VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Ladislav Tylich

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

MODUL PRO PLÁNOVÁNÍ VÝROBY V MES
PRODUCTION SCHEDULING SUBSYSTEM FOR MES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Ladislav Tylich

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Václav Kaczmarczyk, Ph.D.

BRNO 2018

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. Ladislav Tylich ***ID:*** 154899
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Modul pro plánování výroby v MES

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout způsob plánování výrobních operací pro konkrétní úlohu a provést implementaci do existujícího systému pro řízení výroby (MES).

1. Seznamte se s problematikou MES systémů a zadokumentujte jejich vlastnosti.
2. Proveďte rešerši matematických metod pro plánování výrobních operací a zadokumentujte.
3. Pro konkrétní výrobní proces navrhněte optimalizační úlohu a proveďte simulace.
4. Seznamte se se současným stavem jádra poskytnutého MES.
5. Modifikujte jádro MES tak aby nebylo vázané na žádnou komerční knihovnu.
6. Navrhněte modifikace směřující k implementaci modulu pro plánování výroby do MES.
7. Zadokumentujte veškeré činnosti tak, aby na práci bylo možno navázat v budoucnu.

DOPORUČENÁ LITERATURA:

Pásek J.: Skripta Automatizace procesů

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 14.5.2018

Vedoucí práce: Ing. Václav Kaczmarczyk, Ph.D.
Konzultant:

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

ABSTRAKT

Cílem práce je seznámení s MES systémy, jejich vlastnostmi, vymezením hranic mezi ostatními systémy používaných v automatizaci. Dále je popsána problematika plánování výrobních operací. Je provedena rešerše matematických metod aplikovatelných pro plánování výrobních operací. Pro konkrétní optimalizační úlohu je navržen optimalizační model a jsou provedeny simulace. Jádro poskytnutého MES systému bylo přepsáno do webového rozhraní s nekomerční platformou a jsou uvedeny změny potřebné pro integraci modulu pro plánování výroby do systému.

KLÍČOVÁ SLOVA

MES, rozvrh výrobních operací, LP, rozvrh práce, body-shop scheduling

ABSTRACT

The objective of this thesis is to introduce MES systems with their properties and relations to other automation systems. Furthermore production scheduling theory is introduced with applicable mathematical methods. For given scheduling problem is created optimization model and basic serie of simulations is accomplished. The core of an existing MES system is transformed to web non-comercial platform. All necessary changes are listed in order to integrate production scheduling subsystem to the existing MES system.

KEYWORDS

MES, production scheduling, LP, job-shop scheduling, body-shop scheduling

TYLICH, Ladislav. *Modul pro plánování výroby v MES*. Brno, 2018, 91 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Václav Kaczmarczyk, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Modul pro plánování výroby v MES“ 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 ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Václavu Kaczmarczykovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat panu Ing. Radovanu Holkovi CSc. za pomoc při návrhu datových modelů a panu doc. RNDr. Jaromíru Baštincovi CSc. za trefné podněty při návrhu plánovacího algoritmu. Mé díky ale hlavně patří celé mé rodině za podporu během celého studia.

Brno ....

podpis autora

Obsah

[1]Program byl testován s databází MySQL, jelikož byl tento DB stroj vybrán pro budoucí aplikaci.

Seznam obrázků

Seznam tabulek

Úvod

Práce sestává z osmi kapitol. První kapitola je cílena na funkce MES systému od raných fází jeho vývoje. Následně jsou uvedeny hranice systému MES v kontrastu s dalšími systémy zajišťující fungování podniku jako celku, jejich možné překrývání a vstupně/výstupní toky dat. V další kapitole je popsána problematika plánování výrobních operací. Kapitola má za cíl uvést plánování a rozvrhování ve výrobě, ale zároveň tato kapitola uvádí plánování výrobních operací do souvislosti se systémem MES a dalšími systémy v historickém kontextu.
Následně je provedena rešerše matematických metod pro plánování výrobních operací a uvedeny možnosti aplikace vybranými výpočetními prostředky. Kapitola čtyři se zabývá rozborem konkrétního problému, aplikací matematických metod a provedením základních simulací. Kapitoly 5 až 7 jsou zaměřeny na transformaci původního jádra MES systému na nekomerční platformu.
V kapitole pět jsou shrnuty přínosy a stav poskytnuté verze MES, v kapitole šest je pak proveden návrh systému a kapitola sedm dokumentuje zprovoznění systému, zvolené frameworky a platformy. Následuje porovnání vlastností s původní verzí.
Poslední kapitola uceluje informace předchozích kapitol. Zabývá se kroky vedoucími k implementaci plánovacího modulu do systému MES s využitím výsledků simulací a informací o návrhu modelu rozvrhu.

1 MES systémy

V této kapitole budou popsány systémy *Manufacturing Execution System - Výrobní řídicí systém* (MES), jejich vývoj a funkce. Také budou uvedeny standardy, se kterými tyto systémy souvisí, případně navazující systémy MOM.

1.1 Historie

V období 70. - 80. let 20. století začaly pronikat počítače do průmyslu. Hlavní cíle byly zbavit se papírování (od různých náčrtů, pracovní instrukce, objednávky) a vytvoření nástroje pro dohled nad výrobním závodem. První takový zavedený systém se označoval jako *Material Requirements Planning - Plánování potřeb materiálu* (MRP), který samostatně vznikl již v 60. letech. Na rozdíl od ostatních způsobů plánování, nastupující MRP systémy měly plánovací časový horizont menší s rozdělením na intervaly. Zavedením logiky plánování potřeb materiálu tzn. požadavků na nákup materiálu, byly stanoveny očekávané materiálové příjmy a výrobní harmonogramy, posléze potom celkové finanční požadavky [1]. Tento proces byl ale v továrnách jakéhokoli rozsahu složitý a časově náročný:

- 1. týden - shromažďování dostupných dat (obvykle z minulého měsíce)
- 2. týden - manuální zadávání dat do počítačového systému, tisk reportů
- 3. týden - vytvoření informací, krátkých reportů (neaktuálních) z dostupných dat pro řídicí oddělení

Tento proces se samozřejmě opakoval od dalšího týdne dále. Jak je zřejmé, bylo nutné zajistit aktuální informace z továrny.
S příchodem počítačů do průmyslu, resp. řízení procesů pomocí počítačů koncem 80. let dalo vznik CIM - Computer-Integrated Manufacturing, tedy předchůdci MESu.

1.2 Příchod standardizace a vznik MES systémů

V roce 1992 došlo v USA ke vzniku neziskové organizace MESA (Manufacturing Enterprise Solution Association - Sdružení výrobních podnikových řešení), tedy organizace sdružující vývojáře systémů pro řízení výrobních závodů. To dalo vznik termínu MES (Manufacturing Execution System - Výrobní řídicí systém), kterým se od té doby tyto systémy označují, přestože jejich modifikace existovaly už dříve (MCS). Rozdíl byl v tom, že původní systémy byly většinou šité na míru zákazníkům a měly úzký rozsah použitelnosti [1]. Hlavním cílem MESA je vytvoření platformy pro sdílení zkušeností a znalostí o systémech MES a jejich uplatnění v praxi [2]. Ze světově známých členů (zároveň i sponzorů) organizace MESA lze jmenovat firmu Siemens nebo Rockwell Automation.

1.2.1 Vznik

MES jako takový vznikl v roce 1994. První zařazení systému MES v hierarchii podniku bylo v rámci tzv. *Kontextového modelu* v roce 1996, jehož hlavní charakteristikou je překrývání MES s ostatními systémy (*Enterprise Resource Planning - Plánování podnikových zdrojů* (ERP), SCM - řízení dodavatelského řetězce a procesní řízení - PLC, DCS).
Ke konci 90. let byly postupně uvedeny standardy ISA-88 a ANSI/ISA-95. Zatímco standard ISA-95 je zaměřen na výměnu informací mezi ERP a MES, ISA-88 je zaměřen na dávkové řízení a modulární rozdělení při řízení procesů. Přesto ale dochází ke vzájemnému překrytí standardů, kde společnou úrovní je procesní buňka/jednotka. Znamená to, že v rámci funkce MES systému se poskytují informace ERP až na úroveň procesní jednotky/buňky, detaily z procesního řízení jsou příliš podrobné [3].

