České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Systém na sledování výroby

Bc. Jan Herzán

Vedoucí práce: prof. Dr. Ing. Zdeněk Hanzálek

Studijní program: Otevřená informatika

Obor: Softwarové inženýrství

11. května 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jan Herzán**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Systém na sledování výroby**

Pokyny pro vypracování:

1. Navrhněte vhodná data pro sběr z výrobního procesu. Navrhněte vhodný datový model. Zvolte vhodné rozhraní pro datový vstup a výstup.
2. Navrhněte business logiku pro zpracovávání dat a serverovou architekturu.
3. Navrhněte grafické uživatelské rozhraní pro klienta, přizpůsobené pro vstupy z různých periferií (klávesnice, myš, čtečka čárových kódů, čtečka čipů).
4. Implementujte systém jako webovou službu, ke které je možné přistupovat skrze webové prohlížeče podporující standard HTML5.
5. Otestujte funkčnost serverové části i klienta.

Seznam odborné literatury:

Harjunkoski, I., Bauer, R. (2014): Sharing Data for Production Scheduling Using the ISA-95 Standard. Frontiers in Energy Research, 2, Article 44.

Manufacturing Execution System - MES: Jürgen Kletti, Springer 2007

Vedoucí: prof.Dr.Ing. Zdeněk Hanzálek

Platnost zadání: do konce letního semestru 2015/2016

[redacted]

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

[redacted]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 3. 2015

Poděkování

Rád bych zde poděkoval panu Prof. Dr. Ing Zdeňku Hanzálkovi, Ing. Janu Smejkalovi a Ing. Janu Dvořákovi za jejich rady a čas, který mi věnovali při řešení dané problematiky.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 10. 5. 2015 .....

Abstrakt

Tato práce je věnována problematice MES systémů. V teoretické části se práce zabývá jednotlivými funkcemi systému a jejich použitím v praxi. Dále pojednává o propojení s jinými systémy, které se podílejí na spravování továren. Jedna z kapitol je věnována metodě hledání kritické cesty, která k funkcím MES systémů neodmyslitelně patří. V praktické části se práce zabývá návrhem jednoduchého MES systému, jeho funkcionalitou a datovou strukturou. Na základě návrhu je tento systém implementován a testován. Výsledkem je MES systém, který může sloužit jako samostatný systém pro malou výrobní společnost, nebo jako modul do většího celku.

Abstract

This thesis is focused on Manufacturing Execution Systems. The theoretical section of the thesis looks into functionality of such systems and their practical usage. Next part is about connections with other systems which participate on the factory management. One chapter is focused on the critical path algorithm which is inseparable part of MES systems. The practical section of the thesis is concentrated on designing a simple MES system. It is mainly focused on functionality and data model. Implementation and testing is based on the design. The outcome is MES system that can be used as a self-standing system or as a module for bigger manufacturing solution.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Rychlá, kvalitní a pružná výroba je klíčem k úspěchu každé továrny, která chce přežít na nelítostném trhu. Aby bylo možné takového stavu docílit, je potřeba sledovat každý proces v továrně a detailně jej analyzovat. V tomto směru je v dnešní době možné použít velké množství softwarových nástrojů, se kterými je jednodušší tuto analýzu provádět. Takovýto systém se nazývá MES - Manufacturing Execution System.

Většina moderních výrobních společností má již zavedený některý z těchto MES systémů, kterých je na českém trhu velké množství. Takovýto MES systém umí spravovat zakázky a rozdělovat je na jednotlivá pracoviště, zaznamenávat údaje z výroby, upozorňovat na události a zaznamenávat polohu jednotlivých výrobků na skladu. Mnohé MES systémy obsahují mnohem více funkcí, než zde bylo vyjmenovano.

Když si představíme výrobní halu bez MES systému, pak chce provozovatel mít zpětnou vazbu. Tu však musí získávat přímo od pracovníků na výrobním patře. (Je samozřejmé, že nikdo na sebe neprozradí svou neaktivitu, nebo že nesplnil něco, co měl.) A zde vznikají nepřesnosnosti, se kterými se továrna nikdy nemůže posunout dále. Oproti tomu, pokud je zaveden MES systém, žádna data nemohou být zkreslena. Systém je zaznamenává sám a okamžitě odhaluje jakákoliv selhání nebo nedbalost. Systém nemá svědomí ani morálku, data tedy budou vždy přesná. Současně systém dokáže přijímat signály ze strojů a analyzovat je. Tím je možné provádět hlubší analýzy a současně to umožňuje reagovat okamžitě na poruchy.

MES systémy také umožňují zobrazování obecných informací na informačních tabulích, takže personál na hale je okamžitě informován o patřičných událostech (například: zavolání mistra k určitému pracovnímu místu, přivolání manipulanta pro odvoz výrobků na sklad ...).

1.1 Cíl práce

Konfigurace a ovládání těchto MES systémů bývá komplikované z důvodu jejich složitosti a komplexnosti. Nasazení rozsáhlého MES systém do malé společnosti s sebou nese velké pořizovací náklady. Na druhou stranu velká část funkcí nemusí být využita. Cílem této práce je vytvořit jednoduchý MES systém, který bude sám o sobě funkčně dostačující pro malou výrobní společnost. Tento systém je však současně navržen tak, aby mohl být součástí většího systémového celku (jako modul). Jak systém přesně pracuje je popsáno v kapitolách o funkcionálních a nefunkcionálních požadavcích (5 a 6).

1.2 Obsah práce

Tato práce obsahuje pojednání o MES systémech na českém trhu a jejich srovnání. Jedna z kapitol se zabývá algoritmem pro hledání kritické cesty a jeho použití v praxi. Hlavní náplní práce je ale tvoření MES systému, implementace jeho funkcí a jeho testování.

Kapitola 2

Použité termíny

2.1 Kontinuální výroba

Kontinuální výroba[6] (nebo též masová výroba) je výroba velkého množství standardizovaných produktů. Tento druh výroby je spojován s výrobní linkou, která zajišťuje vysoký výrobní výkon s nízkými náklady na produkci. Masová výroba se používá ve velkém množství průmyslů, typicky je spojována s automobilovým, potravinářským a chemickým.

2.2 Diskrétní výroba

Diskrétní výroba[5] je druh výrobní techniky, při které se vyrábí jeden, nebo malá skupina výrobků, které jsou pro specifického zákazníka. Tento druh výroby je spojen s vysokou kvalitou produktů, ale ovšem s odpovídající cenou. Vyznačuje se též vysokou pružností a možností přizpůsobit produkt k potřebám zákazníka.

2.3 Šaržovitá výroba

Šaržovitá výroba[3] je druh výrobní techniky, kdy se produkt vyrábí po určitých skupinách zvaných šarže. Každý z výrobků se přesouvá mezi všemi stanovišti a je zpracován stejným způsobem, dokud jich nevznikne požadovaný počet. Následně, pokud chceme vyrábět výrobky jiného druhu, je výrobní linka přestavěna a celý proces se opakuje. Tato technika je výhodná pro výrobu sezónních produktů nebo pro malé fabriky, které si nemohou dovolit kontinuální výrobu. Na druhou stranu přestavování a testování správnosti nastavení linky je považováno za neproduktivní, tudíž nevýdělečný čas pro výrobnu. Tento způsob výroby se používá zejména v pekařství, výrobě obuvi a ve farmaceutickém průmyslu.

2.4 KPI – Key performance indicators

Klíčové výrobní ukazatele pomáhají přibližovat kvalitu a efektivnost výrobního procesu a odrážejí skutečné hospodaření podniku. Jednotlivé ukazatele lze počítat za libovolné období v reálném čase, tudíž je možné dělat důležitá rozhodnutí okamžitě.

2.4.1 MTBF – Mean time between failures

Mean time between failures[7], českým názvem střední doba mezi poruchami, je aritmetický průměr dob mezi poruchami. Touto dobou se rozumí čas od chvíle, kdy zařízení začalo pracovat, až do poruchy, tedy momentu, kdy zařízení není schopné vykonávat svojí funkci. Výpočet je možný provést pomocí následujícího vzorce:

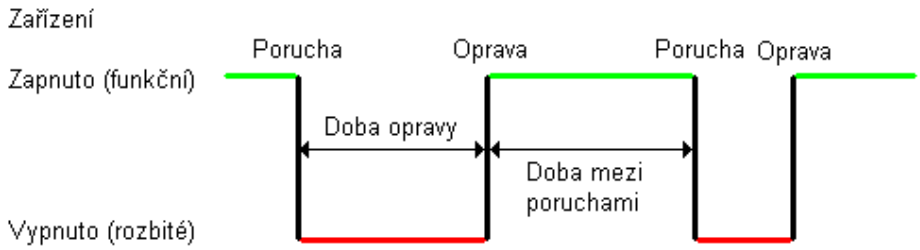
$$MTBF = \frac{\sum_{P_i \in DobyMeziPoruchami} P_i}{početPoruch}$$

2.4.2 MTTR – Mean time to repair

Mean time to repair[8], českým názvem střední doba opravy, je aritmetický průměr dob, které jsou potřeba k uvedení zařízení do provozuschopného stavu. Touto dobou rozumíme čas

od poruchy do opětovného spuštění stroje. Výpočet je možný provést pomocí následujícího vzorce:

$$MTTR = \frac{\sum_{P_i \in DobyOpravy} P_i}{početPoruch}$$



Obrázek 2.1: MTBF a MTTR

2.4.3 OEE – Overall Equipment Effectiveness

Overall equipment effectiveness[13], českým názvem celková efektivita zařízení (CEZ), je kvantitativní ukazatel efektivnosti využití zařízení. Tento parametr ukazuje, jak dobře se využívá dané zařízení z hlediska provozního času, výkonu a kvality výroby. Výpočet je možný provést pomocí následujícího vzorce:

$$OEE = \text{dostupnost} \times \text{výkon} \times \text{úroveňKvality}$$

Z Obrázku 2.2 je vidět, že ztráty na produktivitě zařízení jsou způsobeny několika základními faktory:

- Seřizování a nastavování zařízení (plánované prostoje)
- Poruchy vyvolané chybou zařízení nebo nedostatkem vstupního materiálu
- Redukce rychlosti (náběhové rychlosti, zpomalení z provozních důvodů)
- Ztráta kvality (výroba zmetků)

Obrázek 2.2: Overall Equipment Effectiveness

Hodnoty OEE se u standardních továren pohybuje mezi 30 až 65 %. Pokud bychom se dívali na závody světové třídy, pak zjistíme, že dosahují až hodnoty 85%.

2.5 Data object model

Data object model, zkráceně DOM, je aplikační model, který umožňuje spravovat prvky ve webové stránce. DOM tak tvoři velmi jednoduchý, ale účinný nástroj, kterým je možné měnit, přidávat, odebírat nebo přesouvat jednotlivé prvky na stránce. Data object model má stryktně stromovou strukturu, jejíž nejvyšší rodičovský element je takzvaný *document*. Pokud chceme přistupovat k nějakému prvku ve stránce, musíme k němu přistupovat jako k potomkovi elementu *document*.

2.6 Focus

Focus (občas do češtiny překládané jako "fokus") je označení pro zaměření uživatelských akcí na konkrétní prvek nebo element. Pokud se omezíme pouze na význam v *HTML5* technologiích, pak tím můžeme rozumět označení prvku, se kterým uživatel prováděl iterakci (například tlačítko, pole pro vyplnění textu, odkaz nebo jiný formulářový prvek). Například pokud máme formulář, který obsahuje několik vyplnitelných políček, je možné prvnímu políčku dát focus a následně je do něj umožněno uživateli zapisovat text. Jakmile je uživatel hotový, předá focus jinému prvku (buď myší, nebo pomocí tlačítka *Tab*) a takto postupuje až do vyplnění celého formuláře. Avšak ne na všechny prvky DOMu může být nastaven focus. Pokud si vezmeme například obyčejný kontejnerový element * *, tak tento element sám o sobě není schopen dostat focus. Je potřeba jej pro tento účel speciálně upravit. Focus je důležitý pro všechny prvky, které nějakým způsobem zachytávají uživatelskou aktivitu, protože na základě focusu DOM předává jednotlivé události správným prvkům ve stránce. Na základě těchto událostí stránka může reagovat a poskytovat uživateli zpětnou vazbu.

