) |
| ts_downtime | INT(11) |
| ideal_cyklus | INT(11) |
| num_product | DOUBLE |
| availability | FLOAT |
| performance | FLOAT |
| quality | FLOAT |
| OEE | FLOAT |

Table Name: OEE_device

| Column Name | Datatype |
|---|---|
| date | DATE |
| equip_id | VARCHAR(64) |
| part | INT(11) |
| nok | INT(11) |
| ts_production | INT(11) |
| ts_maintenance | INT(11) |
| ts_setting | INT(11) |
| ts_disorder | INT(11) |
| ts_break | INT(11) |
| availability | FLOAT |
| performance | FLOAT |
| quality | FLOAT |
| OEE | FLOAT |

Obr. 5.13: Deklarace tabulek k výpočtu OEE

### 5.3.1 Naplnění tabulek a výpočet OEE indexů

V rámci mé práce jsem přišel na dvě možnosti, jak naplnit daty tabulky, které jsem připravil a v textu zmínil dříve. První způsob byl, podobně jak tomu bylo v případě přesouvání dat z Chronologu do MariaDB, využitím dalšího UCB bloku. Naopak druhý způsob se zaměřoval na funkcionality právě MariaDB serveru, a to přesněji Triggers a Schedulers.

Nakonec jsem se rozhodl i přes později zjištěné omezení využít právě funkcionalit MariaDB databáze. Tímto omezením je v našem případě myšleno nemožnost využít

Schedulers nastavení databáze, jelikož tato možnost je továrně zakázána společností B&R Automation. I přes tohle omezení se mi naplnění tabulek daty povedlo. Důvodem výběru právě této metody bylo především jednoduší správa v případě dalších úprav, jelikož úpravy se prování přímo v databázi a není nutné využívat systém APROL a složitějších scriptů.

Triggers funguji podobně jako klasické SQL příkazy, ale rozdíl je že se provádí v případě definovaných událostí (INSERT, DELETE, UPDATE). Triggers se definují v rámci tabulek v databázi a mohou přistupovat k datům před nebo po dané události. V závislosti na platformě triggers mohou přistupovat ke všem datům v tabulce nebo jen k danému řádku, v mém případě MariaDB umožňuje pouze řádkový přístup.

Pro naplnění před vytvořených tabulek pro výpočet OEE jsem implementoval Triggers do většiny tabulek. Tabulka kde tato funkcionalita není vložená je Alarm, jelikož data v ní obsažená nepotřebujeme pro výpočet. Na následující stránkách popíši jednotlivé funkce pro dané tabulky.

### Device

Tento implementovaný Trigger v tabulce Device způsobí že pokaždé když se nastaví stroj pro danou zakázku nebo se přenastaví vyrábějící se produkt, tak se vytvoří nový řádek v tabulce *OEE_contract*. Tento zápis je velice důležitý, jelikož kdyby k němu nedošlo, funkčnost ostatních Triggers bude znemožněna, jelikož se jedná o inicializační zápis do tabulky. Data vložená do tabulky jsou název zařízení, označení zakázky a produktu, ideální takt stroje a počet vyráběných výrobků. Ostatní hodnoty jsou inicializačně nastaveny na nulovou hodnotu. Vše tohle můžete vidět ve výpisu 5.1.

Výpis 5.1: Trigger v tabulce Device

```
1  CREATE TRIGGER `BUR_PDA`.`xczips00_PDA-Device_TRIGGER`
2  AFTER INSERT ON `xczips00_PDA-Device` FOR EACH ROW
3
4  BEGIN
5  /* ------------ Contract ------------ */
6  INSERT INTO BUR_PDA.OEE_contract(ts,equip_id,contract,
7    product,part,nok,ts_complete,ts_production,ts_downtime,
8    ideal_cyklus,num_product,availability,performance,
9    quality,OEE)
10 VALUES(new.ts_start,new.equip_id,new.contract,
11   new.product,0,0,0,0,0,new.ideal_cyklus,
12   new.num_product,0.0,0.0,0.0,0.0);
13 END
```

### Produkce

Tabulka Produkce obsahuje Trigger jenž spolupracuje s oběma před vytvořenými tabulkami. Trigger nevytváří nové řádky, ale upravuje hodnoty již vytvořených triggrem v tabulce Device. Informaci, jaký řádek tabulky má editovat je dáno podmínkou WHERE. V případě tabulky *OEE_contract* se provede editace hodnoty part. Editace spočívá v součtu vyrobených kusů nově přibylého řádku s již existující hodnotou, ale jen pro daný stroj, zakázku a produkt. Tím se docílí, že na konci výroby budeme znát přesný počet vyrobených kusů pro jednotlivé zakázky nebo produkty. U tabulky *OEE_device* se provádí identická editace, ale jen pro daný den a stroj. Výše popsaná editace je zobrazena na výpisu 5.2.

Výpis 5.2: Trigger v tabulce Produkce

```
CREATE TRIGGER `BUR_PDA`.`xczips00_PDA-Produkce_TRIGGER`
AFTER INSERT ON `xczips00_PDA-Produkce` FOR EACH ROW

BEGIN
/* ------------ Contract ------------ */
UPDATE BUR_PDA.OEE_contract t1
SET t1.part = t1.part + NEW.part_count
WHERE t1.contract = NEW.contract AND
  t1.product = NEW.product AND
  t1.equip_id = NEW.equip_id;

/* ------------ Device ------------ */
UPDATE BUR_PDA.OEE_device t1
SET t1.part = t1.part + NEW.part_count
WHERE t1.date = cast(NEW.ts as DATE) AND
  t1.equip_id = NEW.equip_id;
END
```

### NOK

Podobně jak je to u tabulky Produkce, tak funguje Trigger v tabulce NOK ukázaný ve výpisu 5.3. Na rozdíl od produkce, u které přibývají nové řádky každým cyklem zápisu, tak zápis pro nekvalitní kusy nemusí vůbec proběhnout. V takovém případě v před vytvořených tabulkách zůstává výchozí proměnná tedy nulová.

Výpis 5.3: Trigger v tabulce NOK

```
CREATE TRIGGER `BUR_PDA`.`xczips00_PDA-NOK_TRIGGER`
```

```
2  AFTER INSERT ON `xczips00_PDA-NOK` FOR EACH ROW
3
4  BEGIN
5  /* ------------ Contract ------------ */
6  UPDATE BUR_PDA.OEE_contract t1
7  SET t1.nok = t1.nok + NEW.nok_number
8  WHERE t1.contract = NEW.contract AND
9    t1.product = NEW.product AND
10   t1.equip_id = NEW.equip_id;
11
12 /* ------------ Device ------------ */
13 UPDATE BUR_PDA.OEE_device t1
14 SET t1.nok = t1.nok + NEW.nok_number
15 WHERE t1.date = cast(NEW.ts as DATE) AND
16   t1.equip_id = NEW.equip_id;
17 END
```

### Stav

Trigger v tabulce Stav je nejrozsáhlejší Trigger v databázi, pro jednoduší popis jsem se rozhodl kód rozdělit na dvě části. Část pro tabulku *OEE_contract* a druhou část pro tabulku *OEE_device*. Také pro zkrácení výpisu jsem zredukoval podobný kód a nechal jsem na ukázku jen jednu jeho variantu.

V případě příchodu nového zápisu stavu do tabulky Stav se tento zápis roztřídí do potřebných sloupců tabulky *OEE_contract* pomocí podmínek IF. Na výpisu 5.4 můžeme vidět ukázku tohoto rozřazení pod komentářem "Ukladani casu". Do proměnných *ts_complete*, *ts_production* a *ts_downtime* se ukládají definované stavy v milisekundách. Pokud bychom chtěli provést výpočet OEE bez času nastavování stroje je potřeba pozměnit kód daného Trigger. Jakmile máme takto roztříděné časy jednotlivých stavů můžeme přejít k výpočtu indexů.

Pro výpočet indexů jsem prvotně chtěl použít Scheduler možnost databáze, ale jak jsem již zmínil tato možnost není běžně k dispozici. Proto jsem se rozhodl použít navazující Trigger aby jednotlivé editace probíhaly sériově, bohužel ani tahle možnost nefungovala podle mého plánu, proto jsem upřednostnil aktuální řešení. Výpočet OEE indexů probíhá automaticky ve stejném Triggeru jako rozřaďování stavů, ale jen pod podmínkou že daný stav je určen pro zápis dat. Nevýhodou tohoto postupu je že výpočet indexů je vázán podmínkou jen pro shodné zařízení, proto může tento výpočet v případě obsáhlejších databází zabrat určitý čas.

Výpis 5.4: Trigger v tabulce Stav pro Contract

```
/* Ukladani casu */
IF NEW.state = 30 OR NEW.state = 40 OR
  NEW.state = 50 OR
  NEW.state = 60 THEN
UPDATE BUR_PDA.OEE_contract t1
SET t1.ts_complete = t1.ts_complete + NEW.ts_duration
WHERE t1.contract = NEW.contract AND
  t1.product = NEW.product AND
  t1.equip_id = NEW.equip_id;
END IF;

/* stejne pro stav 30 a 40 */
/* ... */

/* Vypocet ukazatelu */
IF NEW.state = 0 THEN
UPDATE BUR_PDA.OEE_contract t1
SET
t1.availability =
(t1.ts_production / t1.ts_complete) * 100,
t1.performance =
  ((t1.ideal_cyklus * t1.part) / t1.ts_production) * 100,
t1.quality =
  ((t1.part - t1.nok) / t1.num_product) * 100,
t1.OEE =
  (t1.availability * t1.performance * t1.quality) / 10000
WHERE t1.equip_id = NEW.equip_id;
END IF;
```

V případě druhé části kódu Trigger pro tabulku *OEE_device* se musí vytvořit potřebný záznam pro daný den. To je docíleno částí kódu pod komentářem "Pridani radku"viz výpis 5.4, který zjisti, jestli existuje zápis pro daný stroj v daném dni a pokud ano nic neprovádí. V opačném případě přidá do tabulky nový řádek s daným datumem a názvem daného stroj a opět zbylé hodnoty nastaví na nulové. Jedná se o podobnou inicializaci jako tomu je u Trigger v tabulce *Device*. Jakmile je tahle podmínka vyhodnocena postupuje se do části programu, kde probíhá podobné třídění času jak tomu bylo v části kódu pro Contract. Rozdíl je především v tom že zde máme dané časy rozděleny podrobněji, a to na *ts_maintenance*, *ts_setting*,

*ts_production*, *ts_disorder* a *ts_break*. S těmito časy následně pracuji podle potřeby pro výpočet celkového OEE.

Většina indexů v této části kódu se počítají jiným způsobem než tomu bylo v případě OEE pro jednotlivé zakázky. Hlavní rozdíl je ve výpočtu performance dané linky, který se počítá jako aritmetický průměr všech performance v tabulce *OEE_contract*. Tento postup je nutný, jelikož každý stroj nebo produkt může mít jiný ideální takt stroje a tím by byl klasický výpočet nemožný. Další rozdíl je v případě indexu Availability, který se již nepočítá jako pomocí celkového času výroby ale z celkového času směny. Zbylé indexy se počítají stejným způsobem.