1.2.2 Definice podle MESA

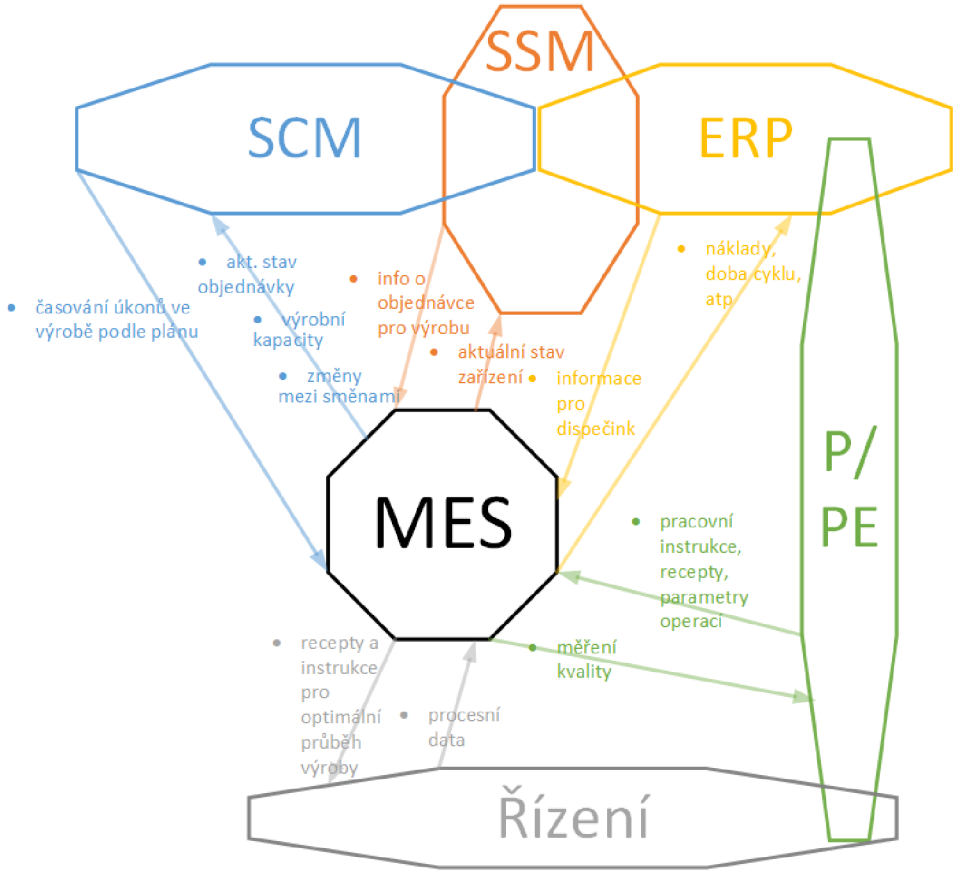
Systémy MES (Manufacturing Execution System - Výrobní řídicí systém) poskytují informace, které umožňují optimalizaci výrobních činností od uvedení objednávky do finalizace výrobků. S využitím aktuálních a přesných údajů MES řídí, iniciuje, reaguje a reportuje výrobní činnosti, jakmile nastanou. Výsledná rychlá odezva na měnící se podmínky, spojená se zaměřením na snížení aktivit bez člověkem přidané hodnoty, řídí efektivní provoz zařízení a procesů. MES zlepšuje návratnost provozních aktiv, stejně jako urychluje okamžité doručení, reakce na změny zásob, zvyšuje hrubý zisk a peněžní tok (*cash flow*). MES poskytuje důležité informace o výrobních činnostech v rámci celého podniku a dodavatelského řetězce prostřednictvím obousměrné komunikace [5].

1.3 MES v kontextu dalších systémů

MES je celopodnikový informační systém poskytující informace k efektivnímu vykonávání operací a naplnění obchodních cílů. Patří mezi několik hlavních typů informačních systémů cílených na výrobní společnosti. Každá z těchto kategorií systémů zahrnuje různé funkce a typy produktů. Následuje charakteristika hlavních kategorií souvisejících výrobních systémů [5], jejich návaznost na MES je znázorněna na modifikovaném kontextovém modelu na Obr.1.1):

- **ERP** (Enterprise Resource Planning - Plánování podnikových zdrojů) - zahrnuje finance, správu objednávek, plánování výroby a materiálů a související funkce

- **SCM** (*Supply Chain Management* - Řízení dodavatelského řetězce) - týká se předpovědi, distribuce a logistiky a řízení dopravy, elektronického obchodu a pokročilých plánovacích systémů (APS)
- **SSM** (Sales and Service Management - Řízení prodeje a služeb) - zahrnuje SW pro automatizaci prodeje, konfiguraci produktu, definici služeb, produktovou návratnost atp.
- **P/PE** (Product and Process Engineering - Výrobní a procesní inženýrství) - součástí jsou CAD/CAM software, modelování procesů a správa dat o produktech (PDM)
- **Řízení** (Controls) - hybridní systémy SW/HW, zahrnuje PLC, DCS, DNC a SCADA systémy a jiné automatizované jednotky pro řízení procesů



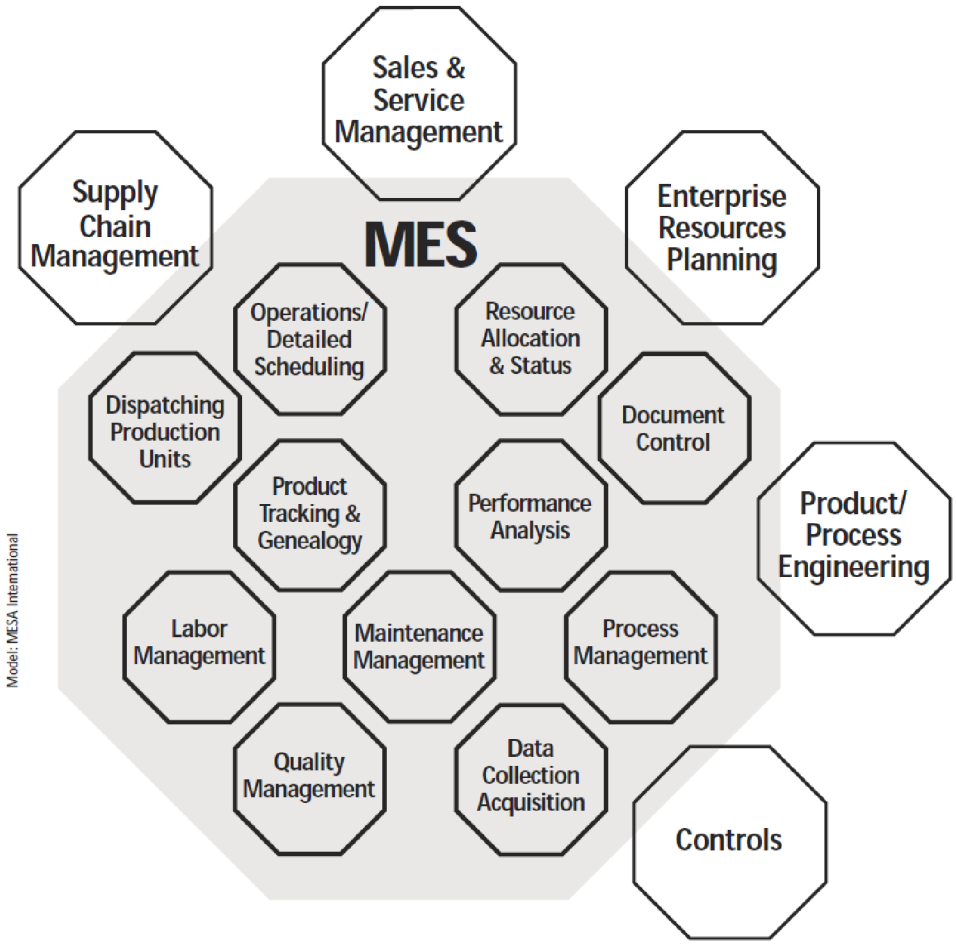
Obr. 1.1: Zobrazení konkrétních entit v rámci spolupráce MESu s dalšími typy systémů v podniku.

Většina společností potřebuje nějakou funkci z každé z šesti kategorií (spolu se systémem MES). Rozsah a detailní funkce požadovaná každou z kategorií aplikací se liší podle typu výroby (spojitá, diskrétní, dávková, montáž nebo smíšená) a stylu plánování (výroba na objednávku MTO, montáž na objednávku ATO, na základě

individuálního přístupu k zákazníkům ETO). Diferencovanost aplikace stejné kategorie pro různé typy výrob a různý styl obchodování platí převážně pro MES.
Výrobní systém se ale překrývá se zmíněnými systémy. Dispečink může být prováděn jak z MES, tak z ERP nebo i systémy v řízení mohou obsahovat funkce pro sběr dat atp., ale žádný nenabízí stejnou funkcionalitu jako MES [5]. Navíc jsou také velké rozdíly v časových rámcích, MES pracuje i v časových horizontech sekund, zatímco pro systémy podnikové sféry (jako ERP) jsou časová období v řádu dnů až roků [7].

1.4 Funkce MES systémů

Pro vytvoření určitého rámce pro MES byl vytvořen seznam 11 funkcí, které by systémy měly obsahovat [1]. Tyto funkce jsou jednoznačně definovány v [5] (odkud jsou také převzaty). Model, který zobrazuje funkce MESu a jeho návaznosti na další systémy, se nazývá Funkční model (Obr.1.2).



Obr. 1.2: Funkční model výrobního systému MES [5]

Řízení a přidělování zdrojů - správa prostředků a zdrojů zahrnuje stroje, pracovní nástroje, materiál a další vybavení a entity jako dokumenty potřebné k zahájení pracovní operace. Poskytuje podrobnou historii těchto prostředků a zdrojů a zajišťuje jejich správné nastavení k použití a monitorování stavu v reálném čase (aktuální dostupnost). Tato funkce se také stará o jejich budoucí rezervace, kvalifikace (např. školení personálu [7]) a dispečink pro zajištění požadavků při plánování operací.

Operativní plánování výroby - vytváří frontu definující pořadí pracovních úkonů ([7]) na stroji, založenou na prioritách, recepturách výrobních jednotek a dalších charakteristikách. Pokud jsou operace správně zařazeny ve frontě, dochází k minimalizaci času seřizování strojů. Rozpozná překrývající se nebo paralelní operace, takže dokáže optimálně naplánovat čas pro změnu nastavení / výměnu strojů nebo přepis příslušných receptur (změna vzorů pro výrobu).