2.7 MES a ERP systémy

Manufacturng Execution System, zkráceně MES, je druh systému, který pomáhá výrobnímu patru zrychlovat a zefektivnit výrobu a současně zaznamenává data o výrobě. Oproti tomu Enterprise Resource Planning systém, zkráceně ERP, je určen pro obchodní a manažerská oddělení společností a s výrobou pomáhá buď pouze částečně, nebo je za tímto účelem, propojen s MES systémem. Více o MES a ERP systémech v Kapitole 3.

2.8 Kritická cesta

Kritická cesta je nejdelší cesta mezi počátečním a koncovým bodem grafu. Její výpočet se používá v různých odvětvích projektování a plánování. Pomocí tohoto výpočtu můžeme nalézt kritické úkoly, které nesmějí být zpožděny, jinak by se prodloužil celý proces výroby. Více o kritické cestě a jejím nalezení v Kapitole 4.

2.9 WSDL

Web service description langauge[18], zkráceně WSDL, je jazyk pro definici rozhraní, které nabízí webová služba. Toto rozhraní je definováno pomocí XML a definuje metody a datové typy, které jsou rozhraním přijímané a jaké jsou návratové hodnoty těchot metod.

Kapitola 3

MES systémy

První náznaky MES systému se objevují již v osmdesátých letech minulého století. Malé systémy začaly vznikat pro jednotlivá řídící oddělení společností, jakými jsou plánování výroby, personální oddělení a oddělení řízení kvality. Systémy však vznikaly jako separátní složky, které nebyly navzájem propojené. Tyto malé systémy tvořily střípky, jejichž spojením se poté zrodil MES systém.

Postupem času a zvyšováním míry používání počítačové techniky ve výrobě (tento jev je též označován jako Computer Integrated Manufactoring, zkráceně CIM), se přestalo hledět na data z jednotlivých sektorů jako na oddělené informace a začaly se mezi nimi tvořit spojitosti.

Začátkem devadesátých let výrobci systémů pro sběr dat zahájili vylepšování svých specializovaných systémů přidáváním funkcionalit z jiných odvětví výroby. Tímto způsobem se funkce shlukovaly do větších celků, pomocí kterých se mohla sbírat data a provádět nad nimi základní analýza. Tímto způsobem vznikly tři základní odvětví systémů, které tvoří základní kameny MES systémů tak, jak je známe dnes:

- Systémy pro kontrolu produkce
  - Sběr dat z terminálů
  - Sběr dat ze strojů

  - Ovládání a nastavování jednotlivých strojů

- Systém pro správu zaměstnanců
  - Logování uživatelů a logování časových záznamů
  - Kontrola přístupů
  - Krátkodobé plánování lidských zdrojů

- Systém pro správu kvality
  - Kontrola výstupů
  - Inspekce strojů

3.1 Funkce MES systému podle organizace MESA

Problém MES systémů byl již hodněkrát zpracováván různými společnostmi za účelem zabránění trivializace tohoto problému. Nejvýznamnější organizací, která se touto problematikou zabývá a která je označována jako nejzkušenější, je MESA (Manufacturing Execution Solutions Association).

Tato organizace se svým pragmatickým přístupem k věci vytvořila dvanáct skupin funkcí, jejichž kombinací můžeme sestavit MES systém.

3.1.1 Detailní plánování

Optimalizace rozvrhu a organizace objednávek je klíčem k úspěchu pro každou výrobu. MES systém rozvrhuje jednotlivé úlohy tak, aby výsledný rozvrh bral v úvahu výkon jednotlivých zařízení a veškerá omezení pro zakázky daná výrobními postupy. Rozvrh nemusí být vždy tvořen přímo MES systémem, ale může být vygenerován v ERP systému a následně pouze předán.

3.1.2 Správa zdrojů

Správa zdrojů zahrnuje řízení a monitorování jednotlivých strojů a výrobních nástrojů, které se nacházejí ve výrobní hale.

3.1.3 Zaznamenávání a zobrazování aktuálního stavu zdrojů

Uživatel musí mít přístup k aktuálnímu stavu zařízení (například jestli je aktivní, stojí, nebo je v poruše). Tento stav se musí také zaznamenávat, aby bylo možné zpětné vypočítat klíčové výrobní ukazatele.

3.1.4 Správa dokumentů

Pracovníkovi na daném stanovišti musí být vždy jasný výrobní postup. Pro jednoduchou a jasnou distribuci výrobních postupů, informací o materiálu nebo způsobech skladování výrobku je v MES systému integrovaný modul pro správu a zobrazování dokumentů.

3.1.5 Správa materiálu

Kontrolu nad stavem materiálu na skladě má systém EPR. Úkolem MES systému je kontrola správného materiálového vstupu do jednotlivých úkonů. Zaznamenávání vstupního materiálu je též důležité pro správné sestavení rodokmene výrobku.

3.1.6 Výkonnostní analýza

Tento modul obsahuje porovnání a evaluaci naplánovaných cílů a cílů skutečně dosažených na základě zaznamenaných dat. Také je zde možné zjišťovat jednotlivé klíčové výrobní ukazatele (viz Kapitola 2.4).

3.1.7 Správa postupů

Dodržování správného výrobního postupu pomáhá zvyšovat kvalitu a rychlost produkce. Systém pomáhá dodržování výrobních postupů a dále zjednodušuje nastavování zařízení (například pomocí automatických konfigurací nebo kontroly použití správných nástavců).

3.1.8 Plánování údržby a servisu

Každé zařízení, má-li správně fungovat a má-li fungovat dlouho, musí být pravidelně servisováno. Tyto servisní sekvence musí být zanášeny do rozvrhu a musí se s nimi počítat.

3.1.9 Správa procesů

Sytém pomáhá kontrolovat a řídit plynulost práce na jednotlivých stanovištích a upravovat rozvrh na základě zatížení konkrétních zařízení.

3.1.10 Správa kvality

Kontrola výrobního procesu napomáhá v udržování ideálních podmínek pro produkci kvalitního zboží a výrobků. Tato kvalita je klíčová pro snížení nákladů, zejména díky snížení zmetkovosti a reklamací.

3.1.11 Sběr dat

Sběr, zaznamenávání, organizace a zobrazování dat z výrobních strojů a od pracovníků je stěžejním prvkem každého MES systému.

3.1.12 Sledování výrobků a jejich rodokmene

Systém udržuje dokumentaci všech událostí, které se týkají výroby konkrétního produktu. Tyto záznamy obsahují detaily o materiálu, jednotlivých zákrocích pracovníků, kteří na výrobku pracovali, a podmínky, ve kterých byl výrobek vytvářen.

Na systém MES nesmí být nahlíženo pouze jako na soubor funkcionalit. Jeho úkolem je tvořit zaznamenávací a komunikační kanál napříč celou výrobní společností. Cílem systému je pomoci společnosti přežít a být konkurenceschopná na nelítostném trhu.

3.2 Rozdíly mezi MES a ERP

Abychom mohli tvořit srovnání MES systému se systémem ERP, musíme si nejdříve definovat, co vlastně ERP (Enterpise Resource Planning) systém je.

ERP systémem je myšlen systém, který pokrývá širokou funkční oblast podniku, jakými jsou účetnictví, logistika, obchod, marketing, finance, skladové hospodářství, správa zákazníků, správa majetku. Mimo další může spravovat též oblasti výroby. Proč by tedy měly existovat systémy jako je MES, když ERP systém je též schopen pracovat v oblasti výroby?

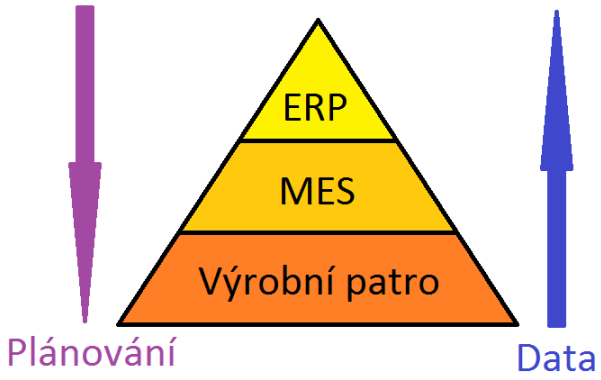
Důležitým a možná největším rozdílem mezi těmito druhy systémů, je pojetí času. ERP systém pohlíží na fungování společnosti z obchodního hlediska, tudíž časové horizonty, ve kterých se pohybuje, jsou v řádech dnů, týdnů až měsíců. Tím dává uživateli možnost plánovat na dlouhou dobu dopředu a sledovat výrobní ukazatele z patřičného nadhledu. Oproti tomu MES systém se dívá na fungování společnosti z opačného konce, tedy z konce výrobního. Časové horizonty jsou úměrně kratší než u ERP, tedy v řádech dnů, hodin, minut, někdy dokonce v řádu sekund. Data jsou přenášena s každým cyklem stroje, což může činit desítky parametrů během velmi krátkých časových okamžiků. Tyto parametry jsou v reálném čase zpracovávány a vyhodnocovány. Je tedy očividné, že díky své pružnosti se MES systém bude lépe přizpůsobovat potřebám výroby jako takové.

Pokud ještě zůstaneme u rozdílu vnímání času, pak je důležité podotknout, že systém,

který má být co nejblíže spjatý s výrobou, musí být dostupný nepřetržitě. Tato schopnost je důležitá pro třísměnnéhý provoz. ERP systémy využívají víkendových dnů a nočních hodin pro aktualizace a důležité výpočty, zatímco MES systém je navržen tak, aby změny probíhaly bez dopadu na výrobu.

ERP systém je navrhován tak, aby byla data, která se v něm nacházejí, přístupná kdykoliv a odkudkoliv. To zajišťuje obchodnímu oddělení rychlý přehled pro smlouvání termínů dodávek a rychlý přístup k informacím, které si klient žádá. Takovýto přístup vyžaduje permanentní připojení na internet, jak zdrojového systému, tak i klientské části. Ve výrobním prostředí však nemůžeme takovéto podmínky vždy zaručit. Prostředí továren bývá plné rušivých signálů a elektromagnetického šumu, který znemožňuje kontinuální datový přenos vzduchem. Ani datový tok skrze kabelové vedení nemusí být vždy zcela spolehlivou cestou. Z tohoto důvodu se MES systémy vždy snaží být nasazené co nejblíže lokálnímu provozu, aby nebyly vystaveny problémům nedostatečně kvalitního připojení nebo výpadkům sítě.

V neposlední řadě hraje velký rozdíl mezi těmito druhy systémů specializace. ERP systém se snaží vyhovět potřebám všech společností, a to i takovým, které nemají s výrobou dočista nic společného. Jejich univerzálnost však častokrát nemusí vyhovovat veškerým potřebám firmy. V posledních letech byly zaznamenány trendy specializačních modulů do ERP systémů, ale takto tvořená rozšíření nebyla zatím přijata s kladným ohodnocením. Oproti tomu MES systémy se snaží specializovat na konkrétní odvětví výroby. Proto je velice důležité při volbě MES systému zvážit, zdali pokrývá požadavky, které jsou zákazníkem požadovány. Hierarchii systémů můžeme vidět na Obrázku 3.1.



Obrázek 3.1: Hierarchie systémů

3.3 MES systémy na českém trhu

V dnešní době se na českém trhu objevuje mnoho MES systémů s různou funkcionalitou. V této kapitole jsou vybrány některé z nich.

3.3.1 Pharis

Systém Pharis je vyvíjen společností UNIS, a.s., která je ryze českou společností a působí na trhu od roku 1990. Její primární činností je řízení technologických procesů. Systém PHARIS společnost začala vyvíjet v roce 2002.

Pharis[16] je výrobní informační systém, jehož úkolem je výrobu připravit, zahájit, řídit, získávat data z výroby a provádět jejich vyhodnocení. Systém je určen pro moderní plánování a řízení výroby. MES Pharis je založen na webových technologiích a svou modulovatelností (obsahuje 35 různých modulů) umožňuje přizpůsobení celého systému k potřebám výroby. Původně byl vyvíjen pro farmaceutický průmysl, ale po přidání některých funkcí byl systém schopen řešit problémy komplexního řízení i v diskrétní výrobě, jakou je kovovýroba a lisování plastů. Převážná většina uživatelů jsou významní dodavatelé plastových výlisků pro automobilový průmysl. Dále je používán při výrobě nábytku nebo v nástrojárnách.