Výpis 5.5: Trigger v tabulce Stav pro Device

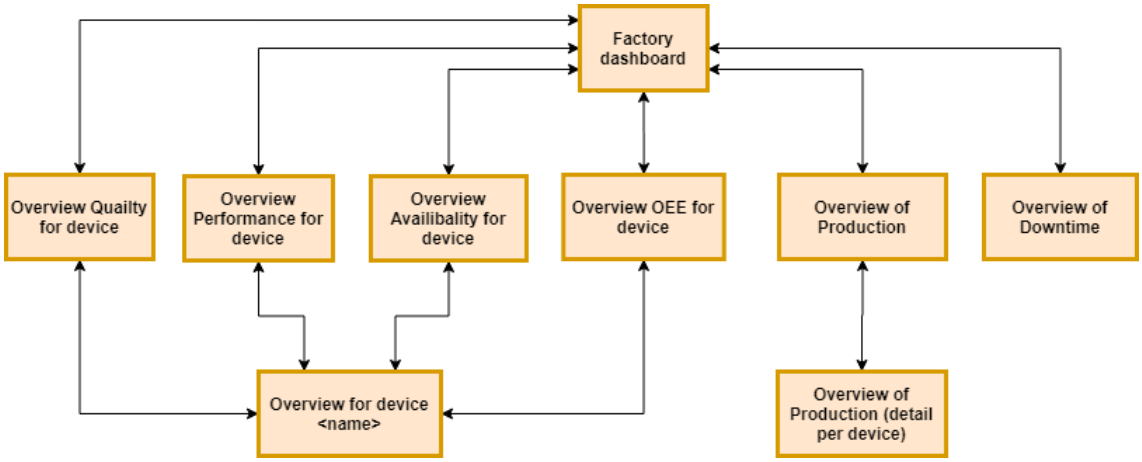
```
/* Pridani radku  */
IF (NOT EXISTS(SELECT 1 FROM BUR_PDA.OEE_device
    WHERE BUR_PDA.OEE_device.date = CAST(NEW.ts AS DATE)
    AND BUR_PDA.OEE_device.equip_id = NEW.equip_id))
THEN
INSERT INTO BUR_PDA.OEE_device(date,equip_id,part,nok,
  ts_maintenance,ts_production,ts_setting,ts_disorder,
  ts_break,availability,performance,quality,OEE)
VALUES(CAST(new.ts AS DATE),new.equip_id,0,0,0,0,0,0,0,
  0.0,0.0,0.0,0.0);
END IF;

/* Ukladani casu */
/* Podobny zapis jako v pripade Contract*/
/* ... */

/* Vypocet ukazatelu  */
IF NEW.state = 0 THEN
UPDATE BUR_PDA.OEE_device t1
LEFT JOIN (SELECT cast(ts as DATE) AS ts_new,equip_id,
  AVG(performance) AS performance_avg
FROM BUR_PDA.OEE_contract
GROUP BY ts_new,equip_id) t2
ON t1.date = t2.ts_new AND t1.equip_id = t2.equip_id
SET t1.performance = t2.performance_avg
WHERE t1.date = CAST(NEW.ts AS DATE) AND
  t1.equip_id = NEW.equip_id;

```

```
29 UPDATE BUR_PDA.OEE_device t1
30 SET t1.availability =
31   (t1.ts_production / 32400000) * 100,
32 t1.quality = ((t1.part - t1.nok) / t1.part) * 100,
33 t1.OEE =
34   (t1.availability * t1.performance * t1.quality) / 10000
35 WHERE t1.date = CAST(NEW.ts AS DATE) AND
36   t1.equip_id = NEW.equip_id;
37 END IF;
```

Takto vypracované indexy se mohou kdykoliv použít v jakémkoliv vyhodnocovací programu. V našem případě pro tuto činnost využívám SW od společnosti Jaspersoft, ale klidně bych mohl použít UCB blok pro použití potřebných dat přímo v APROL systému nebo přes PLC přímo do HMI panelu u linky.

# 6 Zobrazení dat

Poslední částí mé diplomové práce je získaná data zobrazit pomocí dashboardů, to jsem docílil pomocí portfolia programů společnosti Jaspersoft. Více o programech, které jsem k tomu použil v kapitole 3.5. Vzhled vytvořených dashboardů jsem konzultoval hlavně s odborníky ze společnosti B&R Automation, kde výstupem byl list nejčastěji požadovaných zobrazovaných informací. S těmito informacemi jsem se nadále snažil pracovat a vytvořil jsem systém dashboardů, které jsou navzájem provázané. Celý systém provázaní dashboardů je možno vidět na obrázku 6.1, jedná se o systém, ve kterém je možné z dashboardu týkajícího se výrobního závodu prostupovat dolů, až k požadovanému stoji a nazpátek. To samozřejmě neznamená že jednotlivé dashboardy se nedají použít samostatně, vše záleží na nastavení serveru. Všechny vytvořené dashboardy jsou možné k nahlédnutí v příloze C.




Obr. 6.1: Systém provázání dashboardů

V APROL systému se dané dashboardy nejčastěji zobrazují pomocí webového prohlížeče. Jejich zobrazení ale není vázáno na dané APC, pokud je APC, přesněji JasperReport Server, připojeno k nějaké vnitřní síti dají se na této síti zobrazit dashboardy na jakýmkoliv zařízení který má webový prohlížeč. V této práci byl pro zobrazení dashboardů v systému APROL využit program DisplayCenter, což je základní vizualizační prostředí systému APROL. Způsob zobrazení těchto Dashboardů a dalších dat v DisplayCentra je v příloze B.

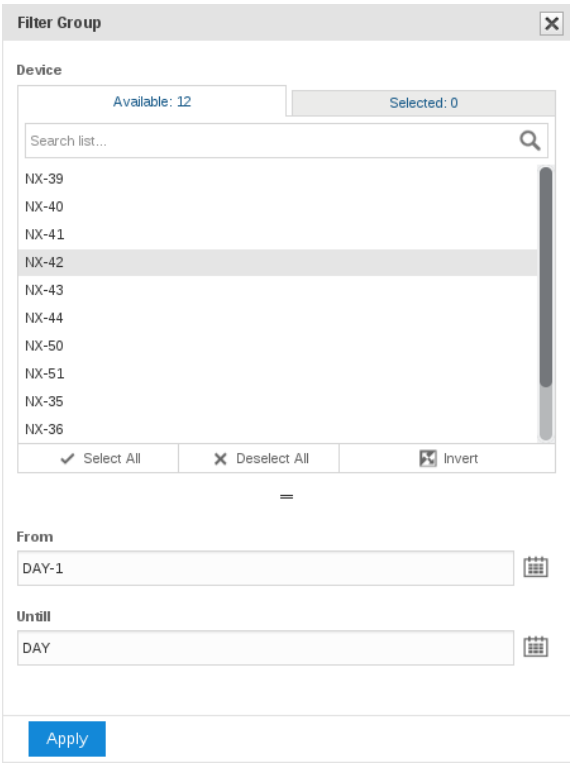
## 6.1 Parametry

Jedním důležitým prvkem Jaspersoft programů jsou správně nastavené parametry. Díky těmto parametrům je možná provázat jednotlivé dashboardy a předávat si mezi nimi informace. Tyto parametry také umožnuji efektivní filtraci a umožňují provést

inicializační nastavení dashboardu. Využití parametrů můžete vidět ve formě popup okna filtru na obrázku 6.2.

Ve svém projektu využívám několik parametrů, které kombinuji do speciálních SQL příkazů typických pro Jaspersoft. První z nich je **$X{[BETWEEN], <název tabulky>.<proměnná> ,p_from, p_till}**. Tento příkaz funguje stejně jako funkce BETWEEN v normálním SQL dotazu, jen s tím rozdílem že parametry p_from a p_till jsou datového typu TimestampRange, který umožnuje nastavení defaultní hodnoty. To znamená že při každém puštění dashboardu není potřeba zadávat časové rozmezí pro které chceme vidět data. Tato činnost se provede automaticky na aktuální den.

Druhý z příkazů je **$X{IN, <název tabulky>.<proměnná>,p_equip_id}**, který umožnuje multiple select dotaz. To znamená že parametr p_equip_id, datového typy Collection, obsahuje vektor proměnných odkazující se existující hodnoty v dané tabulce. Pokud daný vektor je prázdný automaticky se bere že jsou vybrány všechny možné hodnoty.



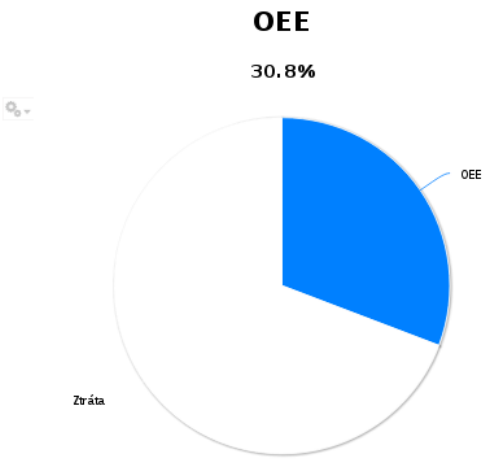

Obr. 6.2: Popup filter pro dashboard

## 6.2 Jednotlivé prvky dashboardu

V této podkapitole si přiblížíme některé z dílčích prvků použitých dashboardů, a to z toho důvodu, jelikož potřebovali uzpůsobit SQL dotaz pro jejich funkčnost. Pro ostatní nezmíněné prvky jsou následně použity standartní dotazy pro jejich funkčnost nebo podobné již popsaným.

### Procentní zobrazovač indexů

Vybral jsem tento způsob grafického zobrazení získaných indexů, jako jednu z nejlepších možností. Jedná se o klasický HTML5 chart z přidaných číselným zobrazením daného indexu a nadpisem. Výhodou využití tohoto prvku pro zobrazení je možná následná editace vzhledu pomocí ikonky vlevo nahoře. Takto může operátor jednoduše změnit zobrazení mezi Pie chartem, Semi-pie chartem nebo klasickým Columm zobrazením.




Obr. 6.3: Ukázka grafického procentního zobrazovače indexů

HTML5 chart není primárně určený pro zobrazování jedné proměnné způsobem zobrazeným na obrázku 6.3. Proto bylo potřeba pozměnit SQL dotaz. Pro funkčnost bylo potřeba přidat nejméně jednu další proměnnou. To jsem docílil sloučením dvou dotazů pomocí funkce UNION. V prvním dotazu se ptám na průměrnou hodnotu OEE s danými parametry a ve druhém dotazu se ptám na průměrnou hodnotu rozdílu hodnoty OEE od její maximální hodnoty. Tím docílím že výsledná tabulka bude mít potřebné dvě hodnoty pro správnou funkčnost chartu, ukázka je na výpisu 6.1.