Dispečerské řízení - zajišťuje přidělování a sledování zdrojů a kapacit potřebných pro výrobní proces ([7]), konkrétně pro výrobní jednotku. Příkazy z dispečinku jsou dostupné v pořadí, v jakém musí být vykonány. Jakmile nastane událost ve výrobě, jsou změny promítány v reálném čase. Umožňuje měnit předepsaný plán operací ve výrobě. Má k dipozici procesy přepracování (*rework*) a postupy pro záchranu a schopnost řídit množství probíhající práce na kterémkoli stanovišti ve výrobě díky správě dat z bufferu (*buffer management*).

Správa dokumentace - řídí a kontroluje záznamy / formuláře, které musí být udržovány s výrobní jednotkou včetně pracovních pokynů, receptů, výkresů, standardních provozních postupů, dávkových záznamů. Kontroluje poznámky o technických změnách, komunikaci mezi jednotlivými směnami. Zajišťuje verzování dat [7]. Dovoluje editovat „pevné“ (*as built*) nebo naplánované (*as planned*) informace. Posílá pokyny k výrobním operacím včetně informací operátorům nebo recepty pro řízená zařízení. Zahrnuje také kontrolu a integritu ochrany životního prostředí, zdravotních a bezpečnostních předpisů a informací ISO. Zajišťuje ukládání historických dat.

Sběr a archivace dat - tato funkce poskytuje rozhraní související se ziskem provozních, výrobních a parametrických dat, které obsahují formuláře a záznamy, které byly připojeny k výrobní jednotce (resp. procesu). Jedná se vlastně o sběr procesních a výrobních dat, stavů zařízení apod [7]. Tato data mohou být shromažďována z výroby buď manuálně nebo automaticky v aktuálním časovém rámci.

Řízení práce (Řízení laboratoře) - poskytuje stav personálu v časovém rámci. Zahrnuje vykazování času a docházky, sledování osvědčení (kvalifikace). Má také

schopnost sledovat nepřímé aktivity, jako je příprava materiálu nebo práce s nástroji, jako základ pro kalkulaci podle činností. Při určování optimálních přiřazení může spolupracovat s Řízením a přidělováním zdrojů.

Řízení kvality - poskytuje analýzu naměřených dat z výroby v reálném čase, aby byla zajištěna správná kontrola kvality výrobků a aby byly identifikovány problémy vyžadující pozornost. Může doporučit opatření k nápravě problému včetně korelace příznaků, akcí a výsledků, k určení příčiny problému. Může zahrnovat sledování SPC/SQC (řízení statistik procesu a kvality) a správu off-line kontrolních operací. Může obsahovat i analýzu v systému pro řízení laboratorních informací (LIMS).

Řízení procesu - monitoruje výrobu a poskytuje rozhodovací podporu operátorům za účelem opravy a zlepšení činností v průběhu procesu (pokud chybí řídicí systém [7]) nebo provede opravy automaticky. Tento monitoring může být na provozním stanovišti (např. v rámci modulu zařízení - *intra-operational*) zaměřen konkrétně na stroje nebo zařízení, které jsou monitorovány a řízeny, stejně jako z nadřazené úrovně (například z procesní jednotky), což sleduje proces z jedné operace na druhou. Může se jednat o správu alarmů k zajištění, že si je operátor vědom změn v procesu, které jsou mimo přijatelné tolerance. Poskytuje rozhraní mezi inteligentními zařízeními a MES systémem prostřednictvím sběru dat (*Data Collection/Acquisition*).

Řízení údržby - sleduje a směřuje aktivity k údržbě zařízení a nástrojů pro zajištění jejich dostupnosti pro výrobu. Zajišťuje plánování pro pravidelné nebo preventivní údržby, stejně jako reakce (alarmy) na okamžité problémy. Udržuje historii událostí a problémů k zajištění podpory při diagnostice problémů.

Genealogie a trasování výroby - poskytuje vhled do všech pracovních míst výroby. Stavové informace mohou zahrnovat, kdo na něm pracuje, komponenty materiálů dodavatele, dávku, sériové číslo, aktuální výrobní podmínky a případné poplachy, přepracování nebo jiné výjimky týkající se výrobku. Funkce on-line sledování také vytváří historický záznam. Tento záznam umožňuje sledovatelnost komponent a použití každého koncového produktu (a zpětnou dohledatelnost jeho rodokmenu).

Analýza výkonnosti - poskytuje aktuální reporty z výrobních operací spolu s porovnáním s minulostí a očekávanými výsledky hospodaření. Mezi výkonnostní výsledky patří měřítko využití a dostupnosti zdrojů, doby cyklu jednotky, shody s výrobním plánem a výkonnosti podle norem. Může zahrnovat řízení statistik procesu (SPC). Vykresluje informace shromážděné z různých funkcí, které měří provozní

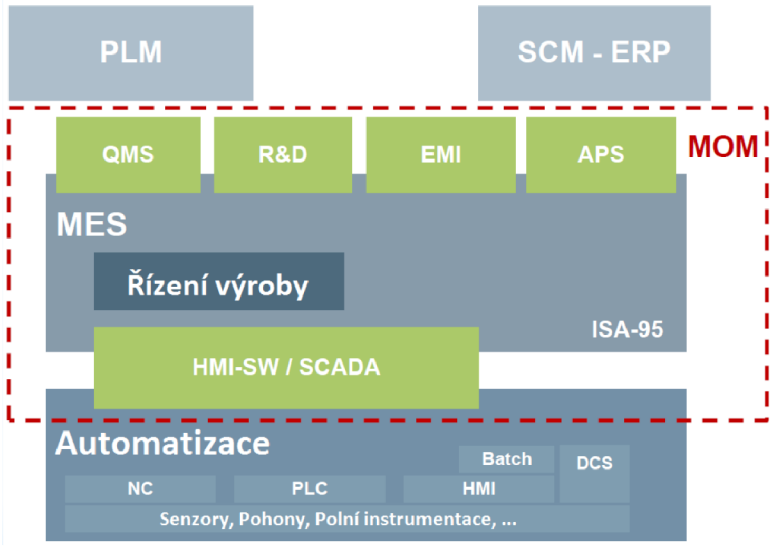
parametry. Tyto výsledky mohou být připraveny jako report nebo prezentovány online jako současné hodnocení výkonu.

1.4.1 Důsledky

Sumarizací funkcí MESu a s přihlédnutím k jeho komunikačním partnerům (Obr.1.2), se došlo k označení OT, tedy operační technologie. Dnes se ale vlivem neustálého pokroku (konvergence k IT) již tyto systémy řadí mezi ostatní informační systémy. Většina funkcí se navzájem obohacuje (prolínají se). Například sběr a archivace dat přispívá ke genealogii výrobku, řízení kvality přispívá k analýze výkonnosti stejně jako možné zasílání trendu dat do řízení údržby.
Funkce MES jsou pro většinu výrobců rozhodující kvůli obchodním tlakům, které vyžadují nové a přísnější procesy, které uspějí jak u zákazníků, tak u akcionářů. Např. správné rozdělení zdrojů může znamenat nejen rozdíl mezi „zdravým“ a minimálním ziskem, ale také včasnou produkci potřebnou k naplnění očekávání zákazníků. Analýza výkonnosti podniku je klíčovým faktorem nejen při rozhodování, co a kolik produkuje, v jakém zařízení, ale také v plánování kapitálu do budoucna [5].

1.5 Vznik systému MOM

V roce 2002 byl uveden tzv. *Collaborative* model neboli „Model MES systému vznikající spoluprací“, označovaný také jako c-MES. Hlavní myšlenka tohoto modelu je především rošíření původního MES o schopnost integrace dalších systémů, případně lidských zdrojů. Kromě ISA-88 a ISA-95 se jedná také o rozšíření původního systému o interoperabilitu business aplikací (OAG) a model dodavatelského řetězce (SCOR) [1]. Jednalo se vlastně o počátky MOMu.
Zkratka **MOM**(Manufacturing operations management - Řízení výrobních operací) vznikla v roce 2005. Nejedná se o konkrétní systém, ale o soubor aktivit nebo funkcí, které rozšiřují systém MES. Obsahuje navíc výrobní inteligenci (EMI), plánování (APS) a další moduly, jak je zobrazeno na Obr.1.3. MOM má hlavní podíl na konvergenci OT k IT, a tedy v dnešní době stále častější zařazení MESu k informačním systémům. Podle [7], se pojem MES dnes používá k popisu SW systémů, které pracují s vrstvou MOM. Všechny funkce jsou potom rozšířeny.
Jak zmínil člen MESA Dennis Brandl ve webcastu ([4]), původně měl vzniknout obohacením systému MES o nové funkce systém *Manufacturing Enterprise System*, nicméně protože akronym zůstal stejný, docházelo k nechtěným záměnám, zůstalo se proto u označení MOM.



Obr. 1.3: Rozdíl mezi MES a MOM a spolupráce s dalšími systémy [7].