3.3.2 Diames

Diames[1] je Švýcarský proaktivní MES systém, který pracuje v reálném čase. Systém tvoří rozhraní mezi ERP systémem a dílenskou úrovní. Diames zahrnuje certifikovaná rozhraní s nejrozšířenějšími ERP systémy, jakými jsou například SAP nebo Baan. Systém obsahuje sadu funkcionalit pro údržbu, řízení kvality, řízení lidských zdrojů a sledování toku materiálu. Tím pomáhá zefektivňovat výrobu a snižovat náklady.

3.3.3 Hydra MES

Systém Hydra MES[10] má již dlouhou historii a aktuálně vychází ve verzi 8. Tento MES systém je produktem německé společnosti MPDV Mikrolab GmbH a je již nasazen u více než 600 zákazníků po celém světě, čímž si upevňuje místo mezi světovými jedničkami na trhu. Tento systém splňuje všechny požadavky moderního MES systému. Je tedy dobrým informačním článkem mezi výrobním patrem a obchodně zaměřenou částí společnosti nebo patrem managementu. Hydra poskytuje lidem pracujícím ve výrobním prostředí relevantní informace a ohodnocení, funkce pro plánování, řízení lidských zdrojů a řízení kvality. Současně systém nabízí tyto funkce pro mobilní zařízení, aby mohl uživatel reagovat, jakmile je to potřeba.

3.3.4 Maggio

MES systém Maggio[9] je z dílny společnosti FINSOFT, s.r.o. Systém se zaměřuje především na liniovou (kontinuální) výrobu - textilní, chemický a potravinářský průmysl. Maggio je modulární systémová platforma, která vytváří variabilní systém monitoringu, plánování a řízení výroby malých a středně velkých podniků. Systém je koncipován tak, aby mohl běžet buď samostatně, nebo paralelně s ERP systémem.

3.3.5 Srovnání MES systémů

Každý MES systém se specializuje na různá odvětví výroby, tudíž ne všechny se hodí nasadit všude. V Tabulce 3.1 můžeme nalézt srovnání jednotlivých MES systémů podle portálu MESCentrum.cz[14].

	Pharis	Diames	Hydra MES	Maggio
Kontinuální výroba	2	3	2	5
Šaržovitá výroba	2	1	5	3
Diskrétní výroba	5	4	5	2

Tabulka 3.1: Srovnání MES systémů podle portálu MESCentrum.cz

Kapitola 4

Algoritmus pro hledání kritické cesty

Metoda hledání kritické cesty byla vyvinuta dvojicí Morgan R. Walker and James E. Kelly v padesátých letech minulého století pro řízení projektů správy továren. V dnešní době se používá v mnoha odvětvích jakými jsou stavebnictví, softwarové projekty, výzkumné projekty nebo výroba. Obecně lze tuto metodu použít v jakémkoliv projektu, ve kterém se nacházejí vzájemně provázané a závislé činnosti.

Ve výrobním procesu můžeme nalézt spoustu kritických míst, na kterých závisí včasné vyhotovení zakázky a tím i uspokojení zákazníka. Úkolem MES systému je odhalovat tato úzká hrdla výroby a upozorňovat na ně při průchodu výrobním plánem.

4.1 Kritická cesta

Kritická cesta je definována jako nejdelší možná cesta z počátečního do koncového bodu grafu. Každý graf obsahuje alespoň jednu kritickou cestu. Pokud bereme graf jako časové rozložení jednotlivých prací ve výrobním procesu, uzly jako jednotlivé stavy výrobku a hrany jako činnosti, které k danému stavu vedou, pak je kritickou cestou posloupnost činností, která

bude trvat nejdelší dobu. Pokud by se některá z činností prodloužila, prodlouží se tím doba trvání celé výroby daného produtku. Z tohoto důvodu se každý dobrý výrobní operátor zaměřuje na činnosti, které se na kritické cestě nacházejí.

4.2 Algoritmus

Vstupem algoritmu (pseudokód algoritmu viz Algoritmus 1) pro hledání kritické cesty je orientovaný graf, který má ohodnocené hrany (v algoritmu označený jako $AllNodes$). Hodnota hran odpovídá době trvání jednotlivých činností a orientovanost hran určuje jejich posloupnost.

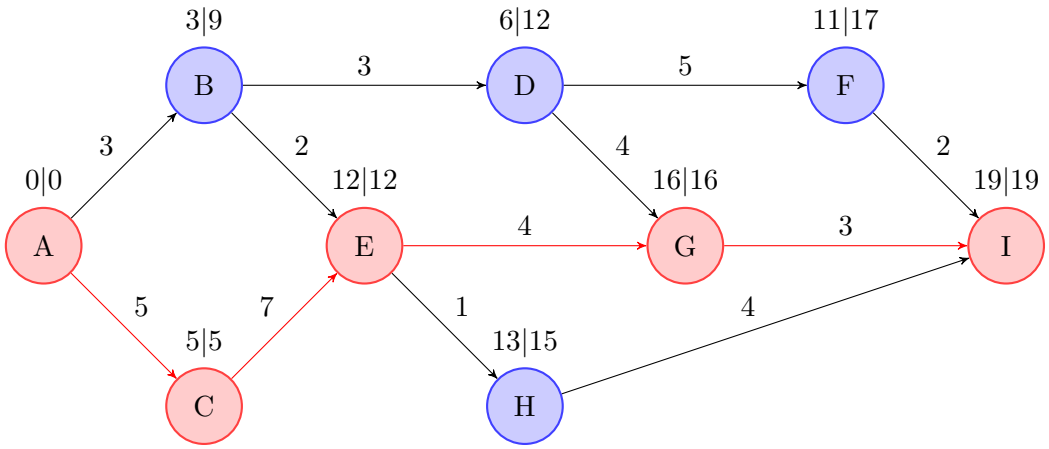
Výstupem jsou posloupnosti uzlů, které tvoří kritickou cestu.

Při inicializaci algoritmu ke každému uzlu přiřadíme dvě proměnné - první naincializujeme na -1 (v pseudokódu je označena jako $forward$), druhou na nekonečno (v pseudokódu je označena jako $backward$). Jednu budeme používat při dopředném průchodu grafem a druhou budeme používat při jeho zpětném průchodu. V počátečních uzlech, tedy v uzlech, do kterých nevede žádná hrana, nastavíme hodnotu dopředné proměnné na 0. Poté procházíme graf tak, že vybíráme uzly, které mají všechny předky již zpracované, a do následujícího uzlu zapisujeme vždy hodnotu jeho předchůdce plus hodnotu hrany. Pokud narazíme na uzel, do kterého vede více vstupních hran, bereme vždy maximum z dostupných hodnot. Tímto způsobem dojdeme až do koncových uzlů grafu, tedy uzlů, ze kterých nevede žádná hrana. Tím je dopředná část algoritmu skončena.

Před zpětným průchodem zkopírujeme ve všech koncových uzlech hodnotu dopředné proměnné do proměnné zpětné.

Zpětný průchod grafem probíhá podobně jako průchod dopředný, s tím rozdílem, že jdeme proti směru hran a hodnotu zpětných proměnných odečítáme. Pokud narazíme na uzel, ze kterého vede více hran, pak vybíráme vždy minimum z možných hodnot. Takto dojdeme až k počátečním uzlům. Tím je zpětná část algoritmu skončena.

Uzly, které mají stejnou hodnotu dopředné a zpětné proměnné leží na kritické cestě.



Obrázek 4.1: Příklad kritické cesty

4.3 Příklad

Na Obrázku 4.1 budeme demonstrovat průchod algoritmem. Nejdříve zapíšeme 0 do bodu A. Následuje dopředná část algoritmu (dopředné proměnné jsou na levé straně). Hodnota dopředné proměnné v bodě B je tedy hodnota bodu A plus hodnota hrany – tedy 0 + 3 = 3. Stejným způsobem doplníme body C a D. Hodnota dopředné proměnnné v bodě E je maximum z hodnot cest přicházejících z bodu B a C – maximum z 3 + 2 (z bodu B) nebo 5 + 7 (z bodu C) – tedy výsledná hodnota je 12. Tímto způsobem dojdeme až do koncového bodu I. Následně přepíšeme hodnotu dopředné proměnné do zpětné proměnné (zpětné proměnné jsou na straně pravé). V první kroku zpětného algoritmu doplníme do bodu H hodnotu 19 - 4 = 15. Stejným způsobem doplníme hodnoty do bodů G a F. Hodnota zpětné proměnné v bodě E bude minimum z cest přicházejících z bodů G a H – minimum z 16 - 4 (z bodu G) nebo 15 - 1 (z bodu H) – výsledná hodnota je tedy 12. Tímto způsobem dojdeme až do počátečního bodu A. Z uvedených výsledků je vidět, že kritická cesta je posloupnost bodů „A, C, E, G, I“.

Výsledkem algoritmu hledání kritické cesty není pouze samotná kritická cesta. Pokud odečteme hodnotu dopředné proměnné od hodnoty zpětné proměnné ve kterémkoliv uzlu, získáme tím dobu, o kterou se může daná úloha zpomalit, aniž by to ovlivnilo celkovou dobu trvání projektu. Zde je též vidět, že všechy body na kritické cestě tuto volnost nemají (jejich možnost opoždění je nulová).

V tomto příkladu máme pouze jeden počáteční uzel i jeden koncový uzel. Pokud by jich bylo více, postup by byl analogický (tedy na začátku se zapíše nula do všech počátečních bodu a po ukončení dopředné části algoritmu se hodnoty proměnných přepisují ve všech koncových uzlech).

4.4 Složitost algoritmu

Při každém průchodu (dopředném i zpětném) grafem projdeme celý graf právě jednou. Při vstupu do každého uzlu je potřeba zpracovat všechny jeho výstupní hrany a následně procházíme všechny jeho potomky. Složitost algoritmu je:

$$\mathrm{O}(početUzlů \times početHran)$$

Algorithm 1 Algoritmus hledání kritické cesty

```
Vstupem je orientovaný graf s ohodnocenými hranami AllNodes.
AllNodes ← {−1, INF}
open ← StartNodes

while !open.isEmpty do                                   ▷ Dopředný průchod
    for all offspring in actualNode.offsprings do
        potentialNewValue = actualNode.forward +
actualNode.EdgeValue(offspring)
        if offspring.forward < potentialNewValue then
            offspring.forward = potentialNewValue
        end if

        if offspring má všechny předky zpracované then
            open ← offspring
        end if
    end for
    Odeber actualNode ze seznamu open
    Zvolení nového actualNode ze seznamu open
end while

for all node in EndNodes do          ▷ Přenesení hodnot z levé na pravou stranu
    node.backward = node.forward
end for

open ← EndNodes

while !open.isEmpty do                                   ▷ Zpětný průchod
    for all parrent in actualNode.parrents do
        potentialNewValue = actualNode.backward − actualNode.EdgeValue(parrent)
        if parrent.backward > potentialNewValue then
            parrent.backward = potentialNewValue
        end if

        if parrent má všechny rodiče zpracované then
            open ← parrent
        end if
    end for
    Odeber actualNode ze seznamu open
    Zvolení nového actualNode ze seznamu open
end while

for all node in AllNodes do                      ▷ Hledání uzlů na kritické cestě
    if node.forward == node.backward then
        Přidání uzlu do seznamu uzlů na kritické cestě
    end if
end for
Výstupem jsou posloupnosti uzlů, které tvoří kritickou cestu/cesty.
```

Kapitola 5

Funkcionální požadavky

Systém musí splňovat všechny následující funkcionální požadavky:

- Systém musí zobrazovat zakázky (rozvržené i nerozvržené).
- Systém musí umožňovat vytvářet záznamy k jednotlivým zakázkám.
- Systém musí obsahovat přihlašovací modul, ve kterém budou evidováni všichni uživatelé, kteří se do systému mohou přihlásit.
- Systém musí umět zobrazovat výrobní postupy (dokumenty a videa).
- Systém musí obsahovat rozhraní pro propojení s rozvrhovacím softwarem, ERP systémem pro externí import dat o zakázkách.
- Systém musí obsahovat rozhraní, které umožní vstup strojově generovaných dat.
- Systém musí umět vyhodnocovat klíčové výrobní ukazatele.
- Systém musí podporovat aktivní upozorňování na události, které si uživatel zvolí.
- Systém musí podporovat sledování výrobků a jejich rodokmen.
- Systém musí podporovat upozorňování pracovníků na zakázky na kritické cestě.
- Systém musí podporovat lokalizaci.