Stejný způsob využití tohoto HTML5 chartu následně uplatňuji na všechny dalších grafické zobrazovače indexů v dashboardech, i když se od sebe mohou na první

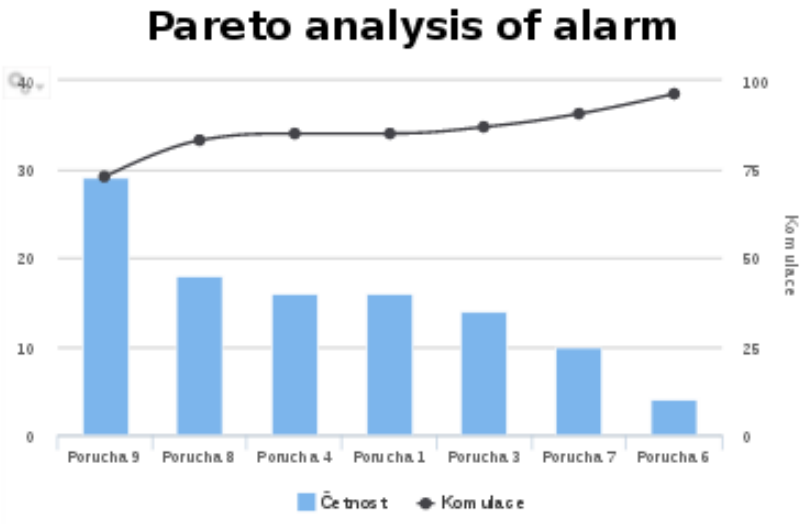
pohled lišit.

Výpis 6.1: SQL dotaz pro grafické zobrzení indexu OEE

```
SELECT  `BUR_PDA`.`OEE_device`.date,
    'OEE' as id,
    AVG( `BUR_PDA`.`OEE_device`.`OEE` ) AS vypocet
FROM `BUR_PDA`.`OEE_device`
WHERE $X{[BETWEEN],`BUR_PDA`.`OEE_device`.date
      ,p_from,p_till}
  AND $X{IN, `BUR_PDA`.`OEE_device`.equip_id,equip_id }
UNION ALL
SELECT  `BUR_PDA`.`OEE_device`.date,
    'Ztráta' as id,
AVG(100 -  `BUR_PDA`.`OEE_device`.`OEE` ) AS vypocet
FROM  `BUR_PDA`.`OEE_device`
WHERE $X{[BETWEEN],`BUR_PDA`.`OEE_device`.date
      ,p_from,p_till}
  AND $X{IN, `BUR_PDA`.`OEE_device`.equip_id,equip_id };
```

### Graf Paretovy analýzy

Pro zobrazení Paretovy analýzy využívám také HTML5 charty, přesněji Multi-Axis chart Columm Spline, kde pro správné zobrazení je potřeba si pohrát v jeho nastavení, ukázka na obrázku 6.4. Ale jeho hlavní část tkví k SQL dotazu.




Obr. 6.4: Ukázka grafického zobrazení Paretovy analýzy

V tomto dotazu využívám funkce INNER JOIN, který mi umožní sloučení dvou tabulek, viz. výpis 6.2. Jedná se o tabulku Alarm, ve které vyhodnocuji data a tabulky Def_Alarm, která obsahuje potřebné popisy. Následně sčítám nastalé alarmy po jednotlivých skupinách pro jejich zobrazení. Problém nastává až v případě četnosti, kde bylo potřeba provést rozsáhlého výpočtu a tím získaná data převést na procenta. Tento způsob vyšel jako jediný funkční z variant které jsem zkoušel v rámci SW Jaspersoft.

Výpis 6.2: SQL dotaz pro grafické zobrzení Paretovy analýzy

```
SELECT `BUR_PDA`.`xczips00_PDA-Alarm`.err_type AS id,
  COUNT(`BUR_PDA`.`xczips00_PDA-Alarm`.err_type) AS pocet,
  (SELECT COUNT(*)
    FROM `BUR_PDA`.`xczips00_PDA-Alarm`
    where $X{[BETWEEN], `BUR_PDA`.`xczips00_PDA-Alarm`.ts
        ,p_from,p_till}
    and $X{IN, `BUR_PDA`.`xczips00_PDA-Alarm`.equip_id
        ,equip_id }) AS komplet,
/* Výpočet četnosti */
((((SELECT COUNT(*)
FROM `BUR_PDA`.`xczips00_PDA-Alarm`
WHERE $X{[BETWEEN], `BUR_PDA`.`xczips00_PDA-Alarm`.ts
    ,p_from,p_till}
AND $X{IN, `BUR_PDA`.`xczips00_PDA-Alarm`.equip_id
    ,equip_id })
- COUNT( `BUR_PDA`.`xczips00_PDA-Alarm`.err_type))
/  (SELECT COUNT(*)
FROM `BUR_PDA`.`xczips00_PDA-Alarm`
WHERE $X{[BETWEEN], `BUR_PDA`.`xczips00_PDA-Alarm`.ts
    ,p_from,p_till}
AND $X{IN, `BUR_PDA`.`xczips00_PDA-Alarm`.equip_id
    ,equip_id }))
* 100) AS vypocet,

`BUR_PDA`.`Def_Alarm`.description AS popis
FROM `BUR_PDA`.`xczips00_PDA-Alarm`
INNER JOIN `BUR_PDA`.`Def_Alarm`
ON `BUR_PDA`.`Def_Alarm`.err_type
    = `BUR_PDA`.`xczips00_PDA-Alarm`.err_type
WHERE $X{[BETWEEN], `BUR_PDA`.`xczips00_PDA-Alarm`.ts
```

```
31      ,p_from,p_till}
32  AND $X{IN, `BUR_PDA`.`xczips00_PDA-Alarm`.equip_id
33      ,equip_id }
34  GROUP BY `BUR_PDA`.`xczips00_PDA-Alarm`.err_type
35  ORDER BY pocet DESC
```

## 6.3 Denní reporty

Pokud pomineme dashboardy, Jaspersoft umožnuje tvorbu vlastních reportů na míry. Tyto reporty fungují na velice podobném principu jako dashboardy, ale na rozdíl od dashboardu udržují pevné rozměry prvků a přesný formát předlohy. Nejčastějším formátem předlohy je stránka A4, ale zvládá většinu běžných formátů. Reporty fungují na řádkovém provádění SQL dotazů, takže celé kouzlo tkví v nakonfigurování vždy jen hlavičky a prvního řádku tabulky, kde o zbytek se postará program. Tato funkcionalita je dokonalá v případě rozsáhlých výpisů z databáze. Pokud chceme kombinovat více než jednu tabulku databáze do jednoho reportu je potřeba využít subreportů.

Jako příklad jsem vytvořil vlastní denní report simulovaných linek, který provede výpis indexů daného závodu a jednotlivých stojů. U každého stroje zobrazí ten den prováděné zakázky s počtem vyrobených kusů, nekvalitních kusů, celkového plánu a vypočítaných indexů (provede přepis tabulky OEE_contract). Také zobrazí, o jaké přesné nekvalitní kusy se jednalo. Takto vytvořený report se dá následně zobrazit v prohlížeči nebo pomocí Scheduler nastavení každý den posílat zodpovědným osobám jako PDF. Tento denní report je obsažen v přiloženém DVD viz, příloha D.

# 7 Závěr

Hlavním cílem této práce bylo vytvořit metody pro sběr a vyhodnocování dat z výroby pomocí softwaru společnosti B&R Automation.

Před samotnou realizací jsem podstoupil řadu komerčních firemních školení, kde jsem se seznámil se základy ovládání SW APROL a Automation Studio od společnosti B&R Automation. Jelikož se jednalo o komerční školení část z nich byla zrušena z důvodu malé účasti, proto mi byli poskytnuty veškeré školící materiály a zrušené školení proběhlo formou samostudia a konzultací.

V rámci proškolovaní jsem vypracovával rešerši sběru dat a KPI s důrazem na OEE. Své poznatky jsem následně v rámci veletrhů a konference konzultoval s praktickými odborníky a zástupci firem v daném oboru. Výstupu tohoto jednání se následně věnuji v kapitolách 1.3 a 2.

Jakmile jsem měl dostatečnou znalost SW společnosti B&R Automation začal jsem pracovat na simulátoru výrobní linky, kterému se věnuji v kapitole 4. Důvodem tvorby tohoto simulátoru byla potřeba proměnlivých dat pro ověření funkčnosti sběru a následného vyhodnocení. To vše samozřejmě nejen pro jednu linku ale vícero linek. Zde jsem se setkal s první překážkou a to především jak naprogramovat PLC aby fungovalo co nejvíc automaticky, ale co nejméně cyklicky. Tento problém jsem vyřešil vlastní jednoduchou random funkcí na které stojí celý simulátor. V rámci simulátoru jsem vytvořil datovou strukturu a vybral způsob komunikace se systémem APROL.

S hotovým simulátorem jsem mohl začít pracovat na sběru dat z výroby, jejich předzpracování a uložení. Rozhodl jsem se pro využití dvou databází pro splnění tohoto úkolu. První interní databáze slouží ke krátkodobému uložení a záloze dat, před přenesením do externí databáze se kterou pak pracují vizualizační SW. V rámci této činnosti jsem v systému APROL vytvořil knihovnu CFC bloků, které k tomu slouží. Přenesením dat do externí databáze má práce s daty neskončila, pro efektivnější práci jsem se rozhodl znovu předzpracovat a tím databázi optimalizovat. Vším tímto a pár dalšími věcmi se zabývám v kapitole 5.

V poslední části své práci jsem vytvořil několik dashboardů pro zobrazení OEE, podrobnější popis lze najít v kapitole 6. Tyto dashboardy jsem mezi sebou provázal takovým způsobem, že si mezi sebou předávají potřebné informace formou parametrů, která následně ovlivňuji prováděné SQL dotazy v databázi.

Celý tento systém sběru a vyhodnocování dat byl průběžně testován a prezentován společnosti B&R Automation jako výstup mé práce, čímž jsem splnil všechny body zadání své práce, zabývající se vyhodnocením efektivnosti výroby. Do budoucna by se dalo přemýšlet o zpřístupnění Scheduling nastavení v MariaDB a tím dosáhnout další možnosti zpracování dat.