1.6 Přínos výrobních systémů

MESA uvádí jako jeden z nejsilnějších aspektů MESu, že kombinací globálních obchodních pravidel a osvědčených postupů u lokálních operací dosahuje celopodnikový vhled nejen na to, co se děje, ale i na to, co by se mělo stát pro splnění obchodních a výrobních cílů. Na téma přínosů MESu vydala MESA koncem 90. let také zprávu (WP# 01), ve které jsou mimo jiné statistiky z výzkumu. Podle něj došlo vlivem MES k redukci času výrobního cyklu průměrně o 45%, redukci času zadávání dat o 75% a více, redukci dodací lhůty průměrně od 27% ([8] z r. 2005 uvádí 25%). Dále redukci „papírování“ mezi směnami o 61% a redukci výrobních defektů o 18%. Výzkum AMR [8] z roku 2005, kterého se účastnilo 20 společností, uvádí průměrnou návratnost (ROI) investice do MES za 12 měsíců.

Systém od *Critical Manufacturing* cmNavigo pro časově náročná průmyslová prostředí, byl testován na několika testovacích klientech. Ti byli naprogramováni pro vykonávání specifické série transakcí, měřil se počet provedených transakcí. Klienti měli přidělen různý počet aplikačních serverů (1 - 4). Z testů vyplynulo, že klient se čtyřmi aplikačními servery zvládl více než 3x více transakcí za sekundu [9]. I v dnešní době plné technologií a cloudu je tedy nutné správně nakonfigurovat infrastrukturu pro fungování klíčových systémů, jako je MES, pro vyhovění konkrétní aplikaci.

Návratnost spolu s dalšími parametry (jako jsou např. klíčové ukazatele výkonnosti KPI) ovlivňuje celkový přínos MESu. MESA vytvořila příručku [6], jak co nejlépe nastavit MES, za účelem zvýšení návratnosti založenou na nalezení otázek, jejichž odpovědi zajistí lepší integraci a zvýšení ROI systému.

1.7 Shrnutí a aktuální trend

Přestože jsou funkce MESu standardizované, mohou se překrývat, a to i napříč různými systémy. Doménou MESu je ale zaměření na vytěžení maxima z celopodnikové produkce a pokročilejší funkce pro optimalizaci provozu. S pomocí pečlivě vybraných dodavatelů SW a systémových integrátorů pak podniky mohou napojit všechny funkce MES na aktuální systémovou základnu podniku pro využití veškerých benefitů, které systém přináší.
Aktuálním trendem je přechod k Industry 4.0., a tedy integrace vertikálního a horizontálního propojení podnikové hierarchie (závod/podnik a závod/jiný závod). Detailní vize MES v Industry 4.0 je [44].

2 Plánování výrobních operací

Cílem této kapitoly je popsat vývoj a metody plánování výrobních operací s uvedením potřebných matematických nástrojů (případně příkladů) použitelných při plánování konkrétní úlohy v technické praxi.
Plánování výrobních operací je tématem propojující management s inženýringem. Je proto nutné popsat nezbytné součásti vývoje managementu plánování v návaznosti na výrobní systém MES.

2.1 Historie a vývoj plánovacích systémů

Vývoj *Manufacturing Planning and Control - Plánování a řízení výroby* (MPC) je v kontextu automatizace (tedy s ohledem na příchod počítačů) popisován již od poloviny 20. století. Celé období vývoje je charakterizováno několika fázemi, jejichž počet se napříč různými zdroji liší ([39] uvádí tři, [38] uvádí pět), nicméně významově se překrývají. V rámci každé fáze dominuje určitý systém, který obstarával plánování výroby a přidělování zdrojů.
Vývoj systémů pro plánování výrobních operací začíná začátkem 50. let. Používaly se metody využívané již dříve při manuálním plánování. Metoda měření a analýzy práce (MTM), která pracuje s předem určenými časy trvání cyklických úkonů. Používal se také systém Reorder point (ROP, ve volném překladu systém znovuobjednání). ROP pracuje s minimálním množstvím zboží (určeným z historických dat), které musí být na skladě, než dojde k opětovnému naskladnění (*reorder*). ROP funguje na úrovni deterministického modelu spojité poptávky nazvaném *Economic Order Quantity* (EOQ). Model EOQ stejně jako MTM se používají i dnes. Pro grafické zobrazení plánování výroby a potřebných zdrojů se používal Ganttův graf [38][40].
Začátkem 60. let přišel systém MRP. Jeho hlavním úkolem bylo naplnění tří základních požadavků:

1. zajištění dostupnosti všech potřebných materiálů pro výrobu a výrobků pro zákazníky
2. minimalizace skladových zásob
3. plánování výroby, dodacích a kupních rozvrhů

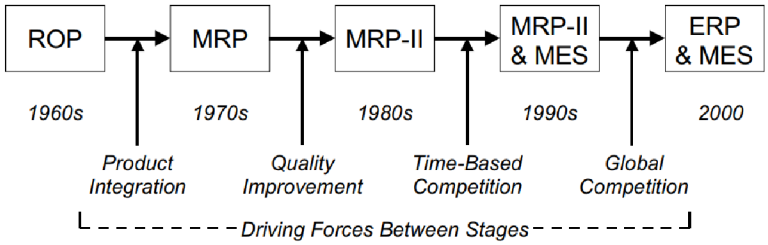
MRP začal postupně nahrazovat ROP systémy, protože nabízel efektivnější řešení, které zamezovalo špičkám (překročení limitů), které vznikaly u ROP systémů díky velkoobjemovému generování objednávek. Systém MRP navíc obsahoval nástroje pro reportování výrobních dat, které mohly v průběhu výroby porovnat hlavní výrobní plán s projektovanou poptávkou po materiálu. Počátkem 70. let bylo již několik stovek uživatelů MRP.

V polovině 70. let začaly systémy *Manufacturing Resource Planning - Plánování potřeb materiálu* (MRP2) nahrazovat systémy MRP jako primární systém pro řízení výroby. MRP2 rozšířil MRP o plánování požadovaných kapacit (CRP) a umožnil tak integrovat MPC systém sjednocující požadavky na materiál a výrobní kapacity do výpočtu celkových nároků na výrobu.
Zefektivnění operací přišlo začátkem 80. let s příchodem filozofie *Just-In-Time* (JIT). Jejím cílem bylo zlepšení základních charakteristik výrobního systému, jako je zlepšení kvality, redukce dílčích výrobních časů, důraz na požadavky zákazníka, minimalizace skladových zásob (již u MRP) atp. Metoda byla poprvé popsána u japonského výrobního systému Toyota (TPS). V dnešní době si mnoho firem přizpůsobuje tuto filozofii do vlastní podoby. V podstatě ve stejné době vznikla metodologie *Theory of constraints* (TOC), ta slouží k identifikaci nejpodstatnějších limitů (omezení), které brání k dosažení potřebného cíle. Postupným zapracováváním získaných informací dochází k redukci těchto limitů [41]. Na této úrovni jsou čtyři možnosti návrhu plánování a řízení výroby (MPC) [40]:

- Plánování kapacit založeno na objednávkách (*Make-to-order* MTO)
- Montáž založena na objednávkách (*Assemble-to-order* ATO)
- Plánování kapacit podle dlouhodobých předpovědí na odbyt výrobku(*Make-to-stock* MTS)
- Plánování podle potřeb zákazníků, bez předem daných konfigurací(*Engineer-to-order* ETO)

V pozdních 80. letech se plánování výroby začalo zaměřovat na schopnost dodavatelů vytvořit nebo přizpůsobit nové produkty a služby aktuálním požadavkům zákazníka. To znamenalo dynamickou výrobu, kde se produkty, procesy nebo výrobní plány mění i z hodiny na hodinu. MRP2 systém vyžaduje velké množství zásahů člověka při přizpůsobování plánu a určování optimální sekvence výrobních objednávek. Přes možnosti plánování kapacit (CRP) a řízení na úrovni výrobního závodu (*Shop Floor Control - Dílenské ovládání* (SFC)) nabízelo MRP2 málo informací k lepšímu způsobu vykonávání těchto činností.
To přispělo ke vzniku MES, který v tomto kontextu reprezentuje rozhraní mezi řízením závodu SFC a řídicími systémy zařízení (*Device Control Systems*). Největší přínos MESu je začlenění klíčových výrobních procesů do systému zaměřeného na požadavky zákazníka (systém přidaných hodnot - *value delivery system*). Poskytuje flexibilní, real-time výrobu a řízení širokého spektra výrobních procesů.
Koncem 90. let vlivem rostoucí globální konkurence v kombinaci se změnou trhu a tecnologií bylo mnoho firem domuceno přehodnotit a upravit stávající organizační strukturu a operační řízení pro produkty a služby. Aby firma byla úspěšná, bylo nutné flexibilní zavádění zdrojů a vytěžení maxima z dostupných informací (aktuální situace na trhu, z výroby) pro naplnění potřeb zákazníka. Jednalo se o řízení

dodavatelského řetězce SCM (*Supply Relationship Management*) zaměřené na zákazníka. Změny přístupu bylo nutné promítnout i do informačních systémů. Systémům MES pro tento účel chyběla dostatečná úroveň horizontální integrace (flexibilita napříč organizační strukturou), posun MPC proto zajistily až ERP systémy [38]. Fáze vývoje jsou na Obr.2.1.



Obr. 2.1: Vývojové fáze plánování a řízení výroby [38].

2.2 Plánování a rozvrhování ve výrobě

Na problematiku plánování operací ve výrobě (ale také jinde, například při plánování směn zdravotních sester apod.) lze pohlížet dvěma způsoby. První je způsob, kdy je dané pořadí operací a je požadavek na přiřazení těchto operací určitému času a prostoru (např. přiřazení zdrojů k operacím), pak se jedná o rozvrhování (v anglické literatuře *scheduling*. Ve druhém způsobu je požadavek na kauzální vztahy mezi operacemi a výběrem operací tak, aby bylo dosaženo cíle. Pak se jedná o plánování jako takové (anglicky *planning*)[42]. Přesto dochází k překrývání významů, a to nejen v české literatuře (např. [43]), kde je to opodstatněné anglickými překlady (*scheduling* lze přeložit také jako „plánování“, viz název kapitoly).