- Systém musí podporovat výkonnostní analýzy (strojů i zaměstnanců).

5.1 Přihlašování uživatelů

Každý uživatel, který chce provádět záznamy v systému, musí mít své uživatelské jméno a heslo, pod kterými se přihlašuje do systému. Na základě uživatelského jména se vytváří známka ke každému záznamu.

Ke každému uživatelskému jménu jsou přiřazena práva, na základě nichž je uživateli umožněn přístup do jednotlivých sekcí. Práva se dělí do několika kategorií:

- Standardní pracovník
  - Pracovník může vytvářet záznamy pouze k zakázkám, které jsou mu přiděleny plánovačem nebo rozvrhem.
  - Pracovník může přistupovat pouze k dokumentům a výrobním postupům, které bezprostředně souvisí se zakázkami, na kterých pracuje.
- Mistr
  - Mistr má možnost zobrazit přehled pracovišť, která má pod svou správou.
  - Mistr má pravomoc potvrdit pracovníkovi úkon „Konzultace s mistrem“.
- Kvalitář/Technolog
  - Kvalitář/Technolog má přístup do sekce rodokmene jednotlivých zakázek.
  - Kvalitář/Technolog má pravomoc potvrdit pracovníkovy úkony „Konzultace s kvalitářem“ a „Konzultace s technologem“.
- Vedoucí směny
  - Vedoucí směny má přístup do sekce zobrazující klíčové výrobní ukazatele.
- Plánovač
  - Plánovač může měnit rozvržené zakázky a jejich parametry.

- Administrátor
  - Administrátor může přidávat, mazat a editovat uživatele.
  - Administrátor může nastavovat práva jednotlivým uživatelům.
  - Administrátor nastavuje upozornění pro jednotlivé události.

5.2 Zobrazování zakázek

Databáze systému uchovává data o jednotlivých objednávkách. Tyto objednávky mohou být seřazeny do rozvrhu. Takovýto rozvrh je tvořen externí entitou, která o správě zakázek nic neví. Na základě toho, zdali je utvořen rozvrh, je možné systém spouštět ve dvou režimech.

První režim lze aktivovat pouze ve chvíli, kdy je rozvrh utvořený. V takovou chvíli nemůže uživatel měnit pořadí zakázek a je nucen je zobrazovat a zpracovávat v pořadí, které je dané rozvrhem.

Druhý režim umožňuje uživateli zobrazovat a zpracovávat zakázky v libovolném pořadí. Uživatel si zvolí, kterou zakázku chce zobrazit. Tento režim může být spuštěn i ve chvíli, kdy je rozvrh utvořen. V takovou chvíli je režim pouze doporučující, nikoliv však závazný.

5.3 Vytváření záznamů

Zaznamenávání skutečnosti je primární funkcionalitou systému. Vytváření záznamů se provádí při zobrazení zakázky. Systém musí evidovat úkony, které byly na zakázce prováděny. Těmito úkony jsou:

- Příprava zakázky
- Standardní úkon na zakázce
  - Tento záznam je doprovázen daty o počtu vyrobených kusů a vyrobených zmetků.

- Manipulace s materiálem
- Porucha stroje
- Přerušení rozpracované zakázky
- Zásah třetí strany
  - Konzultace s mistrem
  - Konzultace s technologem
  - Konzultace s kvalitářem
  - Výpomoc jiného pracovníka

Systém musí též zaznamenávat úkony, které nejsou spojeny s žádnou zakázkou. Těmito úkony jsou:

- Úklid
- Údržba pracovního místa
- Údržba stroje

Další úkony, které též nejsou spojeny se žádnou zakázkou, ale současně jsou systémem vyhodnocovány jako neproduktivní čas pracovníka. Těmito úkony jsou:

- Přestávka
- Pohyb po areálu

Každý záznam je opatřen časovou známkou a současně musí obsahovat označení uživatele, který jej vytvořil.

5.4 Zobrazování dokumentů a výrobních postupů

ERP systémy obsahují dokumentace k jednotlivým typům zakázek. Systém musí obsahovat modul, který je schopen převzít tyto dokumenty z ERP systému a přiřadit je k jednotlivým zakázkám. Důsledkem přiřazení je možnost zobrazení jednotlivých dokumentů při práci přímo na stanovišti. Systém bude podporovat dokumenty ve formátu *.PDF. Za účelem lepší ilustrace je možné přidávat videa do souborů výrobních postupů. Videa musí být ve formátu, který je podporován prohlížeči, které splňují standard HTML5, tedy *.MP4 a *.OGG.

5.5 Rozhraní pro vstup rozvrhu

Jak již bylo řečeno výše (viz Kapitola 5.2), rozvrh je tvořen externím systémem. Tento systém může předávat rozvrh několika způsoby.

Prvním způsobem je přímé vložení dat do datové struktury v databázi. Tento způsob může být použit pokud modul tvoření rovrhu bude rozšířením pro systém a tudíž při implementaci bude autorovi rozvrhu známa struktura databáze.

Druhým způsobem je vytvoření WSDL (viz Kapitola 2.9), které bude vystaveno na předem stanovené URL adrese a tím bude moci rozvrh tvořící entita přistupovat k systému jako k webové službě. Tento způsob má velkou výhodu v platformové nezávislosti a též v tom, že se systémy nemusí nacházet ve stejné lokální síti.

Třetím způsobem předání dat je pomocí datového souboru s předem definovaným formátem. Pro přenosy takto strukturovaných dat se nejčastěji používá formát *.CSV (tedy každý datový kontejner je napsán na jeden řádek a jednotlivé položky jsou odděleny středníkem). Tento způsob předávání dat je velmi jednoduchý, ale současně velice neefektivní v rychlosti předávání informací.

5.6 Rozhraní pro vstup strojově generovaných dat

Při tvoření záznamů mohou být k dispozici data, která jsou generovaná jednotlivými stroji, které aktuálně na zakázce pracovaly. Systém musí obsahovat jednoduché rozhraní, které bude předávat data ze strojů. Systém je následně přiřadí k patřičnému záznamu. Data z tohoto rozhraní mohou nahrazovat některé údaje, které jsou standardně zaznamenávány pracovníkem skrze uživatelské rozhraní.

5.7 Aktivní upozorňování na události

Aktivní upozorňování slouží k uvolnění rukou kontrolních orgánů společnosti, aby nemusely neustále kontrolovat, co se děje na jednotlivých pracovištích. Namísto toho je systém bude automaticky upozorňovat na určité události, které nastanou v provozu. Upozornění může probíhat formou aktivního dialogu na určitou stanici, aktivním dialogem určitému uživateli, SMS zprávou nebo e-mailem.

Administrátor má možnost nastavovat jednotlivé reakce na události. U každé události nastavuje formu reakce a obsah zprávy, která je uživateli odeslána.

Události, na které je možné reagovat:

Byl překročen normovaný čas přípravný	
Kritérium	
O kolik byl překročen	V procentech
Konkrétní pracoviště	Výčet
Konkrétní typ zakázky	Výčet
*Proměnné do zprávy**	
O kolik procent byl překročen	%prep_time_overstep%
ID pracoviště	%work_station_id%
ID zakázky	%task_id%

Tabulka 5.1: Tabulka příkazů pro překročení normovaného času přípravného

* pravý sloupeček obsahuje řetězec, za který je nahrazena hodnota. Tento řetězec může být použit při tvoření zprávy.

Byl překročen normovaný čas výrobní	
Kritérium	
O kolik byl překročen	V procentech
Konkrétní pracoviště	Výčet
Konkrétní typ zakázky	Výčet
Ne/Nachází se zakázky na kritické cestě	T/F
*Proměnné do zprávy**	
O kolik procent byl překročen	%proc_time_overstep%
ID pracoviště	%work_station_id%
ID zakázky	%task_id%
Počet vyrobených kusů	%pcs%
Počet vyrobených zmetků	%waste%

Tabulka 5.2: Tabulka příkazů pro překročení normovaného času výrobního

Porucha stroje	
Kritérium	
Konkrétní pracoviště	Výčet
*Proměnné do zprávy**	
ID pracoviště	%work_station_id%
ID rozpracované zakázky	%task_id%

Tabulka 5.3: Tabulka příkazů pro poruchu stroje

Příliš vysoká zmetkovost	
Kritérium	
Zmetkovost	V procentech
Konkrétní pracoviště	Výčet
Konkrétní typ zakázky	Výčet
*Proměnné do zprávy**	
Zmetkovost v procentech	%prep_time_overstep%
ID pracoviště	%work_station_id%
ID zakázky	%task_id%

Tabulka 5.4: Tabulka příkazů pro příliš vysokou zmetkovost

5.8 Upozorňování uživatelů na kritickou cestu

Zakázky, které se nacházejí na kritické cestě, velmi ovlivňují celkovou dobu trvání výroby. Je nutný zvýšený dohled na tyto části výrobního procesu. Systém musí podporovat zobrazování kritické cesty a tento fakt může být též zobrazován i pracovníkovi, který na

daném úkolu pracuje. Systém může upozorňovat pracovníka informačním dialogem nebo jinou barvou pozadí klientské aplikace.

5.9 Sledování výrobků a jejich rodokmene

Rodokmen koncového produktu je tvořen všemi jeho díly a počátečními surovinami. Systém musí umožňovat přiřadit ke každému vstupnímu materiálu (prefabrikovanému, nebo vyrobenému) evidenčí číslo, pod kterým je materiál veden. Při tvoření záznamu o vytvoření nového výrobku zadá pracovník evidenční čísla do systému a tím se k nově vzniklému výrobku tato čísla přiřadí. Sýstém je pak zpětně schopen vytvořit rodokmen, ze kterého daný výrobek vznikl. Tímto způsobem se dají zpětně dohledávat záznamy k dílům, které jsou často reklamovány, nebo jsou poruchové.

Kapitola 6

Nefunkcionální požadavky

Systém musí splňovat všechny následující nefunkcionální požadavky:

- Systém musí fungovat jako klient-server aplikace.
- Klientská aplikace musí fungovat na všech prohlížečích, které splňují standard HTML5 (konkrétně Internet Explorer 11+, Google Chrome, Mozila firefox, Chrome for Android).
- Klientská aplikace musí fungovat i při náhodném přerušení spojení se serverem (například důsledkem výpadku sítě).
- Klientská aplikace musí fungovat na přenosných zařízeních (tabletech).
- Klientská aplikace musí být ovládatelná pomocí čtečky čárových kodů.

6.1 Fungování klientské části ve webových prohlížečích

Značná část společností, při nasazování počítačů do výroby, častokrát používá staré běžné kancelářské stolní počítače a nasazuje je do výrobny. Tato varianta je velmi levná oproti zavedení speciálního industriálního hardwaru, který je ale odolnější a do prostředí vhodnější.

Obě varianty však mají společný základ v tom, že obsahují operační systém, na kterém lze nainstalovat webový prohlížeč, který splňuje standard *HTML5*. Klientská aplikace fungující v prostředí webového prohlížeče zaručuje možnost použití již instalovaného hardwaru bez nutnosti obměny za hardware speciální.

6.2 Offline fungování klientské části systému

V industriálním prostředí může docházet k výpadkům sítě - přetrhnutí kabelu, WIFI technologie není příliš spolehlivá z důvodu vysokého rušení různým zářením ze strojů nebo důsledkem nekvalitního ovzduší. Klientská aplikace musí fungovat i při přerušení spojení se serverem. Přerušení spojení nesmí mít na činnost uživatele žádný vliv. Aplikace musí při výpadku konektivity plně fungovat nebo snížit omezení funkcionality na minimum. Tento fakt je oznámen uživateli vizuální změnou grafického prostředí, ale jeho práci to nesmí žádným způsobem limitovat.