# Literatura


[1] Štrublíková, I.: *MES systémy ve strojírenství – část 1.* Dostupné na URL: <http://www.mescentrum.cz/clanky/mes-mom/131-mes-systemy-ve-strojirenstvi-cast-1>

[2] Pásek, J., Braun, V.: *Automatizace procesů II – Úroveň řízení výroby.*

[3] *ISA-95 levels.* Dostupné na URL: <https://www.drouiz.com/blog/2014/12/04/niveles-isa-95-levels/>

[4] Světlík, V.: *MES (Manufacturing Execution Systems).* Dostupné na URL: <https://www.systemonline.cz/clanky/mes-manufacturing-execution-systems.htm>

[5] *Co je MES – Výrobní informační systém.* Dostupné na URL: <http://www.mescentrum.cz/o-projektu/co-mes>

[6] Anderson, M.: *What is DCS? (Distributed Control System).* Dostupné na URL: <https://realpars.com/dcs/>

[7] Kula, J.: *Distribuované řídicí systémy a automatizace v chemickém průmyslu.* Dostupné na URL: <http://www.automa.cz/Aton/FileRepository/pdf_articles/9661.pdf>

[8] Kula, J.: *Datový sklad se zaměřením na optimalizaci ETL procesu.* Diplomová práce, Brno, FIT VUT v Brně, 2011, Dostupné na URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117917>

[9] *Hlavní principy datových skladů a proces jejich vytváření.* SystemOnline, Dostupné na URL: <https://www.systemonline.cz/clanky/hlavni-principy-datovych-skladu-a-proces-jejich-vytvareni.htm>

[10] *Datový management.* 2011, Dostupné na URL: <http://tecam.cz/media/cache/file/05/file000296.pdf>

[11] *Nasazení COMES OEE ve společnosti Medin a.s. OEE.* Dostupné na URL: <http://www.oee.cz/3a15c268_fb38_4cc6_91f8_a45fdd47294d.aspx>

[12] Johnsson, Ch., Kirsch. K.: *White Paper Introducing the IS0 22400 standard.*

[13] Patočka, M.: *OEE a odvozené ukazatele TEEP, PEE, OAE, OPE, OFE, OTE a CTE.* Dostupné na URL: <http://www.mescentrum.cz/clanky/mes-mom/133-oee>

[14] Strnad, M.: *Studie efektivnosti využití strojů ve vybraném provozu*. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=65699>

[15] *Calculate TEEP provozu* . Dostupné z URL: <https://www.oee.com/teep.html>

[16] *FAQ*. Dostupné z URL: <https://www.oee.com/faq.html>

[17] Kormanec, P.: *Výpočet OEE (Overall Equipment Effectiveness) – Celková efektívnosť zariadenia*. Dostupné z URL: <https://4industry.consulting/vypocet-oee-overall-equipment-effectivenes-celkova-efektivnost-zariadenia/>

[18] Holá, A.: *Analýza prostojů provozu*. Dostupné z URL: <https://dspace5.zcu.cz/bitstream/11025/25204/1/Diplomova%20prace___Analyza%20prostoju_Alzbeta%20Hola.pdf>

[19] Patočka, M.: *Hodnocení personálu dle OEE*. Dostupné na URL: <http://mescentrum.cz/clanky/mes-mom/171-hodnoceni-personalu-dle-oee>

[20] Durán, O., Capaldo, A. and Andrés Duran Acevedo, P.: *Sustainable Overall Throughputability Effectiveness (S.O.T.E.) as a Metric for Production Systems*. Dostupné na URL: <https://www.mdpi.com/2071-1050/10/2/362/htm>

[21] Stephen, F.: *Defining Failure: What Is MTTR, MTTF, and MTBF?*. Dostupné z URL: <https://blog.fosketts.net/2011/07/06/defining-failure-mttr-mttf-mtbf/>

[22] *Mean Time To Repair (MTTR)*. Dostupné z URL: <https://www.techopedia.com/definition/2719/mean-time-to-repair-mttr>

[23] *Mean Time Between Failures (MTBF)*. Dostupné z URL: <https://www.techopedia.com/definition/2718/mean-time-between-failures-mtbf>

[24] Zikmund, M.: *Paretova (ABC) analýza – mocný nástroj v logistice, marketingu i obchodu*. Dostupné z URL: <http://www.businessvize.cz/rizeni-a-optimalizace/paretova-abc-analyza-mocny-nastroj-v-logistice-marketingu-i-obchodu>

[25] Faron, J.: *Analýza, design a realizace dashboardu pro firemní použití*. Dostupné na URL: <https://is.muni.cz/th/ksx2h/diplomova-prace.pdf>

[26] Few, S.: *Information Dashboard Design: Displaying Data for At-aglance Monitoring*. Analytics Press, 2013.

[27] Few, S.: *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, Incorporated, 2006.

[28] Stanley, G.: *Three types of dashboards*. Dostupné na URL: <http://canworksmart.com/three-types-of-dashboards/>

[29] Rasmussen, N.H., Bansal, M., Chen, C.Y.: *Business Dashboards: A Visual Catalog for Design and Deployment. Wiley*. 2009.

[30] *Balanced Scorecard*. Dostupné na URL: <https://www.vlastnicesta.cz/metody/balanced-scorecard/>

[31] *TM210 - Working with Automation Studio*. Firemní cvičící material, 2019.

[32] *Automation Studio*. Dostupné na URL: <https://www.br-automation.com/cs/produkty/software/automation-studio/>

[33] *TM213 - Automation Runtime*. Firemní cvičící material, 2019.

[34] *TM800 - APROL System Concept*. Firemní cvičící material, 2019.

[35] *MariaDB*. Dostupné na URL: <https://mariadb.org>

[36] *MariaDB - About*. Dostupné na URL: <https://mariadb.org/about/>

[37] *MySQL Workbench*. Dostupné na URL: <https://www.mysql.com/products/workbench/>

[38] *JasperReports Server*. Dostupné na URL: <https://www.jaspersoft.com/products/jasperrepo server>

[39] *Jaspersoft Studio*. Dostupné na URL: <https://www.jaspersoft.com/products/jaspersoft-studio>

# Seznam symbolů, veličin a zkratek

**KPI** Key Performance Indicators
**OEE** Overall Equipment Efficiency
**TEEP** Total Equipment Effectiveness Performance
**SPC** Statistical Process Control
**PCS** Proces Control Systems
**PEE** Production Equipment Efficiency
**OFE** Overall Factory Effectiveness
**OTE** Overall Throughput Effectiveness
**CTE** Cycle Time Effectiveness
**ERP** Enterprise Resource Planning
**MES** Manufacturing Execution Systems
**PLC** Programmable Logic Controller
**DCS** Distributed Control System
**ISO** International Organization for Standardization
**MESA** Manufacturing Enterprise Solutions Association
**MAUP** Automatizace procesů
**SCADA** Supervisory Control And Data Acquisition
**BSC** Balanced Scorecard
**MBO** Management by Objectives
**TPM** Total Productive Maintenance
**PSE** Product Support Efficiency
**OU** Operating Utility
**MTBF** Mean Time Between Failure
**MTTR** Mean Time To Restoration
**SW** Software
**HMI** Human–Machine Interface
**ETL** Extract, Transform and Load
**TCP** Transmission Control Protocol
**IP** Internet Protokol
**TQM** Total quality management
**BI** Business Intelligence
**APC** Automation PC
**CFC** Continuous Function Chart
**PK** Primary Key
**DS** Data Storage

# Seznam příloh

# A Zbylé deklarace datových typů simulátoru

| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Command_typ | | | ☑ | | Struktura prikazu pro cinnost simulatoru, hlavne v manual rezimu |
| TaktPart | BOOL | ☐ | ☑ | FALSE | Realny takt pri vyrobeni jednoho kusu |
| LoginUser | BOOL | ☐ | ☑ | FALSE | Potvrzeni nalogovani uzivatele |
| Udrzba | BOOL | ☐ | ☑ | FALSE | Pozadavek pro udrzbu |
| Odstavka | BOOL | ☐ | ☑ | FALSE | Pozadavek pro odstavku |
| Vyroba | BOOL | ☐ | ☑ | FALSE | Pozadavek pro vyrobu |
| Nastaveno | BOOL | ☐ | ☑ | FALSE | Potvzeni ze je vse nastaveno pro vyrobu |
| Porucha | BOOL | ☐ | ☑ | FALSE | Pozadavek pro poruchu |
| Prestavka | BOOL | ☐ | ☑ | FALSE | Pozadavek pro prestavku |
| Preruseni | BOOL | ☐ | ☑ | FALSE | Pozadavek pro preruseni kroku v AUT |
| PlanPrestavka | BOOL | ☐ | ☑ | FALSE | Planovana prestavka v ramci smeny |
| Smena | BOOL | ☐ | ☑ | FALSE | Ukazatel ze je planovana smena |

Struktura povelů pro ovládání simulátoru

| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Krok_typ | | | | | Kroky simulace |
| Prihlaseni | | | | | 0 - Prichod operatara k lince a prihlaseni |
| Rozhodovani | | | | | 1 - Mezi krok pro rozhodnuti co se na stroji bude dit |
| Udrzba | | | | | 2 - Je provadena udrzba na stroji |
| Odstavka | | | | | 3 - Planovana odstavka stroje (neni zakazka, nebo nejsou splneny podminky pro jeho cinnost) |
| Nastavovani | | | | | 4 - Operator nastavuje stroj pro vyrobu |
| Vyroba | | | | | 5 - Stoj vyrabi vyrobky |
| Porucha | | | | | 6 - Nastala porucha stroje |
| Prestavka | | | | | 7 - Obsluha stroje jde na planovanou prestavku |
| Zapis | | | | | 8 - Rychli krok pro zápis dat |

Dílčí enum struktura stavů simulátoru

| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| MES_typ | | | ☑ | | Dana ziskana z MES |
| PartNumber | UDINT | ☐ | ☑ | 0 | Pocet kusu potrebnych pro splneni zakazky |
| IdealCyklus | UINT | ☐ | ☑ | 0 | Idealni cyklus troje v ms |
| Shift | UINT | ☐ | ☑ | 0 | Delka smeny |

Struktura informací z možné nadřazené vrstvy

| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Stav_typ | | | | | MAN/AUT stav |
| Manual | | | | | 0 - Simulace je ovladana manualne |
| Automatic | | | | | 1 - Simulace je ovladana automaticky |