2.3 Rozvrhování výrobních operací

Rozvrhování v tomto kontextu znamená nalezení konkrétních časových úseků pro výrobní operace. Tato oblast patří do tzv. *back-office* procesů, které se nezabývají kontaktem se zákazníkem a plněním dlouhodobých cílů. Proto v porovnání s hlavním výrobním plánem (MPS) jsou časová období při rozvrhování výrobních operací kratší. Z definice uvedené v předchozím odstavci lze rozvrhování také chápat např. jako přiřazení zaměstnanců k pracovním úlohám nebo úlohy k pracovním stanovištím [10], tedy vazbu mezi zdroji (prostředky), službami a výrobními operacemi.
Jsou dva základní typy rozvrhování:

- Dopředné (*Forward scheduling*) - plánuje rozvrh úkolů od aktuálního okamžiku (úkol je zpracován, co nejdříve je to možné)
- Zpětné (*Backward scheduling*) - rozvrhuje úkoly od potřebného termínu zpracování zpět (úkol je zpracován v nejzazším možném termínu)

Přístup k rozvrhování se liší pro systémy s různým objemem výroby. Rozlišují se systémy velkoobjemové výroby, kam patří např. ropná rafinerie, do systémů středního objemu patří např. výroba automobilů, knih atp. Do nízkoobjemových systémů spadá mimo jiné výroba letadel [12]. Následuje série vlastností a parametrů charakterizujících rozvrh operací [11]:

Vytěžování (*loading*) se týká přiřazení úloh k pracovním centrům nebo stanovištím v těchto centrech, např. pokud lze jedna operace provést na více stanovištích nebo přiřazování výpočetních prostředků jednotlivým operacím. Vytěžování se dělí na:

- vytěžování s ohledem na kapacitu prostředků na stanovišti (stroji / pracovní stanici) (*finite loading*) - bere se v úvahu čas zpracování úlohy v jednotlivých pracovních centrech s ohledem na jejich kapacitu (času / materiálu / lidí)
- vytěžování bez ohledu na kapacitu prostředků (*infinite loading*) - týká se například přiřazení úkolů pracovním centrům bez ohledu na jejich (výpočetní) kapacitu (vznik front)

Vytěžování může být dále provedeno několika způsoby, přičemž použití závisí na konkrétní aplikaci:

- horizontálně - nejprve úlohou s nejvyšší prioritou, která si zabere všechna pracovní centra, která potřebuje, potom úloha s druhou nejvyšší prioritou atd.; jedná se o vytěžování s ohledem na kapacitu prostředků; nevýhodou je prodlení úloh, přestože požadované pracovní centrum je v nečinnosti, protože má přijít úloha s vyšší prioritou; horizontální vytěžování je spíše globální přístup k plánování rozvrhu
- vertikálně - pracovní centra jsou vytěžována úkoly postupně (jeden po druhém) bez ohledu na kapacitu prostředků; může dojít k přetížení pracovních center, což musí manažer řešit přesunem na jiná centra, zavedením přesčasů atp.; jedná se spíše o lokální přístup k plánování rozvrhu

Sekvencování (*sequencing*) řeší pořadí jednotlivých úkolů, výběr dle pravidel priorit (např. FC/FS - forma zásobníku FIFO, EDD - úkol s nejkratší dobou ukončení jde první,...). Některé zdroje řadí sekvencování jako další samostatnou kategorii k plánování a rozvrhování (např. [14] který definuje sekvencování jako řazení jednotlivých úkolů na úrovni zdrojů na stanoviště).

2.3.1 Rozvrhování v systémech velkoobjemové výroby

Tyto systémy jsou charakterizovány standardizovaným vybavením a činnostmi, které poskytují stejné nebo podobné operace nad produkty, které tímto vybavením prochází - jedná se především o spojitou výrobu. Všechny výrobní úkoly mají tedy stejnou sekvenci výrobních operací. Cílem je zajištění maximálního využití pracovních sil a vybavení. Mnoho zásadních rozhodnutí ovlivňujících výsledný rozvrh je určeno již při návrhu systému. Tyto systémy jsou označovány jako *flow systems*, tedy systémy toku a jejich rozvrh je označován jako *flow-shop scheduling*.
Návrh těchto systémů doprovází dva aspekty - vyvažování linky (tzv. *load balancing*) a návrh systémů toku (*flow systems design*). Vyvažování linky přiřazuje úlohy k pracovním centrům s ohledem na sekvenci operací a zajišťuje maximální využití zdrojů (zařízení a lidské síly). Návrh systémů toku je (z důvodu absence potřebné specializace personálu) zaměřen na rozdělení rozsáhlejších úkolů na dílčí úkoly, které personál vykonává. Úspěšnost těchto systémů je zaručena především jejich návrhem, preventivní údržbou, určení optimálních vstupů a výstupů při minimálních nákladech (prostředků ale i času). Změna rozvrhu výroby v těchto systémech se provádí méně často [11].

2.3.2 Rozvrhování v systémech výroby středního objemu

Tyto systémy se řadí mezi velkoobjemové systémy a systémy s MTO systémem (u nízkoobjemové výroby). Výroba zde má spíše charakter MTS (plánování kapacit z dlouhodobého hlediska) než ETO (plánování kapacit podle aktuálních potřeb). Řadí se sem převážně systémy s dávkovou výrobou (např. jídlo v konzervách, kosmetika atp). Plánování rozvrhu se týká časování úloh, sekvence úloh (v jakém pořadí mají být vykonávány) a velikosti dávky [11].

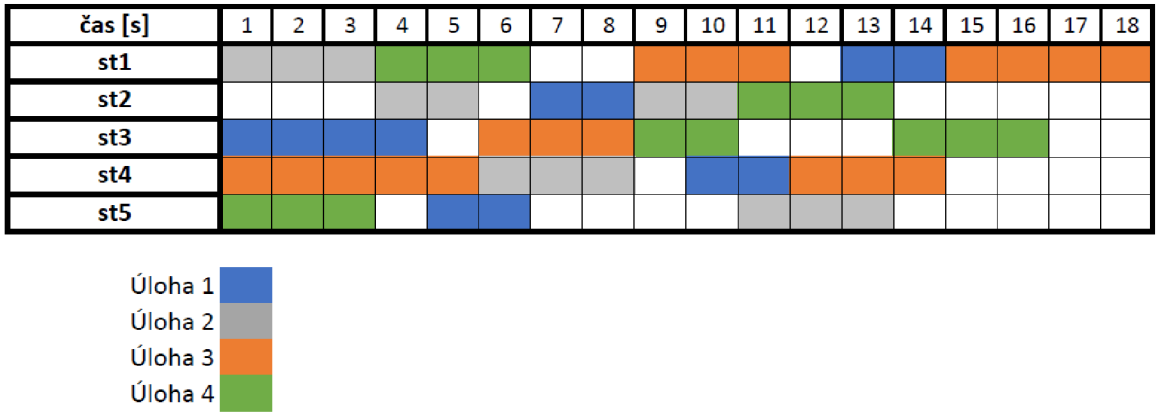
2.3.3 Rozvrhování v systémech nízkoobjemové výroby

Tyto systémy se označují jako systémy „pracoviště“ (*Job-Shop*) a jejich rozvrh se označuje jako *Job-Shop Scheduling* JSS. Plánování výroby má charakter MTO, přičemž objednávky se značně liší (častá změna rozvrhu [12]), a to v procesních požadavcích, času i sekvencích operací. Kvůli této komplexnosti je nemožné vytvoření rozvrhu úkolů před jejich objednávkou. Problematika tohoto typu výroby zahrnuje zatěžování i sekvencování. Určitou „nádstavbou“ na JSS je *Flexible Job-Shop Scheduling* FJSS, pro systémy, kde je více možných pracovišť pro provedení jedné operace [11].

2.4 Ganttův diagram

Jak již bylo zmíněno při popisu historie, jedná se o vizualizační prostředek vzniklý začátkem 20. století. Hlavním účelem je organizace aktuálního využití zdrojů (stanovišť / strojů / lidí / materiálu) v časovém rámci. Časová osa může být diskrétní (rozdělena na pevné úseky) nebo spojitá. Jsou dva hlavní typy Ganttova diagramu. Vytěžovací diagram a diagram rozvrhu [11].

Vytěžovací diagram znázorňuje časy nečinnosti pro skupinu stanovišť / strojů, případně pro pracovní centra. Slouží také ke znázornění kapacit prostředků (kapitola 2.3) [11]. Příklad Ganttova vytěžovacího diagramu s disktrétní časovou osou pro čtyři úkoly a pět stanovišť je na Obr.2.2 (bloky stejné barvy označují pracovní operace).



Obr. 2.2: Příklad Ganttova diagramu zpracovávání čtyř úkolů v rámci pěti pracovních stanovišť.

Diagram rozvrhu je strukturován podle typu rozvrhu, tedy jestli je dopředný nebo zpětný. U dopředného začíná pomyslná nula aktuálním okamžikem. U zpětného je v diagramu znázorněn nejzazší termín zpracování daného úkolu, mělo by být zřejmé, kdy nejpozději by měl být konkrétní úkol začít zpracovánán.