6.3 Ovládání pomocí čtečky čárových kódů

Uživatelé častokrát nemají možnost používat jemnou motoriku rukou, aby interagovali s dotykovou obrazovku, mačkali tlačítka klávesnice nebo pohybovali myší (častokrát nosí rukavice nebo mají ruce špinavé). Oproti tomu čtečka čárových kódů je dostatečně velká, odolná a má pouze jediné tlačítko k ovládání, takže není problém s ní manipulovat v rukavicích. Z tohoto důvodu je nutné, aby uživatel mohl ovládat rozhraní pro zaznamenávání skutečnosti pouze pomocí čtečky čárových kódů.

Kapitola 7

Hardware

Klientská aplikace systému poběží v jakémkoli prohlížeči splňujícím standard *HTML5*. Touto vlastností se otevírá široká paleta zařízení, na které je možné klientskou aplikaci nasadit. Hlavní prioritou je možnost využití hardwaru, který je již použitý a umístěný na daném pracovišti. Současně by však měla být připravena hardwarová sestava, na které bude primárně systém testován a která bude splňovat všechny požadavky, čímž bude umožňovat kompletní funkčnost systému. Dalším důležitým parametrem je samozřejmě cena, která ovlivňuje výběr zařízení.

7.1 Tablet T70

Tablet T70 je voděodolný industriální tablet, který se hodí do prašného a pro normální zařízení nepříznivého prostředí. Důležitým parametrem je jeho odolnost vůči pádům a nárazům, čímž je značně snížená možnost rozbití či poškození při běžném provozu továrny.

Tablet obsahuje připojení Wifi:802.11 b/g/n pro připojení k síti a též rozhraní Bluetooth, které může být využito pro připojení periferií (jako je například čtečka čárových kódů).

Operačním systémem je Android 4.2 - Jelly Bean. Na tomto operačním systému je možné nainstalovat prohlížeč Google Chrome, který bude nastaven jako primární webový prohlížeč

splňující standard *HTML5* a na němž je možné spouštět klientskou aplikaci systému.

Tablet T70 je možné objednávat v několika barevných verzích (žlutá, oranžová, zelená) a je možné nechávat dodatečně doplnit některé periferie. Pro účely systému stojí za zmínku čtečka RFID čipů, které mohou sloužit k identifikačním účelům.

Obrázek 7.1: Tablet T70

7.2 Čtečka čárových kódů CT10

CT10 je lehká přenosná čtečka jednorozměrných čárových kódů. Tato čtečka je schopna propojení s počítačem pomocí kabelu (USB) nebo Bluetooth technologie. Důležitá schopnost tohoto zařízení je komunikovat přes Bluetooth i se zařízeními s operačním systémem Android. Pro případy výpadku spojení je čtečka vybavena interní pamětí, na kterou je možné zaznamenávat až 110 čárových kódů, které jsou následně odeslány jako jeden balík dat cílovému zařízení.

Přednostmi čtečky CT10 je hlavně možnost použití jak přenosného, tak statického zařízení. Zařízení není ani velké ani drahé.

Obrázek 7.2: Čtečka čárových kódů CT10

Kapitola 8

Implementace

Pro implementaci serverové části jsme zvolili technologie .NET, naimplementována byla v jazyce C#. Klientská část používá technologie *HTML5*, *CSS3* a *JavaScript*.

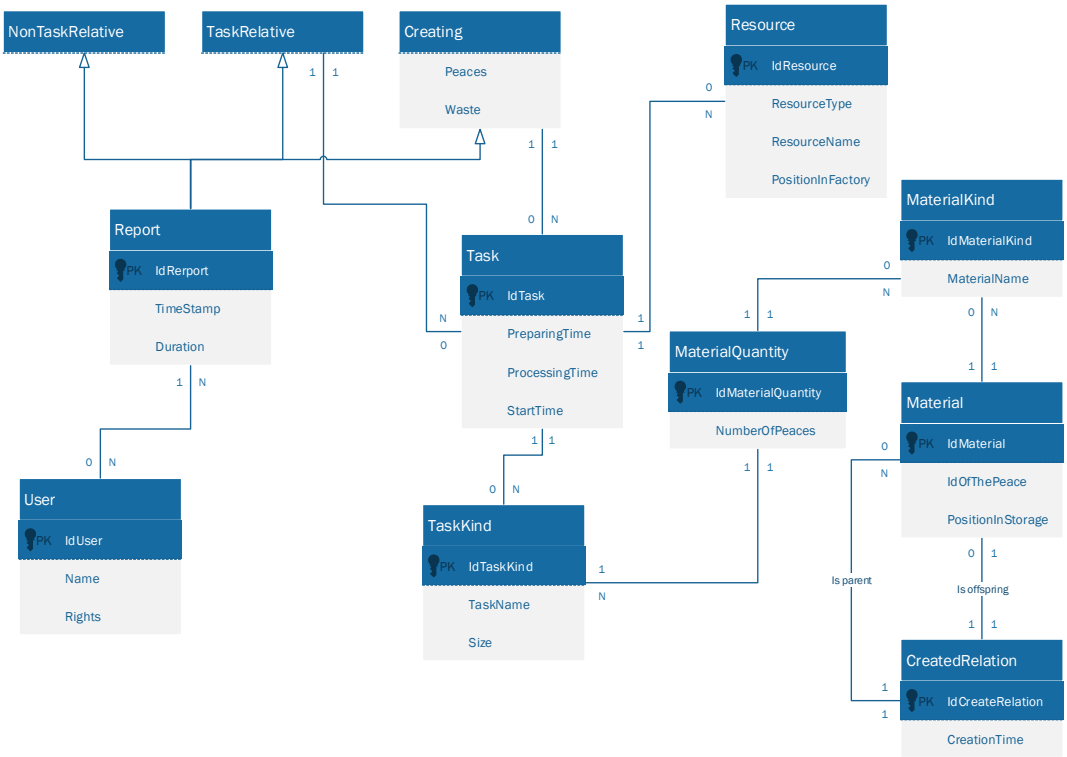
8.1 Server

Serverová část používá architerkturu MVC (Model-View-Controller). Tato architektura se stala trendem moderní doby a dnes bychom už těžko hledali aplikaci, která ji nepoužívá. Má jednu velmi důležitou vlastnost a to přesné oddělení dat, grafiky, aplikační logiky a správy toku událostí. Důsledkem této vlastnosti je jednoduchá implementace a rozšířitelnost, jednoduché a přehledné pokrytí testy a snižuje se tak duplicita kódu.

Pro implementaci *Modelu* byl zvolen .NET Entity Framework, což je otevřená knihovna pod Apache 2.0 licencí. Tato knihovna zajišťuje spojení s databází a její mapování na entity, které lze obsluhovat pomocí C# kódu. Knihovna spravuje vytváření entit, jejich provázání a jejich logiku. S tímto frameworkem mohou programátoři pracovat na vyšší úrovni abstrakce a mohou tak jednodušeji tvořit datově orientované aplikace.

8.1.1 Datová struktura pro zaznamenávání skutečnosti

Systém přijímá rozvrh pomocí rozhraní (viz Kapitola 5.5) a je nutné si ho někam uložit, aby do něj mohla být zaznamenávána skutečnost o provádění zakázek. Zde je vysvětlení jednotlivých entit z Obrázku 8.1.



Obrázek 8.1: Datová struktura pro uchovávání objednávek a zaznamenávání skutečnosti

- *User* - Entita obsahující data o uživateli, jakými jsou jméno a práva pro jednotlivé operace (jednotlivé druhy práv jsou vypsány v Kapitole 5.1).
- *TaskKind* - Entita obsahující data o druhu zakázky. Tato entita obsahuje vazby na entitu udávající potřebné množství materiálu (*MaterialQuantity*) k uskutečnění zakázky a na každou instanci zakázky, která je v rozvrhu (*Task*).
- *MaterialKind* - Entita reprezentující druh materiálu, který může vstupovat do výroby.
- *Material* - Entita reprezentující konkrétní instanci materiálu. Tato entita obsahuje

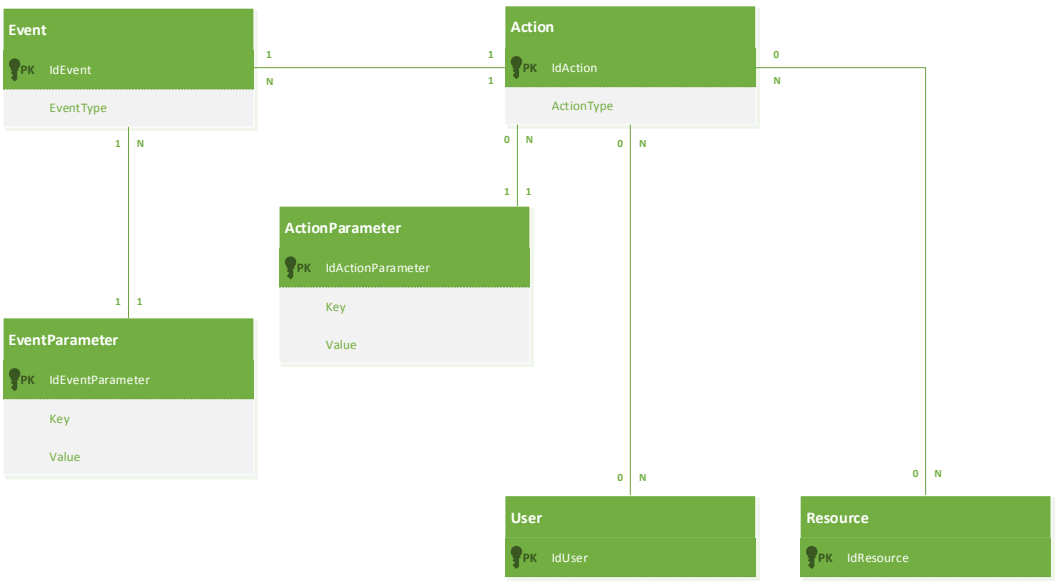
vazbu na *MaterialKind*, který určuje typ materiálu. Současně však obsahuje vztah sama k sobě, vyjádřený pomocí *CreatedRelation*, který určuje vztah mezi "rodičovským" materiálem a produktem.

- *Task* - Entita reprezentující instance jednotlivých rozvržených zakázek. Každá tato instance je již rozvržena na konkrétní zdroj.
- *Resource* - Entita, která reprezentuje stroje a zařízení, které jsou na výrobním patře.
- *Report* - Entita, která umožňuje zaznamenávat skutečnost o provedení jednotlivých zakázek. Tato entita má tři potomky (toto dělení koresponduje s rozdělením v Kapitole 5.3).
  - *Creating* - Tento potomek má vazbu na konkrétní zakázku a současně obsahuje dva parametry, počet vyrobených kusů a počet zmetků.
  - *TaskRelative* - Tento potomek má pouze vazbu na zakázku, ale neobsahuje žadná další data (reprezentuje činnosi, které mají vazbu k zakázce, ale nevzniká jimi nový produkt).
  - *NonTaskRelative* Tento potomek nemá vazbu na zakázku (reprezentuje neproduktivní činnosti).

8.1.2 Datová struktura pro uložení událostí a akcí

Systém umožňuje reagovat na určité události ve výrobě. Na Obrázku 8.2 je znázorněn diagram, který popisuje uchovávání konfigurace.

- *Event* - Entita, která reprezentuje konfiguraci události. Entita obsahuje typ události a má vazbu na své parametry (*EventParameter*) a na akce (*Action*), které mají být provedeny poté, co událost nastane.
- *EventParameter* - Entita reprezentující konfigurační parametr pro události. Parametr obsahuje klíč a hodnotu.
- *Action* - Entita, která reprezentuje konfiguraci akce, která se má provést po náležité akci. Entita obsahuje typ akce, vazbu na své parametry (ActionParameter) a odkazy na entity, se kterými je možné interagovat (*UserAction*, nebo *ResourceAction*).



Obrázek 8.2: Datová struktura pro uchovávání konfigurace událostí a akcí.

- *ActionParameter* - Entita reprezentující konfigurační parametr pro akci. Parametr obsahuje klíč a hodnotu.
- *User* - Entita reprezentující uživatele. Tato entita je totožná s entitou *User* v Kapitole 8.1.1.
- *Resource* - Entita reprezentující stroj (respektive výrobní stanici). Tato entita je totožná s entitou *Resource* v Kapitole 8.1.1.