Dílčí enum struktura chodu simulátoru

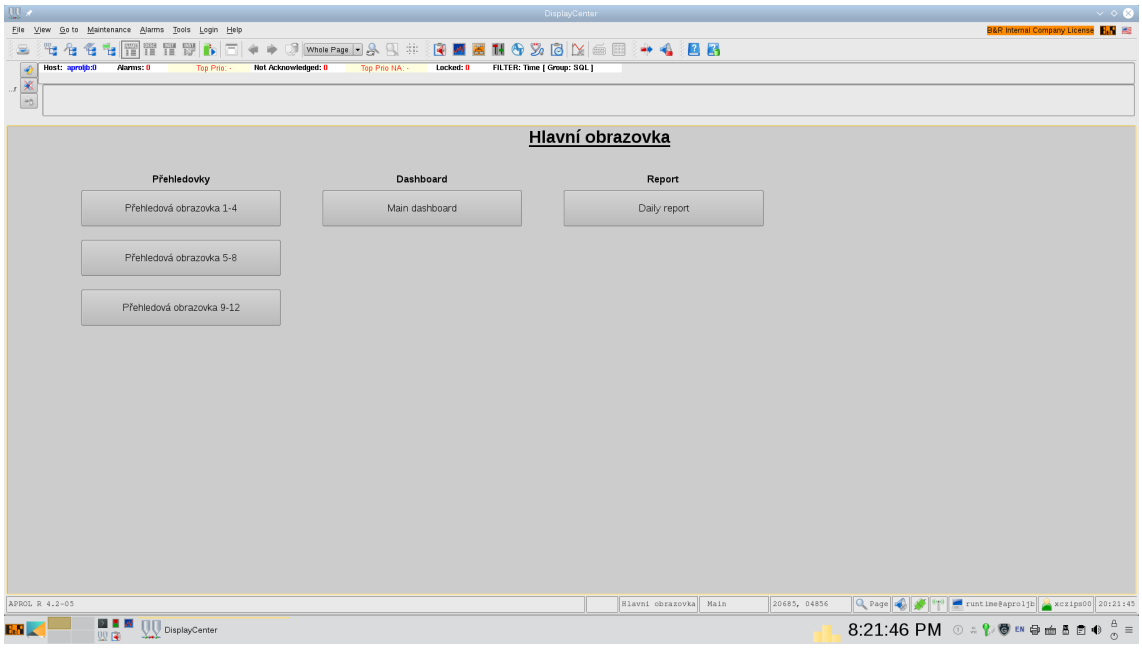
| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Technologie_typ | | | ☑ | | HLAVNI STRUKTURA |
| Aprol | Aprol_typ | ☐ | ☑ | | APROL |
| ReceivedFromMES | MES_typ | ☐ | ☑ | | MES |
| Command | Command_typ | ☐ | ☑ | | Prikazy |
| Status | Status_typ | ☐ | ☑ | | Ostatni |

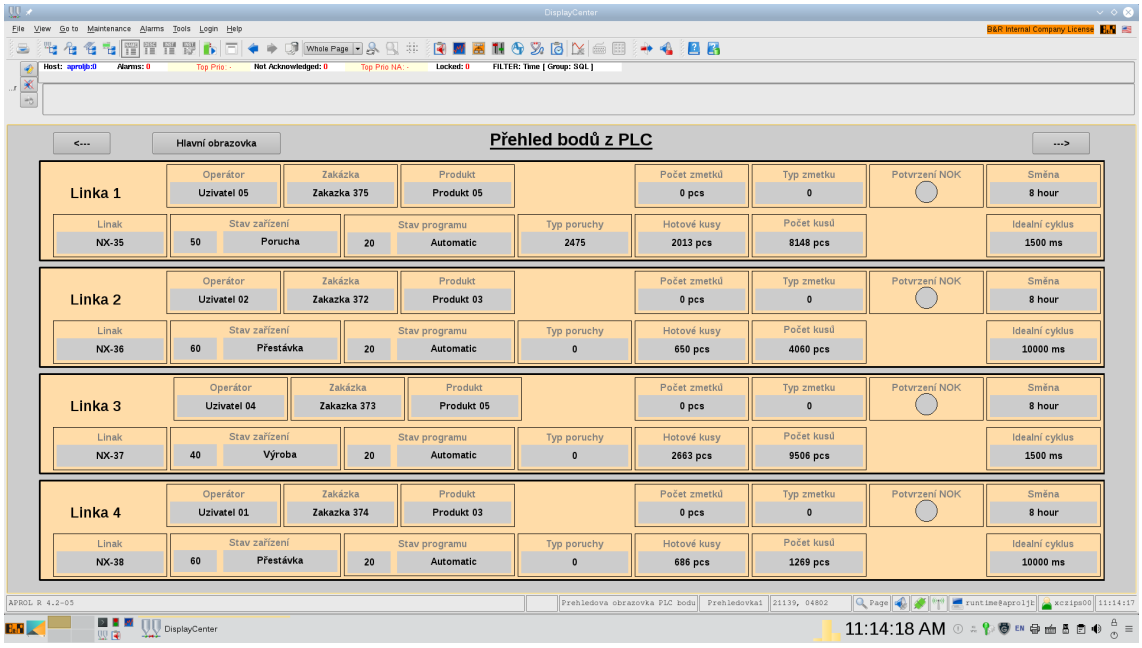
Konečné zapozdření všech datových struktur

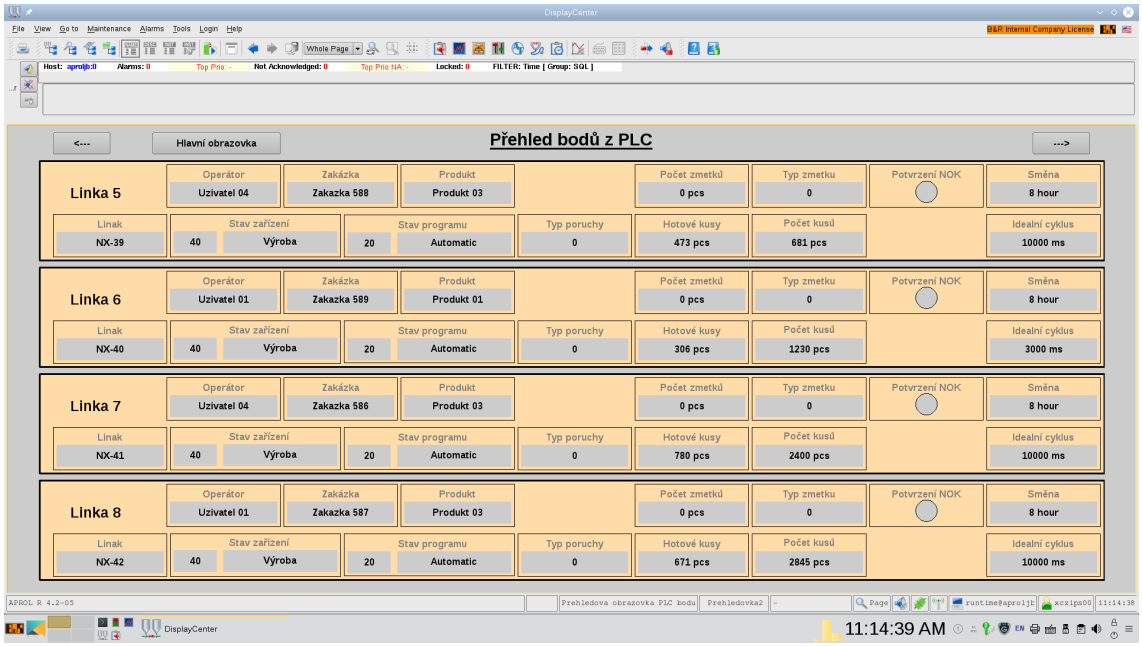
| Name | Type | Reference | Replicable | Value | Description [1] |
|---|---|---|---|---|---|
| Status_typ | | | ☑ | | Struktura promennych pro funkcnost simulatoru |
| Krok | Krok_typ | ☐ | ☑ | | Krok simulace |
| KrokNext | Krok_typ | ☐ | ☑ | | |
| Stav | Stav_typ | ☐ | ☑ | | Stav simulace MAN/AUT |
| RealCyklus | TIME | ☐ | ☑ | | Realny cyklus troje |
| DelkaKroku | TIME | ☐ | ☑ | T#0s | Delka trvani kroku pro jednotline situace |
| VykonVyroby | REAL | ☐ | ☑ | 1.0 | Promnena pro editacy idealniho cyklu stroje |
| RealCyklusReal | REAL | ☐ | ☑ | | Prevedeny idealni cyklus z TIME na REAL |
| StepTakt | USINT | ☐ | ☑ | | Promenna pro simulaci taktu stroje |
| RandomCislo | DINT | ☐ | ☑ | | Random cislo pro AUT rezim |
| Casovac | TON | ☐ | ☑ | | Casovac pro simulaci chodu v jednotlivych krocich |
| CasovacTakt | TON | ☐ | ☑ | | Casovac pro simulaci taktu |
| CitacPart | CTU | ☐ | ☑ | | Citac pocitajici vyrobene vyrobky |