2.5 Plánování výrobních operací

Může se zdát, že plánování výrobních operací se překrývá se sekvencováním, které bylo popsáno u rozvrhování. Nicméně plánování v tomto kontextu znamená uspořádání pracovních operací do požadovaného sledu, který je daný konkrétním úkolem.

3 Možnosti optimalizace při rozvrhování výrobních operací

Slovo „optimální“ vychází z latinského *optimus*, což znamená „nejlepší“. Možnosti optimalizace se tedy zabývají variantami nalezení „nejlepšího“ řešení problému, kterým je v tomto případě rozvrhování výrobních operací. V rámci interpretace úloh rozvrhování výrobních operací lze konkrétní problém formulovat pomocí:

- Programování s omezujícími podmínkami
- Lineárního programování
- Smíšeného celočíselného programování
- Nelineárního programování

Možností formulace úlohy a nalezení „nejlepšího“ řešení je celá řada a poskytnout základní popis všech těchto metod a interpretací je nad rámec jakékoli publikace. V rámci této rešerše budou proto popsány pouze první tři úlohy programování a zároveň bude uveden úvod do nelineárního programování. Vztahy mezi prvními třemi typy popisu problému mohou být formou podmnožin, pokud jsou splněny určité podmínky. Metody řešení jsou popsány v podkapitolách, přičemž ty základní jsou popsány v rámci samostatné kapitoly.
Tyto souvislosti jsou objasněny v následujících kapitolách.

3.1 Programování s omezujícími podmínkami

Jde o paradigma programování vycházející z logického programování, pomocí kterého je problém formulován jako sada omezení (v). Ze všech zmíněných kategorií programování má tato po nelineárním programování nejširší rozsah aplikace (lze jí řešit jak lineární problémy, tak nelineární). Na rozdíl od ostatních nehledá optimální řešení, ale všechna řešení vyhovující podmínkám. To umožňuje některým systémům využít různá řešení v různých situacích. Programování s omezujícími podmínkami má deklarativní charakter, takže formulace problému není závislá na metodě použité pro řešení problému. Zároveň nelze pouze při pohledu na program určit, jak program funguje. Další nevýhodou je výpočetní a časová náročnost při hledání řešení. V cizojazyčné literatuře se lze v souvislosti s touto kategorií programování v matematice setkat se zkratkami CP (*Constrained Programming*) a CSP (*Constrained Satisfaction Problem*) [20]. Interpretace problému pomocí programování s omezujícími podmínkami je následující CSP [21]:
Je dána konečná množina proměnných $X = \{x_1, \ldots, x_n\}$, množina hodnot (doména) $D = D_1 \cup \cdots \cup D_k$ a množina omezení $C = \{c_1, \ldots, c_m\}$, přičemž každé omezení je definováno na podmnožině X.

3.1.1 Příklad definice problému

Zadané proměnné: A, B, C
Doména: $\{0, 1\}$ pro A, $\{1\}$ pro B, $\{0, 1, 2\}$ pro C
Omezení: $A \neq B$, $B \neq C$

3.1.2 Řešení

Pro řešení programování s omezujícími podmínkami se používá široká škála metod od metod pro lineárního programování přes Newtonovu metodu, metody Dynamického programování (rozdělující problém na dílčí podproblémy) a další. Řešení lze najít i využitím genetických algoritmů nebo neuronových sítí [20].

3.2 Lineární programování

Jedná se o optimalizační problém, ve kterém jsou funkce i veškerá omezení lineární. Přestože se tvar problémů lineárního programování může lišit úloha od úlohy, všechny je možné formulovat standardní formou [17]. Ta může být formulována [15]: Nalezněte minimum funkce

$$z = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n = \sum_{j=1}^{n} c_j x_j \tag{3.1}$$

za omezujících podmínek

$$\sum_{j=1}^{n} a_{ij} x_j \left\{ \begin{array}{c} < \\ \leq \\ = \\ \geq \\ > \end{array} \right\} b_i, \; pro\; i = 1, \ldots, m \tag{3.2}$$

$$x_j \geq 0, \; pro\; j = 1, \ldots, n \tag{3.3}$$

Pro pochopení formulace problému lineárního programování je potřeba popsat klíčové pojmy [16].

Rozhodovací proměnné jsou hodnoty, které musí být určeny pro vyřešení problému. Jedná se o neznámé matematického modelu. U této úlohy je jich n a jejich obvyklé označení je $x_1, \ldots, x_n$. Snahou je najít optimální hodnoty pro tyto proměnné.

Kriteriální funkce je rovnice 3.1 a určuje kvantitativní kritérium skupiny rozhodovacích proměnných $(x_1, \ldots, x_n)$, které v praxi mohou znamenat např. náklady, zisk, apod. Obecně může být kritérium minimalizováno nebo maximalizováno (mění se pouze znaménko koeficientu c_j) Někdy se také označuje jako účelová funkce.

Omezení jsou soubor m rovností a nerovností 3.2, definující hranice rozhodujících proměnných. Omezení jsou v praxi určena okolnostmi problému (parametry), například omezenými zdroji, smluvními závazky nebo fyzikálními zákony. b_i jsou hodnoty pravých stran (v zahraniční literatuře označované jako koeficienty *RHS - Right Hand Side*). V praxi se ctí pravidlo pojmenovávání omezení podle účelu.
Do této kategorie potom patří i jednoduchá horní omezení spojená s každou proměnnou x_j a omezující její hodnotu shora kvantitou u_j: $x_j \leq u_j$,. Pokud pro některou proměnnou x_j není hranice u_j uvedena, pak se proměnná nazývá shora neomezená.

Omezení na nezáporné hodnoty je provedeno rovnicí 3.3. Pro většinu praktických použití je požadavek na nezápornost proměnných $x_j \geq 0$, pro $j = 1, \ldots, n$ (tzv. triviální podmínky [15]). Někdy jsou ale proměnné neomezené, potom může být řešením libovolná reálná hodnota.

3.2.1 Příklad definice problému

Najděte maximum funkce $z = 5x_1 + 12x_2 + 4x_3$ s omezeními:
o_1: $x + 2x_2 + x_3 \leq 10$,
o_2: $2x_1 - x_2 + 3x_3 = 8$ a triviálními omezeními
o_3: $x_1, x_2, x_3 \geq 0$.

3.2.2 Řešení

Řešení problému lineárního programování může být nalezeno pomocí grafické metody, simplexové metody nebo duální metody. Patří sem také všechny metody nelineárního programování a metody popsané u programování s omezujícími podmínkami, přičemž jejich časová a výpočetní náročnost bude násobně větší.

3.3 Smíšené celočíselné programování MILP

V celočíselném programování se střetávají dva odlišné světy: spojitá a kombinatorická optimalizace. A to jak při formulaci problému, kde se vyskytují spojité i diskrétní proměnné a funkce, tak v metodách řešení, kde se kombinují techniky z

obou disciplín. Standardní tvar úlohy celočíselného programování je

$$min\ f(x) \tag{3.4}$$

za podmínek

$$g_j(x) \leq 0\ \ \forall j = 1, \ldots, m;\ x_i \in \mathbb{Z};\ \ \forall i \in C, \tag{3.5}$$

kde $C \subseteq \{1, \ldots, n\}$. Jedná se o úlohu programování s podmínkami na celočíselnost některých proměnných. Pokud je $C = \{1, \ldots, n\}$ jde o čistou úlohu celočíselného programování. Jestliže jsou funkce $f(x)$ a $g_j(x)$ lineární, pak jde o celočíselné lineární programování (v zahraniční literatuře označované jako MILP neboli *Mixed Integer Linear Programming*). Speciálním typem lineárního celočíselného programování je programování s binárními proměnnými 0-1, také označované jako bivalentní ([19]) [18].

3.3.1 Příklad definice problému MILP

Jsou dány dvě skupiny objektů, každá o 2 objektech. Každému objektu 1. skupiny je třeba přiřadit právě jeden objekt 2. skupiny. Přiřazení i-tého objektu 1. skupiny j-tému objektu 2. skupiny přinese zisk resp. ztrátu c_{ij}. Je-li i-tému objektu 1. skupiny přiřazen j-tý objekt 2. skupiny, je $x_{ij} = 1$, jinak $x_{ij} = 0$ [19].
$max\ 2x_{00} + 3x_{01} - 3x_{10} - x_{11}$
Omezení: $x_{00} + x_{01} = 1$
$x_{10} + x_{11} = 1$
$x_{00}, x_{01}, x_{10}, x_{11} \in \{0, 1\}$

3.3.2 Řešení

Řešení problému smíšeného celočíselného programování může být nalezeno vedle zmíněné simplexové a duální simplexové metody také pomocí tzv. Gomoryho metody. Jedná se o iterační metodu založenou na řešení úlohy lineárního programování. Další metodou je metoda Větvení a mezí (*branch & bound*), která opět řeší lineární program a pokud řešením nejsou celá čísla, pokračuje rozdělením prostoru daném omezeními na dva mnohostěny, v každém vyřeší úlohu znovu a vybere ten s lepším řešením a pokračuje, než najde vyhovující optimální řešení. [18].