8.2 Komponenta pro čtení čárových kódů

Na základě nefunkcionálního požadavku, aby se klientská aplikace dala pohodlně ovládat pomocí čtečky čárových kódů, musí být vyvinut modul, který umožní zaznamenávání událostí vyvolaných čtečkou a jejich následnou obsluhu. Čtečkou se musí dát obsluhovat

rozhraní pro zaznamenávání skutečnosti a současně musí být umožněno načítání zakázek. To vše uvnitř webového prohlížeče, bez instalace softwarového dodatku nebo prohlížečového pluginu.

Čtečka CT10, ostatně jako většina čteček čárových kódů, se po připojení k počítači chová jako klávesnice. Hodnoty, které se předávají do počítače, a tedy i do prohlížeče, jsou čísla stisknutých kláves. Zde je nutné zdůraznit, že čtečka nepřepisuje čárový kód do počítače, nýbrž simuluje stisk klávesy. Důsledkem tohoto chování je, že pokud má uživatel nastavené jiné než anglické rozložení kláves, mohou se při přepisu objevovat nepatřičné znaky (například pří nastavení české klávesnice se namísto číslic zobrazují znaky s diakritikou, tedy místo *2* se zobrazuje *ě*).

Na základě těchto požadavků byla vyvinuta komponenta, která byla pojmenována BarcodeScannerComponent. Její základ je postaven na technologii jQuery UI, je tedy implementována formou widgetu. Podstata spočívá v obalení libovolného elementu DOMu a odchytávání veškerých událostí klávesnice. Tím je zabráněno závislosti na uživatelově znakové sadě. Komponenta kontroluje focus pro elementy, které obaluje, a díky tomu může bezpečně kontrolovat potřebné události. Komponenta BarcodeScannerComponent též umožňuje zobrazit nad obaleným elementem šedivé překrytí (overlay) a dialogové políčko, ve kterém uživatel zadává kódy viditelně.

8.2.1 Používání komponenty

Nejdříve proběhne obalení některého elemementu (nazývejme tento element “hlavní element”) komponentou BarcodeScannerComponent. Po inicializaci komponenty je obalený element graficky nezměněn. Ve chvíli, kdy uživatel stiskne některé tlačítko klávesnice (nebo jej stiskne čtečka čárových kódů), začne si komponenta (bez žádného vizuálního projevu) zaznamenávat pořadí stisků kláves do vnitřní paměti. Pokud na konci (nebo v průběhu) uživatel zmáčkne klávesu *ENTER*, komponenta vyvolá událost zadání kódu. Tuto událost může odchytávat aplikační logika a následně podle toho provádět akce. Pokud však od posledního stisku klávesy uplyne předem stanovená doba, je kompletní obsah vnitřní paměti vymazán, jako kdyby uživatel nikdy nic nenapsal. Tím se zamezí chybě, která by mohla

vzniknout mylnými stisky kláves.

Pokud uživatel stiskne klávesu *ENTER* bez jakéhokoliv přechozího zadání kódu, je vyvolána událost oznamující, že byl zmáčknut *ENTER*, ale nebyl zadán žádný kód. Tato událost může být obsluhována aplikační logikou zvlášť.

Pokud uživatel předá focus některému z potomků elementu, který je obalený komponentou BarcodeScannerComponent, je mu umožněno ovládat daný prvek, a to včetně zapisování znaků. Důležité však je, že je tento focus po uplynutí časové prodlevy od posledního zmáčknutí klávesy vrácen zpět hlavnímu elementu.

Komponenta ještě nabízí uživateli zobrazit pole pro viditelné zadávání kódu. V takovou chvíli se zobrazí zadávací okénko, ve kterém se zobrazuje již zapsaný kód. Pokud je toto okénko zobrazeno, neprobíhá automatické mazání paměti, tudíž může uživatel psát kód libovolně dlouhou dobu.

8.2.2 Dokumentace k rozhraní

Komponenta používá technologii jQuery UI. Při inicializaci není potřeba předávat žádné povinné parametry. Volitelně lze nastavit:

- *prefix* - každý z prvků, který je vytvořen komponentou, dostává svoje unikátní ID (jelikož to DOM vyžaduje). Toto ID vždy začíná prefixem, aby se zamezilo jejich kolizi. Pokud by se však na jedné stránce nacházelo více komponent stejného druhu, musejí se i ID těchto dvou komponent lišit, tedy musí mít jiný prefix. (Výchozí hodnota: "BCSC001")

- *focusResetTimeout* - Tato proměnná určuje, za jak dlouho bude focus vrácen zpět hlavnímu elementu. Doba je udávána v milisekundách. (Výchozí hodnota: 5000)

- bufferResetTimeout - Tato proměnná určuje, za jak dlouho se vyresetuje obsah paměti, pokud se do něj již nepřidávají další znaky. Doba je udávána v milisekundách. (Výchozí hodnota: 3000)

- *charset* - Komponenta nemusí zaznamenávat všechny klávesy a tudíž ani všechny znaky. V charsetu je možné nastavit, které klávesy jsou zaznamenávány a co jejich stisk znamená pro výsledný kód. Pokud je charset na výchozí hodnotě, jsou zaznamenávány písmena A-Z a číslice 0-9.
- *localization* - Lokalizace potřebných textů.

Rozhraní, kterým je možné ovládat komponentu BarcodeScannerComponent nabízí několik základních metod:

- *focus()* - Bezparametrická metoda, která předá focus hlavnímu elementu.
- *showCodeInserter()* - Bezparametrická metoda, která zahájí viditelné zadávání kódů. Tento mód se sám automaticky ukončí po stisku kláves *ENTER* nebo *ESCAPE*.
- *hideCodeInserter()* - Bezparametrická metoda, která vypne viditelné zadávání kódů.
- *getEventSlots()* - Vrací objekt, do kterého se přidávají posluchače na jednotlivé události. Komponenta vyvolává celkem pět typů událostí.
  - charAdded - Událost je vyvolána při přidání znaku do paměti.
  - nontriggredKeyPressed - Událost je vyvolána v případě, že je zmáčknuta klávesa, ale není pro ni definované mapování na žádný znak.
  - noCodeReaded - Událost je vyvolána při stisku klávesy *ENTER*, ale v paměti není žádný kód k vrácení.
  - codeReaded - Událost je vyvolána při stisknutí klávesy *ENTER* a v paměti je kód, který byl doposud načten.
  - dialogClosed - Událost je vyvolána stiskem klávesy *ESCAPE* při zapnutém viditelném zadávání kódu.

8.3 Multimediální komponenta

Každý z uživatelů systému musí mít okamžitý přístup k dokumentům a výukovým videům, které se týkají jeho práce. Je běžné, že si pracovník v průběhu vykonávání zakázky

```
<video width="640" height="480">
        <source src="sample.mp4" type="video/mp4">
        <source src="sample.ogg" type="video/ogg">
        Your browser does not support the video tag.
</video>
```

Obrázek 8.3: Ukázka implementace videa v *HTML5*

otevře potřebnou dokumentaci a podle ní se řídí. Za tímto účelem byla vyvinuta komponenta, které se předloží dokument ve formátu *.PDF nebo video ve formátu *.MP4 a *.OGG a ona umožní jednoduché zobrazení a procházení souborem.

Komponenta, která získala pracovní název MultimediaComponent, je stejně jako komponenta pro zpracování čárových kódů (viz Kapitola 8.2) tvořena jako jQuery UI widget. Multimediální komponenta obaluje libovolný prvek DOMu a vytváří z něj prostředí pro zobrazení potřebného obsahu. Komponenta může běžet ve dvou základních režimech - Režim zobrazující video a režim zobrazující *.PDF dokumenty.

8.3.1 Režim zobrazující video

MultimediaComponent sama o sobe neobsahuje žádný přehrávač videa. Využívá standardu *HTML5*, který podporuje prvek přehrávající video.

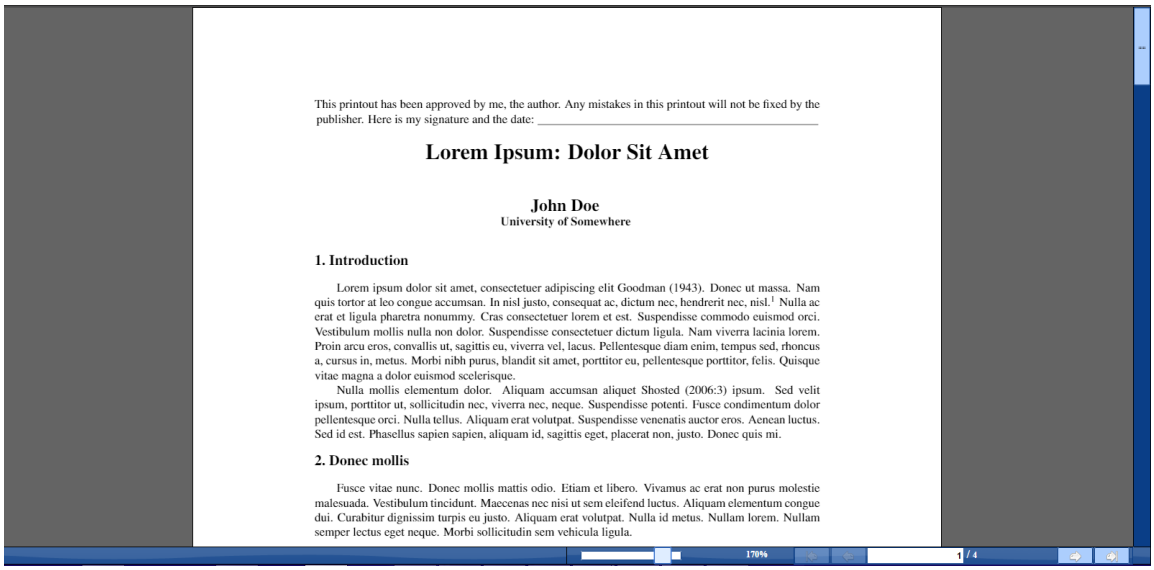
8.3.2 Režim zobrazující *.PDF soubory

Pro zobrazení *.PDF souboru pomocí javascriptu je použita knihovna PDF.js[15]. Autorem knihovny je Andreas Gal, který ji jako experimentální open-source knihovnu začal vyvíjet v roce 2011. V dnešní době je knihovna pod záštitou Mozila Foundation šířena pod Apache Licencí v2.0 a je ji možné použít pro zobrazování PDF souborů v HTML stránce.

Knihovna vykresluje asynchronně jednotlivé stránky pomocí HTML prvku *canvas*. Ve chvíli, kdy je vykreslena jedna stránka, je zavolána metoda pro vykreslení další. Tento způsob

vykreslování má výhodu v tom, že neblokuje načítání stránky, tedy uživatel stále vidí, že stránka stále reaguje. Na druhou stranu je těžké zajistit uživatelsky přívětivé překreslení stránek bez blikání a přeskakování prvků.

Režim zobrazující *.PDF soubory v MultimediaComponent vykreslí jednotlivé stránky, každou do jednoho prvku domu *canvas* a obalí je jednoduchými posuvníky, pouvníkem zoomu a kontrolnímy prvky na stránkování. Výsledek je možné vidět na Obrázku 8.4.



Obrázek 8.4: Zobrazené PDF obalené multimediální komponentou

8.3.3 Dokumentace k rozhraní

Komponenta používá technologii jQuery UI. Při inicializaci je potřeba vložit cestu k souboru, který má být zobrazen. Pokud se cílový soubor načítá z URL adresy, která nekončí příponou, je nutné vyplnit i proměnnou *fileExcention*. Pokud URL končí příponou souboru, komponenta se přetvoří sama.

- *prefix* - Prefix pro unifikaci ID jednotlivých prvků (stejně jako u BarcodeCannerComponent - viz Kapitola 8.2.2). (Výchozí hodnota: “MC001”)
- ***path*** - Cesta k souboru, který by měl být načtený. (Tento parametr je povinný.)

- *pageBorder* - Tloušťka rámečku okolo každé stránky *.PDF dokumentu. Hodnota je udávána v pixelech. (Výchozí hodnota: 1)
- *gapBetweenPages* - Mezera mezi stránkami *.PDF dokumentu. Hodnota je udávána v pixelech. (Výchozí hodnota: 5)
- *fileExtension* - Přípona souboru, který je zobrazovaný multimediální komponentou. Pokud tato hodnota není udána při inicializaci komponenty, je přečtena z přípony souboru, který má být zobrazován. Možné hodnoty jsou "pdf" a "mp4".
- *localization* - Lokalizace potřebných textů.