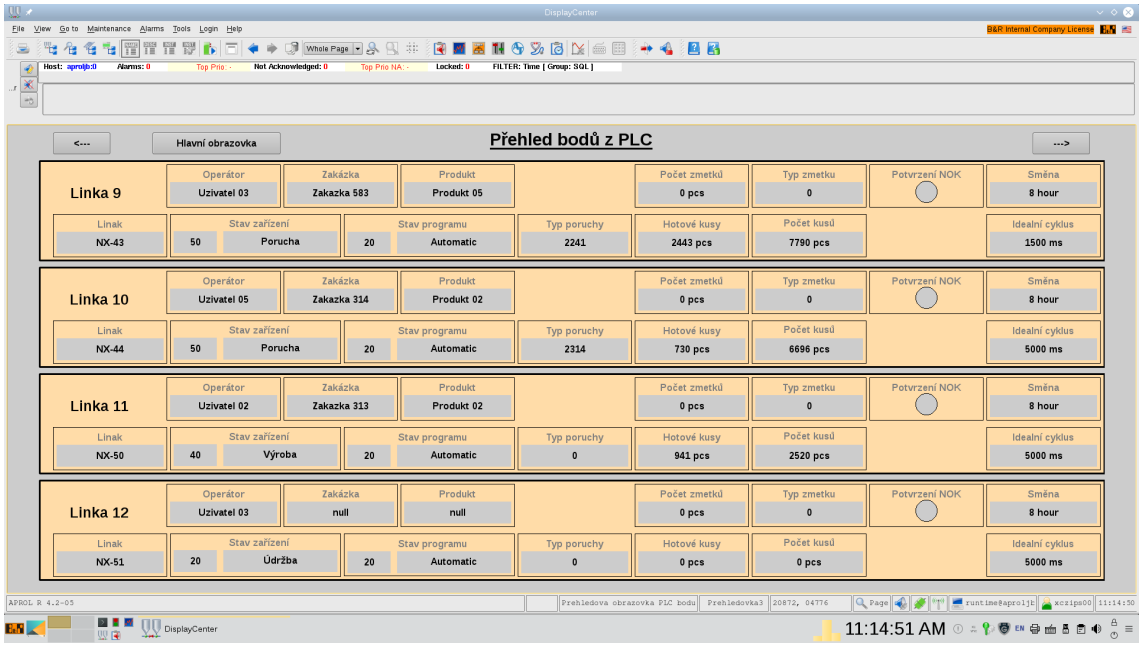
Struktura stavů simulátoru

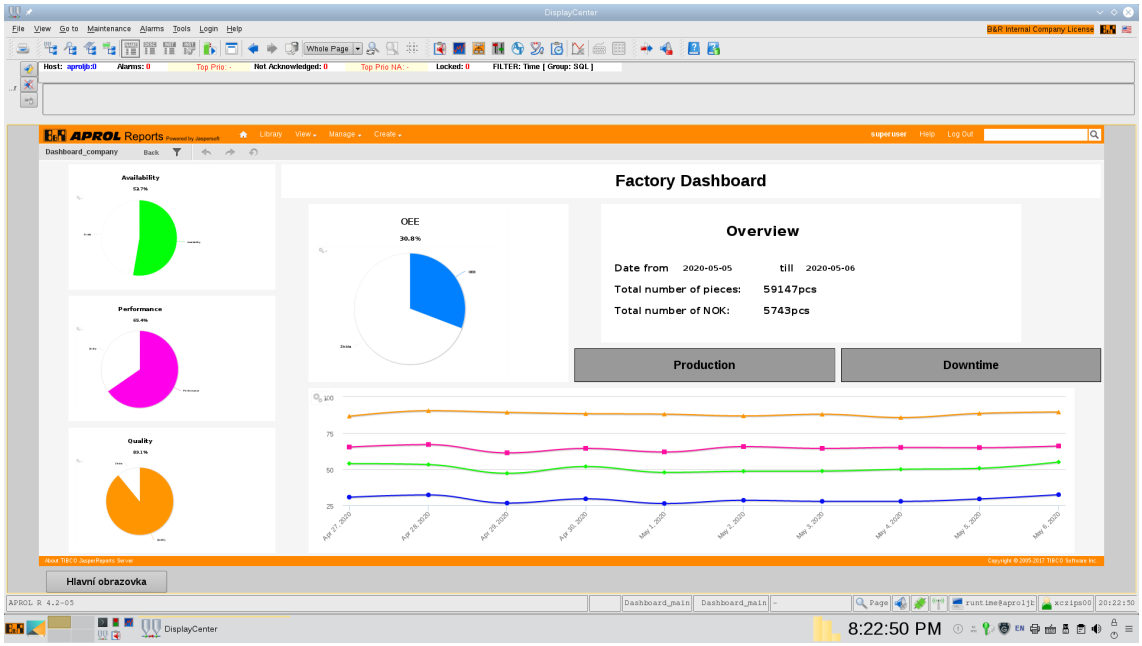
# B Vizualizace v APROL Runtime pomocí DisplayCentra

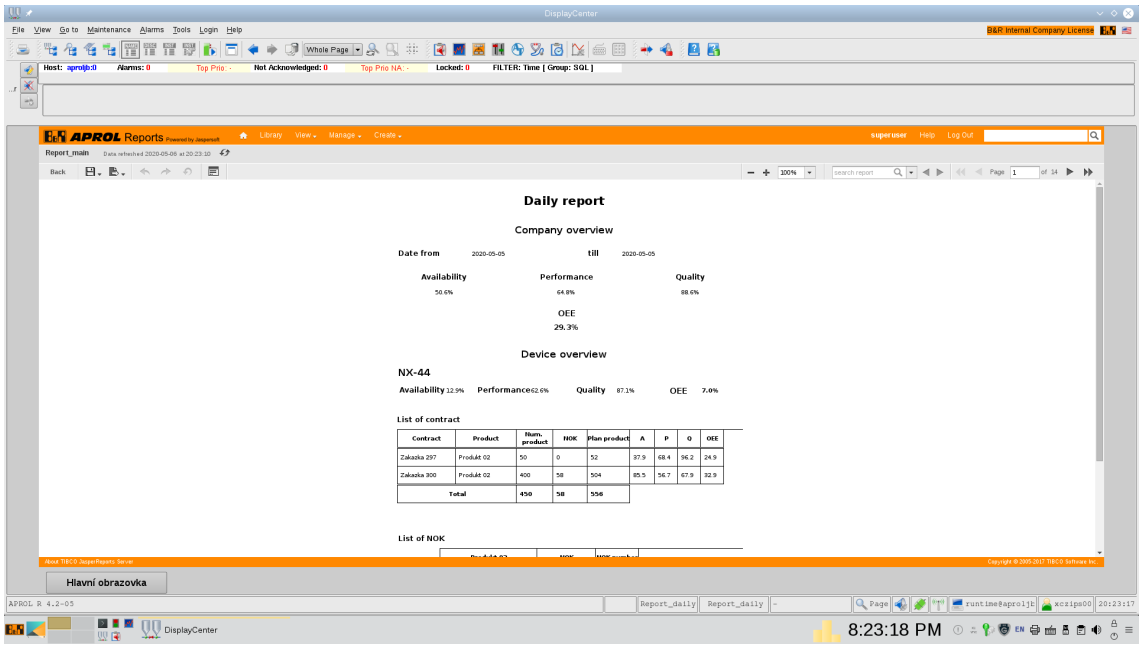


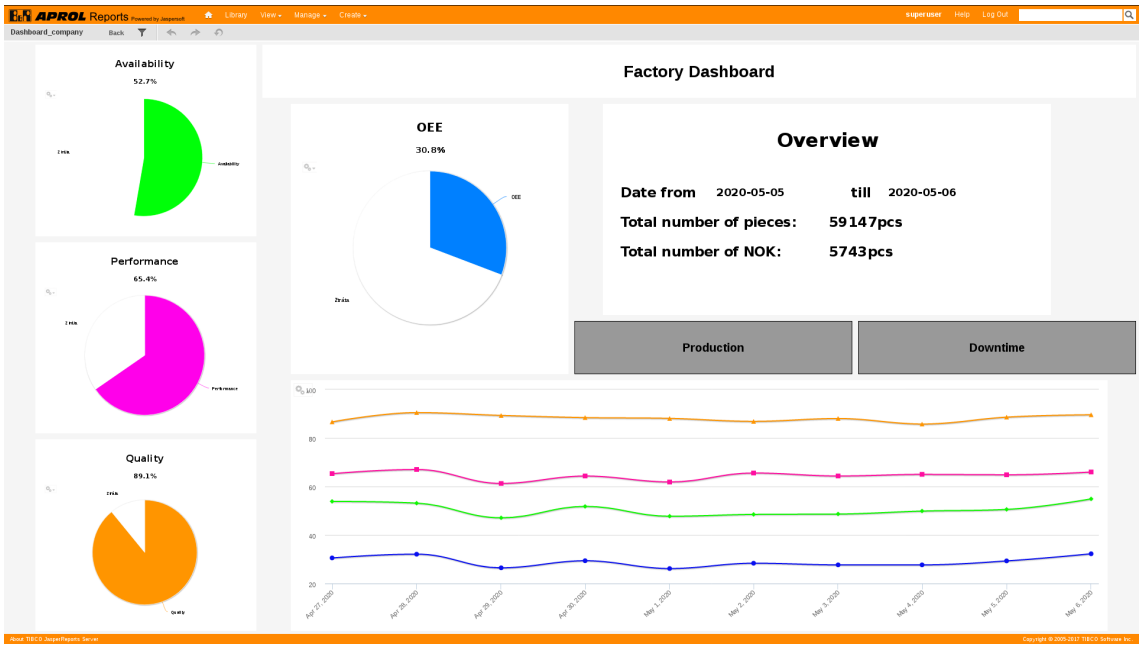

DisplayCenter
File View Go to Maintenance Alarms Tools Login Help
B&R Internal Company License
Host: aprolj0:0 Alarms: 0 Top Prio: - Not Acknowledged: 0 Top Prio NA: - Locked: 0 FILTER: Time [ Group: SQL ]
<---
Hlavní obrazovka
Přehled bodů z PLC
--->
Linka 5
Operátor Uzivatel 04
Zakázka Zakazka 588
Produkt Produkt 03
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-39
Stav zařízení 40 Výroba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 473 pcs
Počet kusů 681 pcs
Idealní cyklus 10000 ms
Linka 6
Operátor Uzivatel 01
Zakázka Zakazka 589
Produkt Produkt 01
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-40
Stav zařízení 40 Výroba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 306 pcs
Počet kusů 1230 pcs
Idealní cyklus 3000 ms
Linka 7
Operátor Uzivatel 04
Zakázka Zakazka 586
Produkt Produkt 03
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-41
Stav zařízení 40 Výroba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 780 pcs
Počet kusů 2400 pcs
Idealní cyklus 10000 ms
Linka 8
Operátor Uzivatel 01
Zakázka Zakazka 587
Produkt Produkt 03
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-42
Stav zařízení 40 Výroba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 671 pcs
Počet kusů 2845 pcs
Idealní cyklus 10000 ms
APROL R 4.2-05
Prehledova obrazovka PLC bodu
Prehledovka2
DisplayCenter
11:14:39 AM

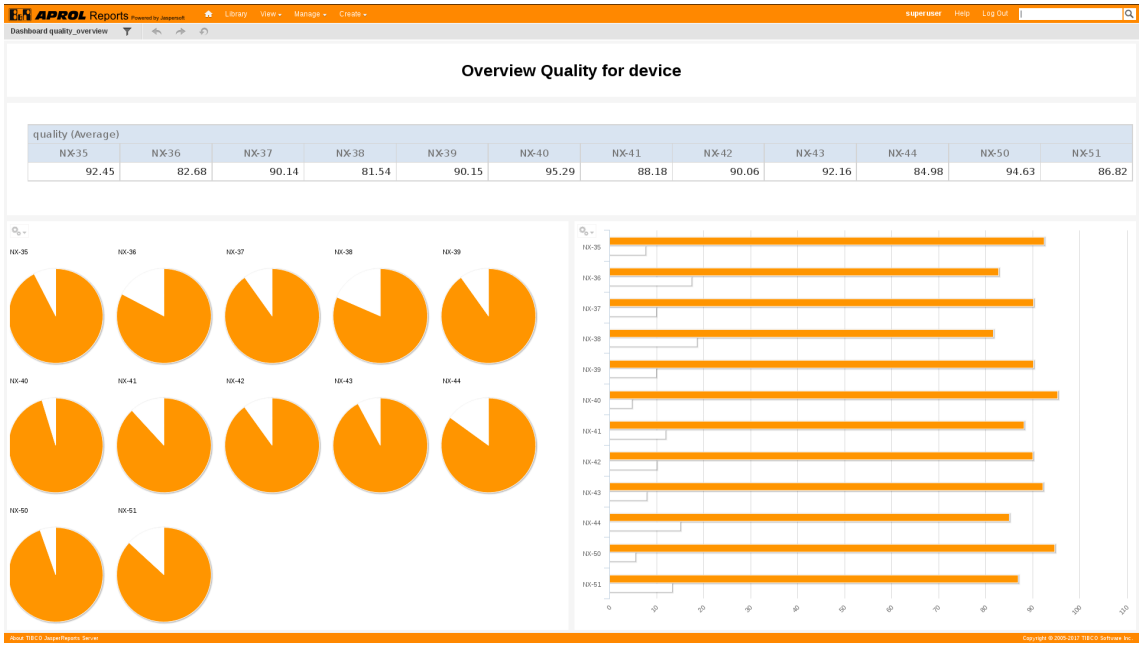

DisplayCenter
File View Go to Maintenance Alarms Tools Login Help
B&R Internal Company License
Host: aprolj0:0 Alarms: 0 Top Prio: - Not Acknowledged: 0 Top Prio NA: - Locked: 0 FILTER: Time [ Group: SQL ]
<---
Hlavní obrazovka
Přehled bodů z PLC
--->
Linka 9
Operátor Uzivatel 03
Zakázka Zakazka 583
Produkt Produkt 05
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-43
Stav zařízení 50 Porucha
Stav programu 20 Automatic
Typ poruchy 2241
Hotové kusy 2443 pcs
Počet kusů 7790 pcs
Idealní cyklus 1500 ms
Linka 10
Operátor Uzivatel 05
Zakázka Zakazka 314
Produkt Produkt 02
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-44
Stav zařízení 50 Porucha
Stav programu 20 Automatic
Typ poruchy 2314
Hotové kusy 730 pcs
Počet kusů 6696 pcs
Idealní cyklus 5000 ms
Linka 11
Operátor Uzivatel 02
Zakázka Zakazka 313
Produkt Produkt 02
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-50
Stav zařízení 40 Výroba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 941 pcs
Počet kusů 2520 pcs
Idealní cyklus 5000 ms
Linka 12
Operátor Uzivatel 03
Zakázka null
Produkt null
Počet zmetků 0 pcs
Typ zmetku 0
Potvrzení NOK
Směna 8 hour
Linak NX-51
Stav zařízení 20 Údržba
Stav programu 20 Automatic
Typ poruchy 0
Hotové kusy 0 pcs
Počet kusů 0 pcs
Idealní cyklus 5000 ms
APROL R 4.2-05
Prehledova obrazovka PLC bodu
Prehledovka3
DisplayCenter
11:14:51 AM