3.4 Nelineární programování

Tato kategorie se také označuje pouze jako matematické programování a patří sem v podstatě všechny případy programování (včetně toho lineárního). Jeho úlohou se

rozumí:
$min\ f(x),\ x \in M,$
kde f: $\mathbb{R}^n \to \mathbb{R}$ je kriteriální funkce a $M \subseteq \mathbb{R}^n$. Množina M je navíc popsána soustavou:
$g_j(x) \leq 0,\ \ j = 1, \ldots, J,$
$h_l(x) = 0,\ \ l = 1, \ldots, L,$
kde $g_j(x), h_l(x) :\ \mathbb{R}^n \to \mathbb{R}$.
Tato obecná definice úlohy matematického programování pro hledání optimálního řešení x^* může být dále kategorizována jako hledání (ostrého nebo neostrého) globálního nebo lokálního minima. Úloha má několik tříd, z nichž již byla popsána třída lineárního programování. Dalšími třídami jsou [22]:

- Optimalizace bez omezení
- Konvexní programování - funkce f(x), $g_j(x)$ jsou konvexní (graf má jen jedno minimum) a $h_l(x)$ jsou lineární
- Ostatní - mezi tyto může patřit např. smíšené celočíselné nelineární programování (MINLP), kvadratické programování a další.

3.4.1 Řešení

Řešení nelineárního programování je značně složitější, což je vždy dáno tvarem funkce. Samotné způsoby řešení problémů matematického programování jsou různé gradientní metody (např. Newtonova), Lagrangeova duální metoda, penalizační bariérové metody (založené na aproximaci omezeného problému neomezeným [30]) a další, přičemž pro každou musí být obvykle splněny určité podmínky optimality (např. podmínky Fritze Johna nebo Karush-Kuhn-Tuckerovy podmínky). Pro některé speciální typy problémů nelineárního programování lze použít i simplexovou metodu [29]. Pro úlohy nelineárního programování jsou aplikovatelné také prostředky umělé inteligence (např. podpůrné vektorové stroje - SVM a další) [22].

3.5 Popis metod pro řešení

Tato kapitola popisuje metody pro nalezení optimálního řešení zmíněných matematických problémů.

3.5.1 Grafická metoda

Jde o nejnázornější metodu řešení. Umožňuje vykreslit omezení jako sadu přímek, které vymezují množinu přípustných řešení (*feasible solutions*). Optimální řešení

je potom zvoleno podle hodnoty objektivní (kriteriální) funkce. Nevýhodou této metody je použitelnost pouze v dvourozměrném prostoru [15]).

3.5.2 Simplexová metoda

Jak bylo zmíněno v předchozích kapitolách, tato metoda slouží k zisku algebraického řešení různých problémů programování v matematice. Myšlenka této metody je přecházení od jednoho přípustného řešení k dalšímu tak, aby byla neustále minimalizována kriteriální funkce [30]. Úloha musí být v tzv. kanonickém tvaru, aby mohla být řešena simplexovou metodou, tj. všechna omezení musí být dána pouze rovnicemi s nezápornou pravou stranou a všechny proměnné $x_j \geq 0$. Formulace pro simplexovou metodu: minimalizujte (maximalizujte)

$$z = \sum_{j=1}^{n} c_j x_j \tag{3.6}$$

za omezujících podmínek

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \; pro \; i = 1, 2, \ldots, m, \tag{3.7}$$

$$x_j \geq 0, \; pro \; j = 1, \ldots, n \tag{3.8}$$

Prakticky to znamená přidání nových proměnných do nerovnic pro získání rovnic a jejich vynásobení -1 pro zajištění nezáporného vektoru pravých stran. Problém v kanonickém tvaru se zapíše do tzv. simplexové tabulky, kde je cílem získat jednotkovou matici, tzn. výsledek. Pak se jen z tabulky zjistí, jestli je ještě možné minimalizovat (maximalizovat) objektivní funkci (podle jejích koeficientů), pokud ano, pokračuje se s upravenou tabulkou a přeměnou proměnných, jinak algoritmus končí, protože se dosáhlo optimálního řešení.
Nevýhodami simplexové metody jsou [15]:

- Degenerace - dochází k opakovanému procházení stejných vrcholů množiny přípustných řešení vlivem nadbytečných omezení, což lze vyloučit úpravou simplexové tabulky pomocí tzv. ϵ-modifikace
- Více optimálních řešení - kdy má hodnota kriteriální funkce stejnou hodnotu např. pro úsečku přípustných řešení
- Neomezené řešení - špatná formulace úlohy nebo nevhodně zvolený parametr; řešením je přidání dalšího omezení
- Množina přípustných řešení je prázdná - špatná formulace úlohy; omezení jsou v rozporu

3.5.3 Duální metoda

Původní úloha lineárního programování se označuje jako primární. V této kapitole bude popsána tzv. duální úloha, která je s tou primární navzájem rovnocenná. Duální úloha oproti té primární řeší minimalizaci funkce z (všechna omezení typu $\leq$), jestliže primární řeší maximalizaci a pokud je úkolem primární úlohy maximalizace, duální řeší minimalizaci (se všemi omezeními typu $\geq$). Vztah mezi koeficienty a proměnnými primární a duální metody je určen speciální tabulkou, která je pro ukázkový příklad 3.2.1 zobrazena v Tab.3.1. V prvním řádku jsou rozhodovací proměnné, ve druhém jsou koeficienty kriteriální funkce a v následujících řádcích jsou jednotlivá omezení. Dvojitá čára odděluje hodnoty pravých stran. Duální proměnné jsou na pravé straně tabulky a duální omezení jsou sloupce tabulky čtené zespod. Vzniklá duální úloha je potom:
Najděte minimum funkce $w = 10y_1 + 8y_2$ za podmínek:
o_1': $y_1 + 2y_2 \geq 5$,
o_2': $2y_1 - y_2 + 3x_3 \geq 12$
o_3': $y_1 + 3y_2 \geq 5$.
Pro shrnutí lze konstatovat, že při splnění určitých podmínek nezáleží, která úloha je primární a která je duální (každá metoda se blíží k optimálnímu řešení z „jiné“ strany). Když tedy známe „dobré“ řešení minimalizace a „dobré“ řešení maximalizace („dobré“ v tom smyslu, že se funkční hodnoty v obou případech od sebe moc neliší), lze s jistotou tvrdit, že skutečné optimum obou úloh má funkční hodnotu v intervalu určeném těmito dvěma „dobrými“ funkčními hodnotami [15].

Tab. 3.1: Vztahy mezi koeficienty primární a duální úlohy pro příklad

	x_1	x_2	x_3		
c_j	5	12	4	0	
o_1	1	2	1	10	y_1
o_2	2	-1	3	8	y_2

3.6 Výpočetní nástroje

V této kapitole budou popsány použité nástroje a programové prostředky pro řešení problémů interpretovaných metodami popsanými v předchozích kapitolách.

3.6.1 IBM ILOG CPLEX Optimizer

Tento komerční optimalizační nástroj (zkráceně jen CPLEX) pro optimalizaci se používá především v komerční sféře pro zvýšení výnosů, atp. Pro akademické použití je nástroj zdarma. Používá pružné a vysoce výkonné matematické solvery CP, LP, MILP a kvadratického programování s omezujícími podmínkami. Zároveň tyto solvery disponují paralelními algoritmy pro smíšené celočíselné lineární programování pro možnost využití více výpočetních zdrojů.
První verze CPLEX vznikla v roce 1988 pouze se solverem LP. Aktuální verze je 12.8., přičemž výpočetní čas úloh se vlivem vývoje výpočetní techniky mnohonásobně zrychlil. Metody pro řešení, které optimalizační nástroj využívá, jsou především simplexová metoda, metoda bariér a heuristické metody. Název CPLEX vlastně vychází z faktu, že je psaný v C a je založen na simplexové metodě (C-Simplex, odtud CPLEX) [26]. Paralelní algoritmy jsou potom použity na tzv. *load balancing* [24].

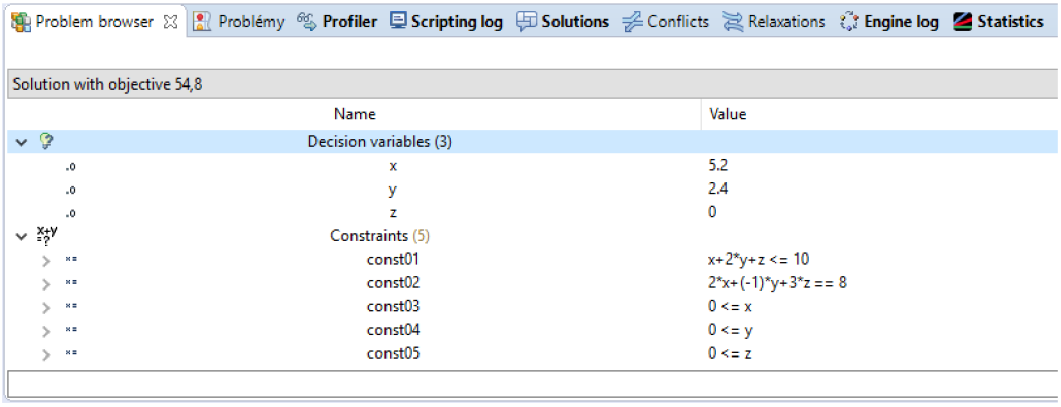
IBM ILOG CPLEX Optimization studio je vývojové prostředí s integrací optimalizátoru CPLEX. V tom se pro definici modelu používá tzv. OPL jazyk (*Optimization Programming Language*).[25]. Studio nabízí kromě „klasických“ vývojářských nástrojů pro ladění (které má ale daleko k intuitivnímu používání), pozastavení simulace, také záložky s pokročilými funkcemi (viz Obr.3.1):

- Problem browser - zobrazení výsledků rozhodovacích proměnných
- Profiler - detaily simulace jako čas běhu, paměťové a výpočetní nároky
- Solutions - popis vývoje a detaily řešení (hraniční hodnoty mezivýpočtů atp.)
- Engine log - další podrobnosti simulace (celk. počet proměnných a omezení) a grafické zobrazení počtu řešení
- Statistics - grafické zobrazení parametrů u složitějších modelů
- Další - DOcplexcloud (běh simulace v cloudu) apod.