Rozhraní, kterým je možné ovládat komponentu MultimediaComponent nabízí několik základních metod:

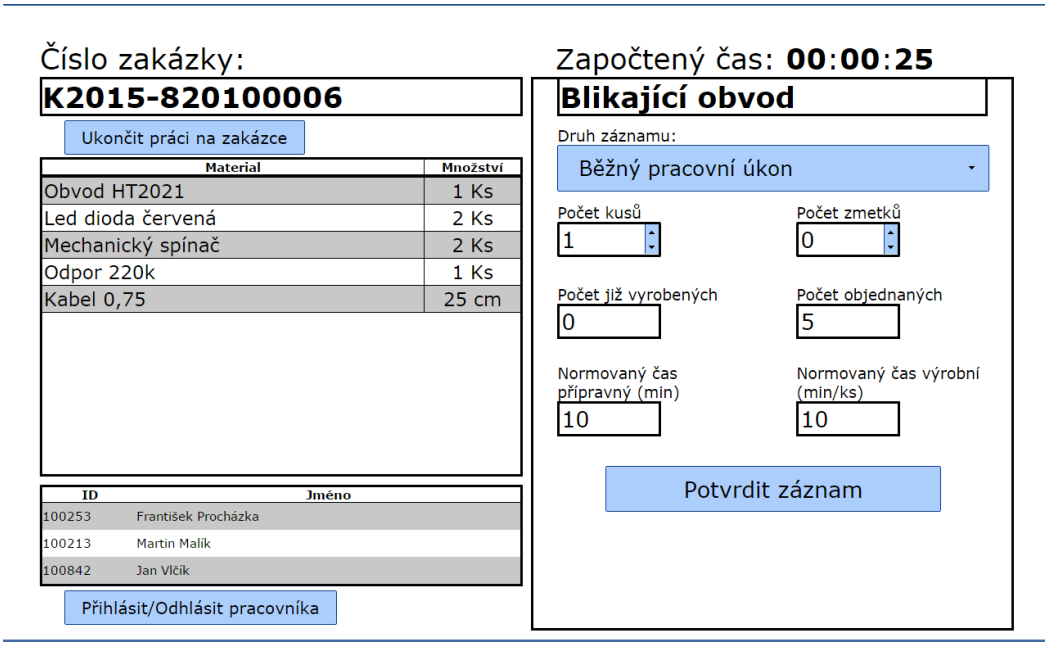
- Metody pro režim zobrazování *.PDF dokumentů:
  - setScroll(x, y) - Metoda, která nastaví posunutí dokumentu. Jako parametr přijímá dvě čísla, která označují posunutí horizontální a vertikální.
  - setScrollOnPage(pageNumber) - Metoda, která nastaví posunutí dokumentu na určitou stránku. Číslování stránek začíná od nuly.
  - setZoom(zoom) - Metoda, která nastaví přiblížení. Jako parametr přijímá jedno číslo - velikos přiblížení.
- Metody pro režim zobrazování videa:
  - play() - Zahájí přehrávání videa.
  - stop() - Zastaví přehrávání videa.
  - pause() - Pozastaví přehrávání videa.
- *getEventSlots()* - Vrací objekt, do kterého se přidávají posluchače na jednotlivé události. Komponenta vyvolává několik druhů událostí:
  - zoomChanged - Změnilo se přiblížení. (Tato událost je vyvolána, pokud je komponenta v režimu zobrazování *.PDF souborů)

– scrollChanged - Změnil se posunutí v dokumentu. (Tato událost je vyvolána, pokud je komponenta v režimu zobrazování *.PDF souborů.)

– události *HTML5* video kontejneru - pokud je komponenta v režimu videa, vyvolává všechny události, které vyvolává *HTML5* standardizovaný kontejner *video*.

8.4 Rozhraní pro zaznamenávání skutečnosti

Pro zaznamenávání dat z výroby obsahuje systém obrazovku, která se uživateli zobrazí ve chvíli, kdy začne pracovat na zakázce (viz Obrázek 8.5). Tato obrazovka obsahuje informace o zakázce a možnost zadání jednotlivých akcí, které byly na obrazovce vykonány. Současně uživatel vidí, kolik času strávil na jednotlivých pracovních úkonech. Cele rozhraní pro zaznamenávání skutečnosti je obaleno komponentou BarcodeScannerComponent (viz Kapitola 8.2), takže její ovládání je možné pomocí čtečky čárových kódů. Současně je možné pomocí čtečky čárových kódů vyplňovat konkrétní materiály, které vstupují do výroby, a tím tak tvořit rodokmen výrobku.



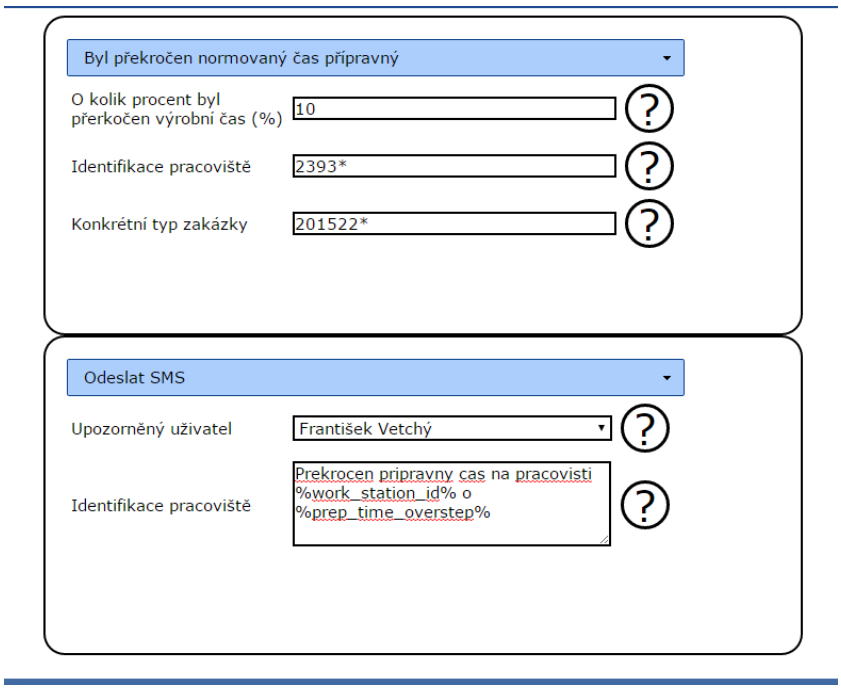
Obrázek 8.5: Rozhraní pro zaznamenávání skutečnosti

8.5 Události a akce

Pokud má uživatel administrátoská práva, může vkládat do systému akce, které se spouští na základě předem daných událostí (viz Obrázek 8.6). Tímto způsobem může řídící orgán společnosti být permanentně informován o událostech na výrobním patře, které se vymykají rozvrhu.

Nejdříve uživatel zvolí jaký druh události musí nastat, aby byly provedeny patřičné akce. Výčet události je možný nalézt v Kapitole 5.7. Po zvolení druhu události se uživateli umožní volba parametrů události. Pro určení identifikace zakázky nebo pracoviště může uživatel použít jednoduchou sadu regulárních výrazů ("*?*" pro nahrazení libovolného znaku a "***" pro nahrazení libovolného počtu libovolných znaků).

Následně může uživatel přiřadit k události libovolný počet libovolných akcí. U akce zvolí uživatele nebo konkrétní stanici, kam se má zpráva odeslat, a následně napíše obsah zprávy. Do zprávy lze vyplňovat řetězce, které jsou při vyvolání události substituované za konkrétní hodnoty.

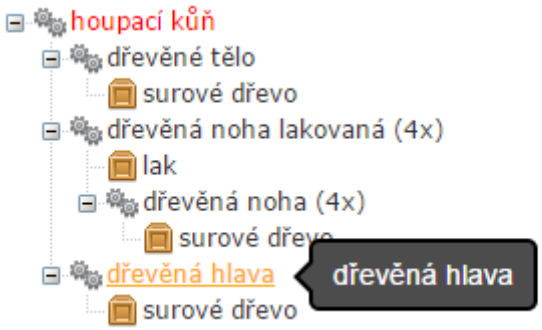


Obrázek 8.6: Ukázka vkládání akce v závislosti na události

8.6 Zobrazování rodokmene

Modul pro zobrazování rodokmene umí na základě identifikačního čísla výrobku najít jeho rodokmen a zobrazit jej (viz Obrázek 8.7). Po rozkliknutí se zobrazí detaily vzniku jednotlivých částí výrobku (například: kdo jej vyrobil, kdy byl vyroben a na jakém stroji). Rodokmen rozeznává dva druhy uzlů:

- Surovina - Daná surovina byla přivezena, nebo vyrobena mimo továrnu, kterou pokrývá tento systém (v rodokmenu je znázorněn ikonkou hnědé bedýnky).
- Výrobek - Daný výrobek byl vyroben v této továrně a může obsahovat svůj rodokmen (v rodokmenu je znázorněn ozubenými kolečky).



Obrázek 8.7: Ukázka jednoduchého rodokmenu

Kapitola 9

Testování

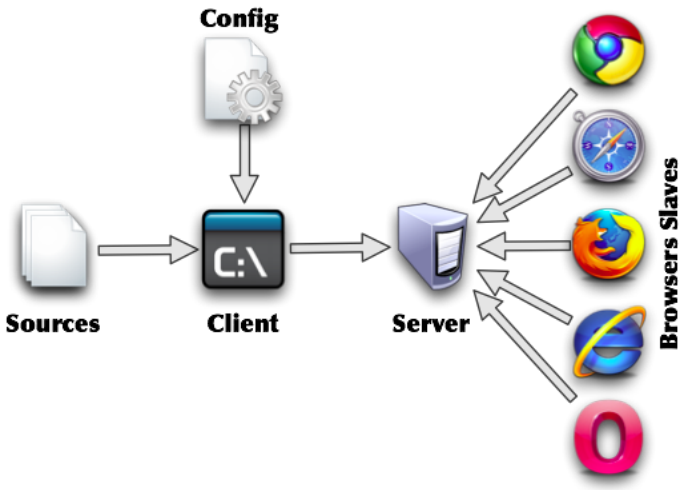
Veškerý kód je pokrytý standardní sadou Unit testů. Tvoření Unit testů pro C# část aplikace (tedy serverovou část) jsou použité standardní nástroje pro testování uvnitř Visual Studia. Pro testování JavaScript kódu (tedy klientské části) je použita knihovna js-test-driver (viz Kapitola 9.1). Pro účely testování klientské části byly zvoleny prohlížeče:

- Google Chrome (verze 42.0.2311.135 m)
- Mozila Firefox (verze 36.0.1.20150305021524)
- Internet Explorer 11 (verze 11.0.9600.177728)

9.1 Testovací knihovna js-test-driver

Cílem knihovny js-test-driver[2] je vytvoření prostředí pro jednoduché testování kódu psaného v jazyce JavaScript. Knihovna je psána v jazyce Java a je spouštěna pomocí příkazové řádky. Jakmile je knihovna spuštěna, vytvoří na lokální stanici uživatele server, ke kterému se připojují jednotlivé prohlížeče (architektura je vidět na Obrázku 9.1). Do těchto prohlížečů je následně nahráván testovaný kód s testy. O tom, který kód bude nahrán do prohlížečů a které testy na něm budou spuštěny rozhoduje konfigurační soubor (standardní

název je “jsTestDriver.conf”). Příklad konfiguračního souboru můžeme vidět na Obrázku 9.2.