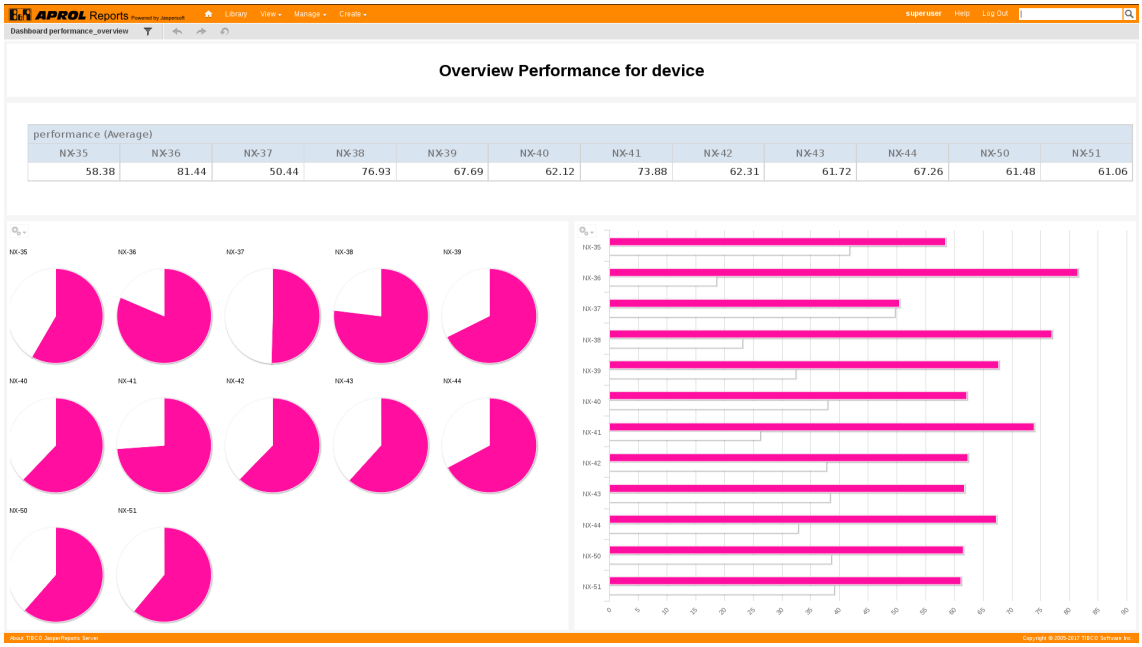

DisplayCenter
File View Go to Maintenance Alarms Tools Login Help
B&R Internal Company License
Host: aprolj0:0 Alarms: 0 Top Prio: - Not Acknowledged: 0 Top Prio NA: - Locked: 0 FILTER: Time [ Group: SQL ]
APROL Reports
Library View Manage Create
superuser Help Log Out
Dashboard_company
Availability
Performance
Quality
Factory Dashboard
OEE
30.8%
Overview
Date from 2020-05-05 till 2020-05-06
Total number of pieces: 59147pcs
Total number of NOK: 5743pcs
Production
Downtime
Hlavní obrazovka
APROL R 4.2-05
Dashboard_main
8:22:50 PM

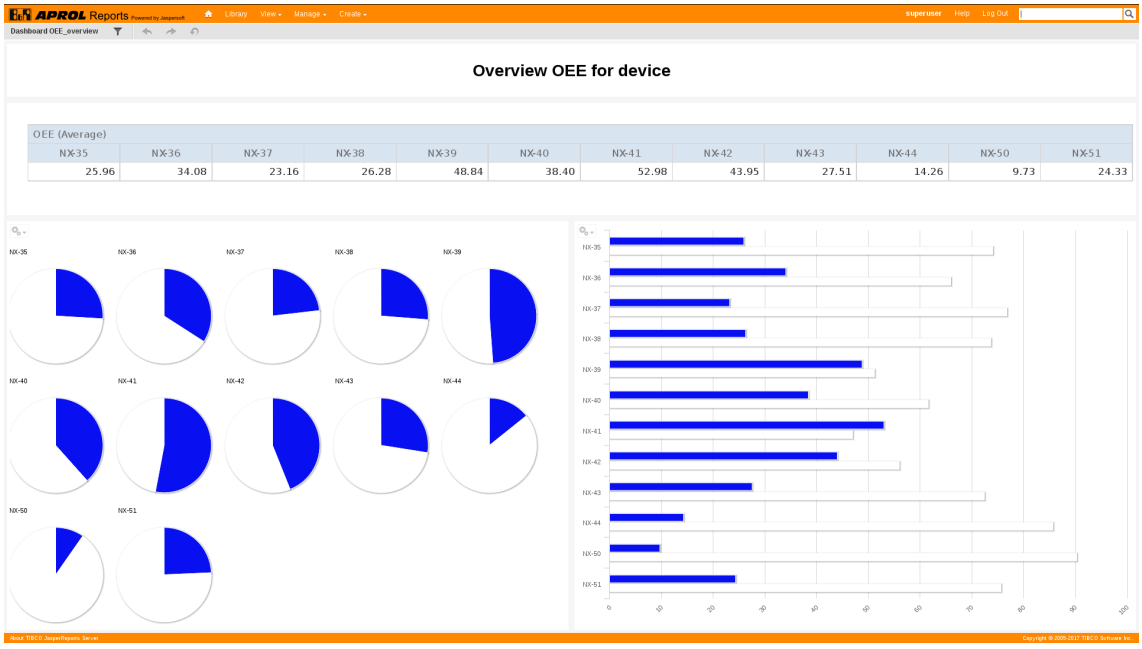

DisplayCenter
File View Go to Maintenance Alarms Tools Login Help
B&R Internal Company License
Host: aprolj0:0 Alarms: 0 Top Prio: - Not Acknowledged: 0 Top Prio NA: - Locked: 0 FILTER: Time [ Group: SQL ]
APROL Reports
Library View Manage Create
superuser Help Log Out
Report_main
Daily report
Company overview
Date from 2020-05-05 till 2020-05-05
Availability 50.6%
Performance 64.8%
Quality 88.6%
OEE
29.3%
Device overview
NX-44
Availability 12.9% Performance 62.6% Quality 87.1% OEE 7.0%
List of contract
Contract Product Num. product NOK Plan product A P Q OEE
Zakazka 297 Produkt 02 50 0 52 37.9 68.4 96.2 24.9
Zakazka 300 Produkt 02 400 58 504 85.5 56.7 67.9 32.9
Total 450 58 556
List of NOK
Hlavní obrazovka
APROL R 4.2-05
Report_daily
8:23:18 PM

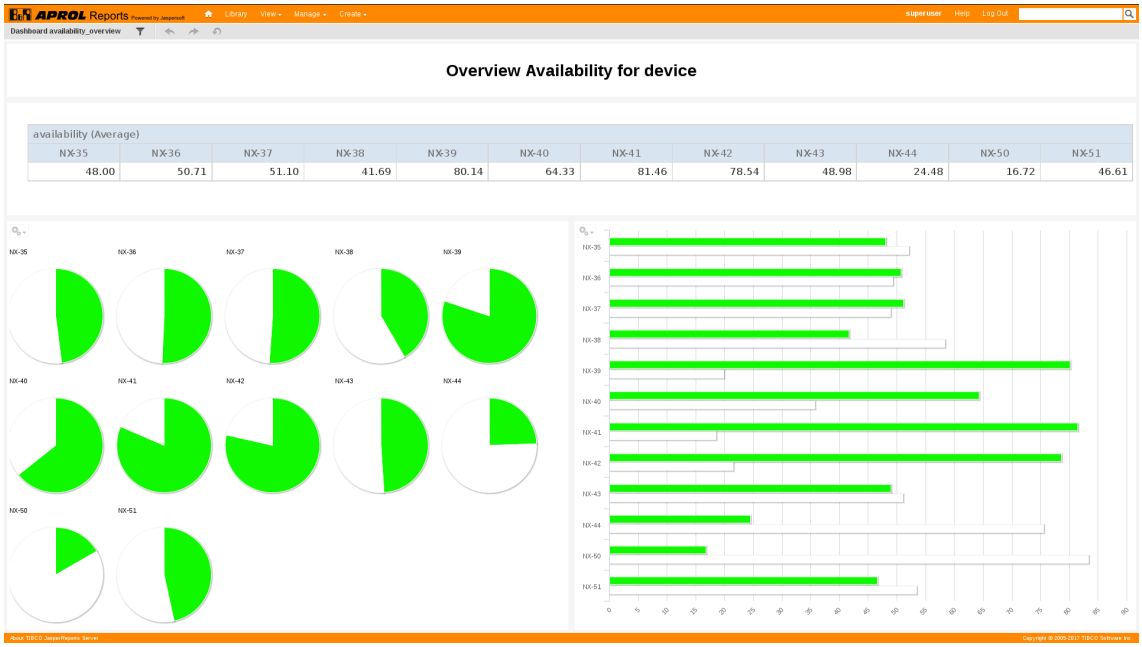
# C Dashboard v APROL pomocí JasperReports Server