Obrázek 9.1: Schéma propojení pro knihovnu js-test-driver

9.2 Metoda Latinských čtverců

Při testování metod nebo částí programu je potřeba vyzkoušet, jak reaguje daný kód na jednotlivé vstupy. V ideálním případě se vyzkouší všechny možné kombinace proměnných a jejich významných hodnot. Takovýto počet testu není reálně možný výkonově zvládnout. Pokud si představíme situaci, že daná metoda má 10 různých vstupů a každý vstup má 8 významných hodnot, celkový počet testů tedy je

$$8^{10} = 1073741824$$

```
server: http://localhost:9876
load:
- Demo/Libs/Scripts/*.js
- Scripts/*.js
test:
- Tests/*.js
```

Obrázek 9.2: Ukázka konfiguračního souboru js-test-driver

Nutné tedy je se spokojit s podmnožinou, která není výpočetně tak náročná. Současně ale musíme vybírat takovou podmnožinu testů, která může odhalit co největší množství chyb. Praxe ukazuje, že testování všech dvojic (takzvané *all-pairs testing*) je velmi dobrou strategií. Tato strategie vychází z myšlenky, že velmi častý zdroj chyb zavisí na souhře dvou parametrů. Chyby vyvolané trojicí nebo vyšší n-ticí parametrů jsou mnohem méně časté. Pokud se vrátíme k našemu předchozímu příkladu, pomocí této redukce se dostaneme na $8^2 = 64$ testů, což je o 8 řádů méně. Pro takovouto redukci se používá metoda latinských čtverců.

Latinské čtverec je matice s n řádky a n sloupci. Každý element obsahuje čísla od 1 do n tak, že ve všech sloupcích a řádcích je výskyt každého čísla maximálně jednou. Pokud nalezneme dva čtverce takové, že v daných čtvercích je každý element v relaci s každým jiným elemementem právě jednou, pak tyto čtverce nazýváme *párově ortogonální latinské čtverce*. Pro každé prvočíslo N, nebo jeho mocninu, existuje N - 1 párově ortogonálních latinských čtverců velikosti $N \times N$.

Pro konstrukci latinských čtverců je nutné si nejdříve zvolit n, které je prvočíslem. Následně jsme schopni vygenerovat čtverce pomocí vzorce (pro $k,\ i,\ j \in \{1, 2, \cdots, n\}$).:

$$A_k(i,\ j) = [k \cdot (i-1) + (j-1)] \bmod n$$

9.3 Testování pomocí latinských čtverců

Metodu latinských čterců použijeme pro otestování výše popsané komponenty pro čtení čárových kódů (viz Kapitola 8.2). Jako první je nutné zvolit, jaké akce budeme testovat. Tyto akce jsou v Tabulce níže (9.1).

Testování budeme provádět tak, že se vždy vybere kombinace pěti akcí a ty se provedou. Na konci každé sekvence akcí uživatel vždy stiskne klávesu enter. Následně bude ověřeno, zdali komponenta vyvolala správné události. Pro provádění akcí nebudeme potřebovat reálného uživatele, ale použijeme opět js-test-drive framework, tudíž tyto testy budou automatizované.

ID akce	Název akce	Popis akce
0	Zadání známého znaku	Uživatel stiskne klávesu na klávesnici a znak na klávese je zadaný v překladovém slovníku komponenty (písmeno M).
1	Zadání neznámého znaku	Uživatel stiskne klávesu na klávesnici, ale znak na klávese není zadaný v překladovém slovníku komponenty (klávesa s tečkou).
2	Zobrazení viditelného zadávání znaků	Uživatel zapne mód pro viditelné zadávání kódu (tím se vypne automatické vymazávání vnitřní paměti).
3	Zmáčknutí klávesy *Escape*	Uživatel stiskne klávesu *Escape*. Tím se vypne mód pro viditelné zadávání kódu (pokud je zapnutý) a vymaže se vnitřní paměť.
4	Čekání dostatečně dlouhou dobu	Uživatel neprovádí žádnou akci pro dostatečně dlouhou dobu (6 sec), aby se vymazala vnitřní paměť.

Tabulka 9.1: Tabulka testovaných akcí

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 4 & 1 & 3 \\ 1 & 3 & 0 & 2 & 4 \\ 2 & 4 & 1 & 3 & 0 \\ 3 & 0 & 2 & 4 & 1 \\ 4 & 1 & 3 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 3 & 1 & 4 & 2 \\ 1 & 4 & 2 & 0 & 3 \\ 2 & 0 & 3 & 1 & 4 \\ 3 & 1 & 4 & 2 & 0 \\ 4 & 2 & 0 & 3 & 1 \end{bmatrix}$$

Obrázek 9.3: Použité latinské čtverce

K utvoření sekvencí testu bude potřeba tří latinských čtverců o velikosti 5×5. Sestavení sekvence proběhne následovně (pro $i,\ j \in \{0, 1, 2, 3, 4\}$):

- První akce je číslo i.
- Druhá akce je číslo j.
- Třetí akce je číslo na pozici $[i; j]$ v prvním čtverci.
- Čtvrtá akce je číslo na pozici $[i; j]$ v druhém čtverci.
- Pátá akce je číslo na pozici $[i; j]$ v třetím čtverci.

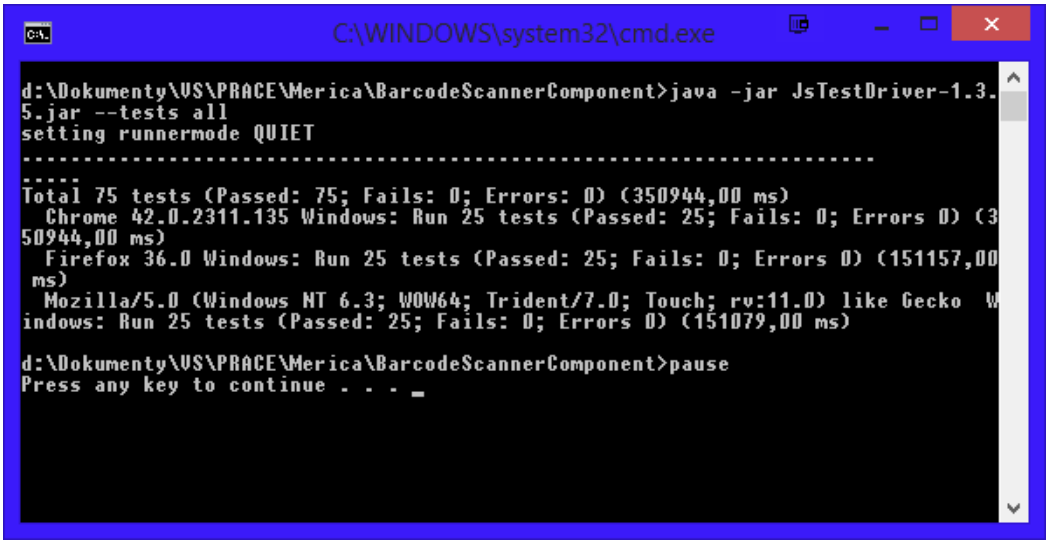
1	0	0	0	0	0
2	0	1	1	2	3
3	0	2	2	4	1
4	0	3	3	1	4
5	0	4	4	3	2
6	1	0	1	1	1
7	1	1	2	3	4
8	1	2	3	0	2
9	1	3	4	2	0
10	1	4	0	4	3
11	2	0	2	2	2
12	2	1	3	4	0
13	2	2	4	1	3
14	2	3	0	3	1
15	2	4	1	0	4
16	3	0	3	3	3
17	3	1	4	0	1
18	3	2	0	2	4
19	3	3	1	4	2
20	3	4	2	1	0
21	4	0	4	4	4
22	4	1	0	1	2
23	4	2	1	3	0
24	4	3	2	0	3
25	4	4	3	2	1

Tabulka 9.2: Tabulka sekvencí testů

Pro lepší představu, jak bude výsledný test vypadat, si můžeme vzít test číslo 15 jako příklad (v Tabulce 9.2 vyznačený tyrkysovou barvou).

1. Uživatel zapne mód pro viditelné zadávání kódu.
2. Uživatel počká 6 vteřin.
3. Uživatel stiskne klávesu s tečkou.
4. Uživatel stiskne klávesu “M”.
5. Uživatel počká dalších 6 vteřin.
6. Uživatel stiskne klávesu *ENTER*. Tento krok je proveden pro každý test, tudíž není vyznačen v tabulce.

Výsledkem tohoto testu bude vyvolání události, že uživatel zadal jeden znak (písmeno “M”) a následně stisknul *ENTER*.



Obrázek 9.4: Výstup testů provedených pomocí knihovny js-test-drive

9.4 Manuální testování

Na tabletu T70 není možné spustit js-test-driver, tudíž testování aplikace muselo být prováděno ručně. Výsledky testů korespondují s výsledkem prohlížeče Google Chrome pro desktop.

Kapitola 10

Závěr

Problematika MES systémů je a bude stále aktuální, protože každý továrník bude chtít uspokojit poptávku pro co nejvíce zákazníků, tím pádem bude vyvíjen větší tlak na výrobní patro. MES systémy budou obsahovat více funkcionality, která bude zvyšovat výkonnost a efektivitu práce.

Systém se podařilo naimplementovat a otestovat. Nejvíce problémů bylo s kompatibilitou čtečky čárových kódů, která se chovala místy nepředvídatelně. Výsledným řešením je komponenta pro čtení čárových kódů (viz Kapitola 8.2), která nakonec odstínila aplikační logiku od problému s kompatibilitou.

10.1 Možné rozšíření

Systém vytvořený touto prací je navržen tak, aby bylo možné jej jednoduše rozšiřovat. Možným rozšířením je například přidání samotného algoritmu pro rozvrhování prací. Toto rozšíření by vyžadovalo obohacení datového modelu, ale rozhodně by znamenalo velký krok dopředu a zvýšení možnosti využití softwaru jako samostatně stojícího systém.

Dalším rozšířením může být kontrola jakosti výrobků. Systém může uchovávat výsledky zátěžových testů na jednotlivých vzorcích a vyhodnocovat ideální parametry pro výrobu.

Dále je možné doplnit další klíčové výrobní ukazatele, které budou pracovat s výsledky kontroly jakosti.

Literatura

[1] AG, C. S., May 2015. The pro-active manufacturing execution system diames.
URL <http://www.diames.com/>

[2] code.google.com, May 2015. js-test-driver library.
URL <https://code.google.com/p/js-test-driver/>

[3] en.wikipedia.org, May 2015. Batch production.
URL <http://en.wikipedia.org/wiki/Batch_production>

[4] en.wikipedia.org, May 2015. Critical path method.
URL <http://en.wikipedia.org/wiki/Critical_path_method>

[5] en.wikipedia.org, May 2015. Job production.
URL <http://en.wikipedia.org/wiki/Job_production>

[6] en.wikipedia.org, May 2015. Mass production.
URL <http://en.wikipedia.org/wiki/Mass_production>

[7] en.wikipedia.org, May 2015. Mean time between failures.
URL <http://en.wikipedia.org/wiki/Mean_time_between_failures>

[8] en.wikipedia.org, May 2015. Mean time to repair.
URL <http://en.wikipedia.org/wiki/MTTR>

[9] Finsoft, s., May 2015. Software na řízení výroby maggio.
URL <http://www.finsoft.cz/software-rizeni-vyroby-maggio/>

[10] GmbH, M. M., May 2015. The mes experts - efficiencies in production with mes.
URL <http://www.mpdv.de/en.html>

[11] Kletti, J., 2007. Manufacturing Execution System - MES. Springer-Verlag Berlin Heidelberg.

[12] managementmania.com, May 2013. Metoda kritické cesty - cpm (critical path method). URL <https://managementmania.com/cs/metoda-cpm>

[13] mescentrum.cz, May 2015. Oee a odvozené ukazatele. URL <http://www.mescentrum.cz/o-projektu/90-mes/clanky/mes-mom/133-oee>

[14] mescenturm.cz, May 2015. Mes/mom řešení. URL <http://www.mescentrum.cz/katalog/porovnani>

[15] mozilla.github.io, May 2015. Pdf.js. URL <http://mozilla.github.io/pdf.js/>

[16] Unis, a., May 2015. O systému pharis. URL <http://www.pharis.cz/cs/o-systemu-MES-PHARIS>

[17] Štrublíková, I., 2008. Mes systémy ve strojírenství. Master's thesis, Brno University of Technology.

[18] w3schools.com, May 2015. Web service description language. URL <http://www.w3schools.com/webservices/ws_wsdl_intro.asp>

Příloha A

Obsah přiloženého CD

- *srcThesis* - Složka se zdrojovým kódem diplomové práce.
- *src* - Složka se zdrojovými kódy.
- *LStests.xlsx* - Tabulka generování testů pomocí metody latinských čtverců.
- *diplomaThesis.pdf* - Diplomová práce ve formátu *.PDF.