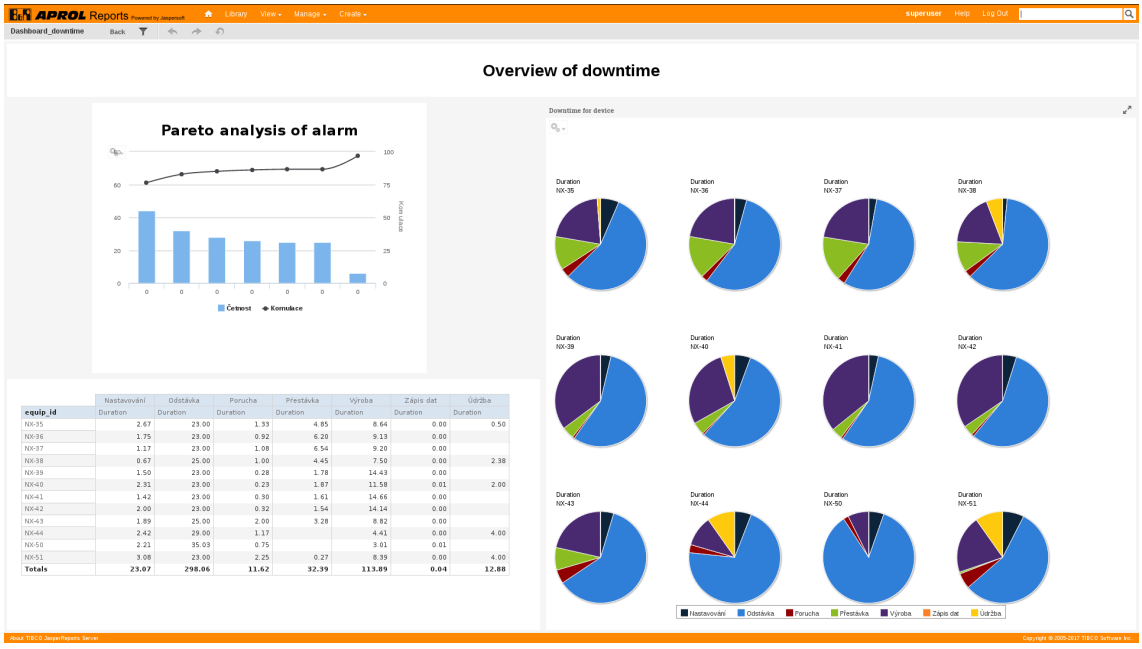


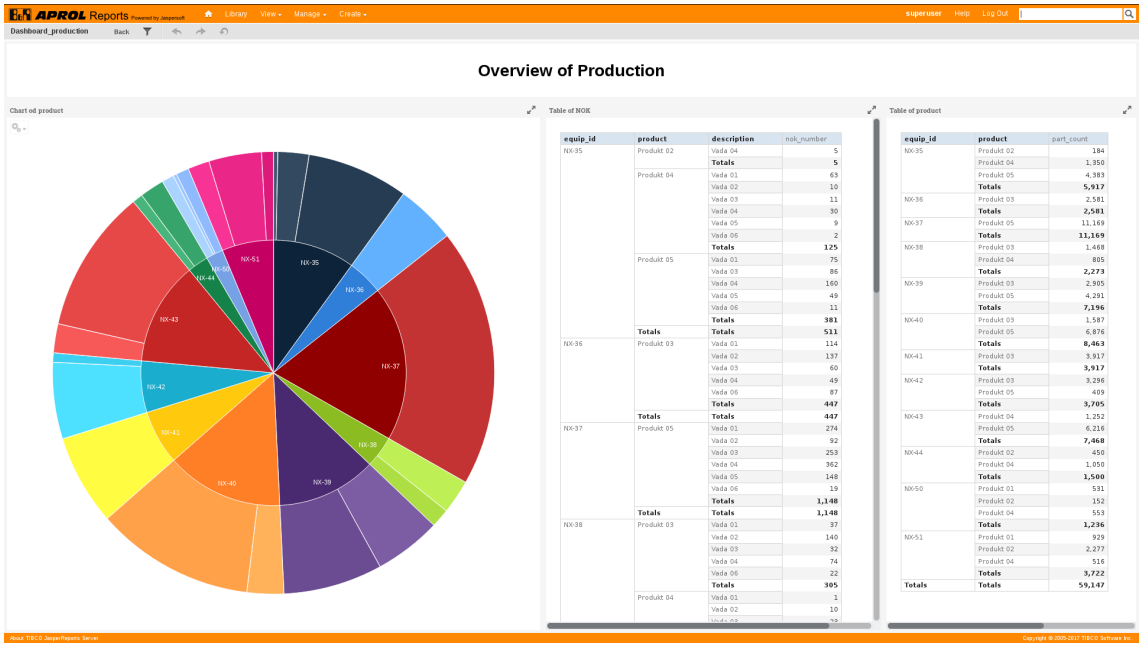

APROL Reports
Library
View
Manage
Create
superuser
Help
Log Out
Dashboard performance_overview
Overview Performance for device
performance (Average)
NX-35 NX-36 NX-37 NX-38 NX-39 NX-40 NX-41 NX-42 NX-43 NX-44 NX-50 NX-51
58.38 81.44 50.44 76.93 67.69 62.12 73.88 62.31 61.72 67.26 61.48 61.06

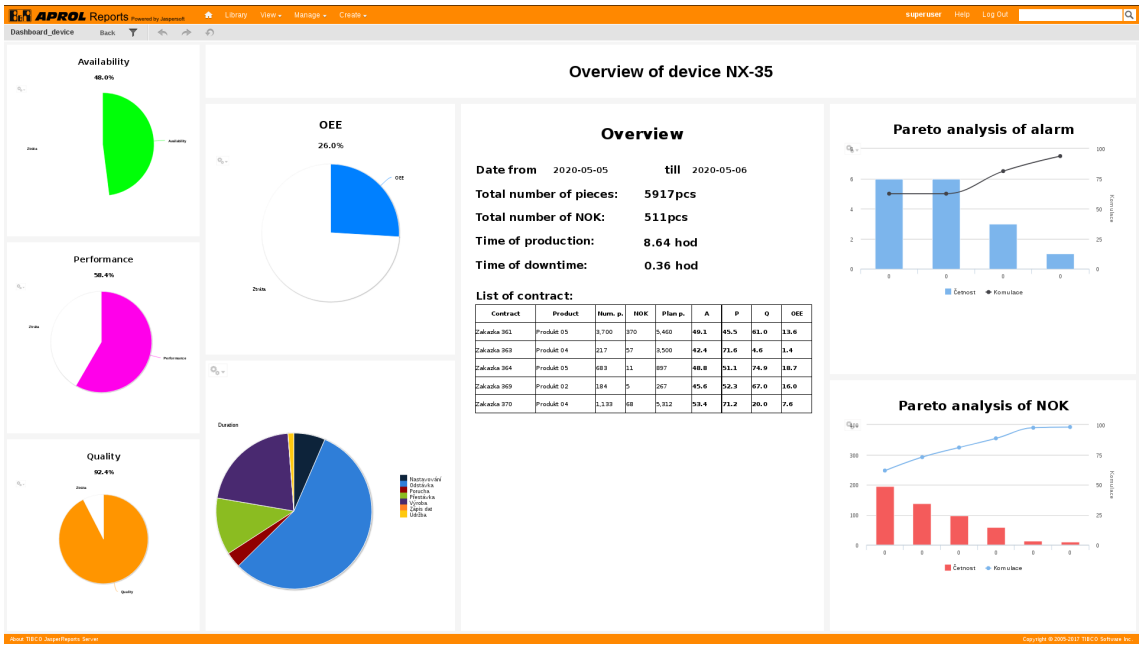

APROL Reports
Library
View
Manage
Create
superuser
Help
Log Out
Dashboard OEE_overview
Overview OEE for device
OEE (Average)
NX-35 NX-36 NX-37 NX-38 NX-39 NX-40 NX-41 NX-42 NX-43 NX-44 NX-50 NX-51
25.96 34.08 23.16 26.28 48.84 38.40 52.98 43.95 27.51 14.26 9.73 24.33




| equip_id | Nastavování Duration | Odstávka Duration | Porucha Duration | Přestávka Duration | Výroba Duration | Zápis dat Duration | Údržba Duration |
|---|---|---|---|---|---|---|---|
| NX-35 | 2.67 | 23.00 | 1.33 | 4.85 | 8.64 | 0.00 | 0.50 |
| NX-36 | 1.75 | 23.00 | 0.92 | 6.20 | 9.13 | 0.00 | |
| NX-37 | 1.17 | 23.00 | 1.08 | 6.54 | 9.20 | 0.00 | |
| NX-38 | 0.67 | 25.00 | 1.00 | 4.45 | 7.50 | 0.00 | 2.38 |
| NX-39 | 1.50 | 23.00 | 0.28 | 1.78 | 14.43 | 0.00 | |
| NX-40 | 2.31 | 23.00 | 0.23 | 1.87 | 11.58 | 0.01 | 2.00 |
| NX-41 | 1.42 | 23.00 | 0.30 | 1.61 | 14.66 | 0.00 | |
| NX-42 | 2.00 | 23.00 | 0.32 | 1.54 | 14.14 | 0.00 | |
| NX-43 | 1.89 | 25.00 | 2.00 | 3.28 | 8.82 | 0.00 | |
| NX-44 | 2.42 | 29.00 | 1.17 | | 4.41 | 0.00 | 4.00 |
| NX-50 | 2.21 | 35.03 | 0.75 | | 3.01 | 0.01 | |
| NX-51 | 3.08 | 23.00 | 2.25 | 0.27 | 8.39 | 0.00 | 4.00 |
| **Totals** | **23.07** | **298.06** | **11.62** | **32.39** | **113.89** | **0.04** | **12.88** |




| Contract | Product | Num. p. | NOK | Plan p. | A | P | Q | OEE |
|---|---|---|---|---|---|---|---|---|
| Zakazka 361 | Produkt 05 | 3,700 | 370 | 5,460 | 49.1 | 45.5 | 81.0 | 13.6 |
| Zakazka 363 | Produkt 04 | 217 | 57 | 3,500 | 42.4 | 71.6 | 4.6 | 1.4 |
| Zakazka 364 | Produkt 05 | 683 | 11 | 897 | 48.8 | 51.1 | 74.9 | 18.7 |
| Zakazka 369 | Produkt 02 | 184 | 5 | 267 | 45.6 | 52.3 | 67.0 | 16.0 |
| Zakazka 370 | Produkt 04 | 1,133 | 68 | 5,912 | 53.4 | 71.2 | 20.0 | 7.6 |

# D Obsah přiloženého DVD

```
/.....................................kořenový adresář přiloženého DVD
├── Jiří Czipszer - DP 2020.pdf
├── Export_xczips00_Diplomka_PLC.zip
├── Export_Project_PDADip.tar
├── Export_Library_xczips00_PDA.tar
├── Export_LibParts_AddOnTools.imp
├── Export_JaspersoftStudio_PDADip.zip
├── Export_JasperreportServer_PDADip.zip
├── Example_Daily_Report.pdf
├── Video
│   └── ...
└── Screenshot
    ├── CaeManager
    │   └── ...
    ├── DisplayCentrum
    │   └── ...
    └── JasperReports Server
        └── ...
```