Pro ukázkový příklad 3.2.1 lineárního programování je výsledek OPL modelu na Obr.3.1. Samotný OPL model zmíněného příkladu je:

```
dvar float x; // decision variables
dvar float y;
dvar float z;

maximize 5*x + 12*y + 4*z;

subject to
{
x + 2*y + z <= 10;
2*x - y + 3*z == 8;
x >= 0;
```

```
y >= 0;
z >= 0;
}
```



Obr. 3.1: Příklad použití vývojového prostředí IBM ILOG CPLEX Optimization studio.

3.6.2 LP Solve

Jedná se o solver smíšeného celočíselného lineárního programování, který je díky licenci GNU *General Public License* volně přístupný. Díky algoritmu *branch& bound* dokáže pracovat s celočíselnými, binárními nebo semi-kontinuálními proměnnými (proměnné, které musí mít hodnotu mezi minimem a maximem nebo nula). Umí pracovat s čistě lineárními, celočíselnými, binárními modely a modely speciálních uspořádaných množin SOS (*Special Ordered Sets*).
lp_ solve je vlastně knihovna, sada rutin, tzv. API rozhraní pro programování aplikací. Proto může být volán téměř z jakéhokoli programovacího jazyka pro řešení MILP. Navíc nabízí jednoduché IDE (integrované vývojové prostředí), kde lze modely vytvářet. Všechny možnosti, jak předat knihovně lp_ solve data jsou:

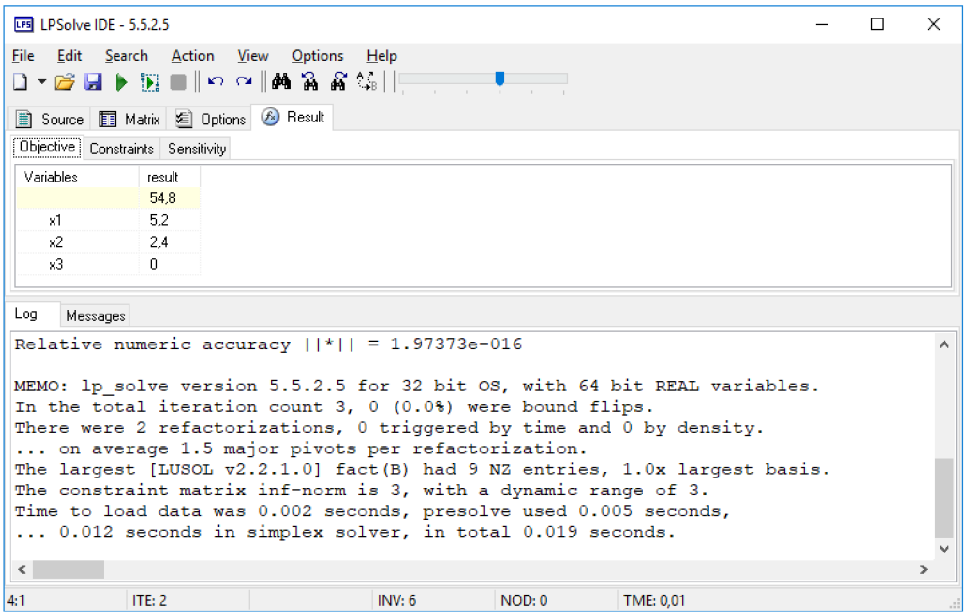
- přes API
- importem souboru se vstupními daty
- ve vývojovém prostředí (IDE)

Navíc lp_ solve není limitován velikostí modelu a pracuje se soubory s příponami `lp` a `mps`. Umožňuje práci i se soubory jiných typů, např. se soubory `mod`, které jsou importovány z programu *MathProg* [27].

LP Solve IDE je vývojové prostředí pro vytváření, editaci a spouštění souborů `.lp` pro zadání problémů lineárního programování. Přestože jeho možnosti jsou oproti CPLEX optimalizačnímu studiu značně omezené, umožňuje zobrazení výsledků v

matici, kontrolu syntaxe modelu a export do nejrůznějších formátů. Disponuje nastavením měřítka (lineární, geometrické atp.), pivotového pravidla (které je používané simplexovou metodou) a limitů simulace. V průběhu simulace zobrazuje výsledky a podrobnosti (např. počet iterací). Samozřejmostí je zobrazení použité metody (primární nebo duální úloha simplexové metody). Pro ukázkový příklad 3.2.1 lineárního programování je výsledek z `LPsolveIDE` na Obr.3.2, přičemž zadání vypadá následujícím způsobem:

```
/* Objective function */
max: 5 x1 +12 x2 +4 x3;

/* Variable bounds */
c1: x +2 x2 + x3 <= 10;
c2: 2 x1 - x2 +3 x3 = 8;
c3: x1 >= 0;
c4: x2 >= 0;
c5: x3 >= 0;
```



Obr. 3.2: Příklad použití vývojového prostředí LPSolveIDE.

3.6.3 Další výpočetní nástroje

Kromě zmíněných nástrojů, které jsou v rámci této práce použity, je k dispozici řada dalších. Pro doplnění přehledu je uvedeno v následujícím odstavci několik dalších nástrojů.

Microsoft Solver Foundation (MSF) je sada vývojových nástrojů pro matematickou simulaci, optimalizaci a modelování, které mohou být použity z kteréhokoli jazyka používajícího .NET framework nebo přes Microsoft Office jako doplněk pro Excel. Dá se použít pro lineární, nelineární / kvazi-Newtonské / kvadratické / kónické programování (forma konvexního programování) a programování s omezujícími podmínkami [28]. Hlavní nevýhodou a důvodem, proč není MSF využit pro simulace v rámci této práce, jsou omezení dána volnou licencí (500 proměnných a pár tisíc omezení). Další nevýhodou je neaktivní komunita a podpora. Přestože v minulosti byla k dispozici akademická licence bez omezení, v současnosti již tato možnost není.

Z dalších optimalizačních nástrojů lze uvést

- Gurobi a OPTANO solver - lze použít v kombinaci s MSF
- Mosek - určen hlavně pro optimalizaci rozsáhlých problémů

4 Praktický optimalizační problém Barman

Zde bude uveden projekt Barman a problém plánování výrobních operací v rámci tohoto projektu. Dále budou aplikovány matematické metody pro vyřešení tohoto problému.

4.1 Představení projektu

Jedná se o projekt autonomního barmana, který v současné době vzniká na Ústavu automatizace a měřicí techniky. Možnostmi, které projekt nabízí, jsou návrh a provoz systému pro řízení výroby (MES) a systému pro plánování podnikových zdrojů. Dále je projekt cílen na integraci, demonstraci a praktické ověřování principů Průmyslu 4.0 (I4.0), jelikož na výsledný nápoj lze pohlížet jako na produkt dávkové výroby (nápoj) nebo jako na produkt diskrétní výroby (sklenice). Jedním z neopomenutelných cílů projektu je široká škála praktických projektů pro studenty UAMT (Ústavu automatizace a měřicí techniky). Podobný projekt byl na podzim loňského roku realizován na půdě CIIRC ČVUT v Praze [32].
Prakticky se jedná o stůl na míchání nápojů. Aktuální mechanická konstrukce stolu, testbed je na Obr.4.1. Skládá se z 5 autonomních buněk, dopravníkového pásu pro hotové výrobky a robotického SCARA manipulátoru pro přesun výrobku mezi jednotlivými výrobními stanovišti. Celkový počet všech stanovišť je tedy 7:

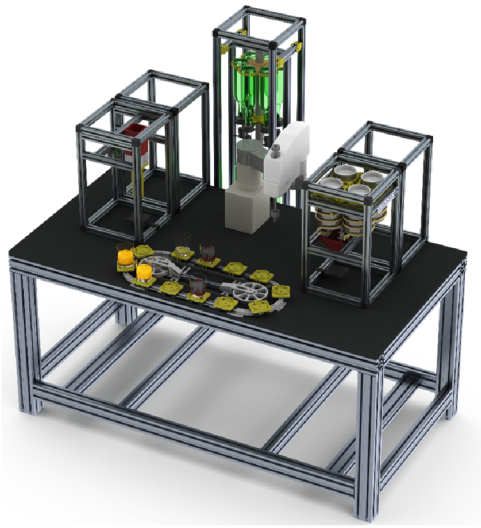
1. zásobník alkoholu (*)
2. zásobník nealkoholických nápojů (*)
3. buňka drtiče ledu (*)
4. buňka míchače nápojů (shaker) (*)
5. buňka zásobníku sklenic (*)
6. dopravníkový pás
7. robotický manipulátor

Autonomní buňky jsou označeny (*) [31].

4.1.1 Proces výroby nápoje (z pohledu zákazníka)

Zadáním objednávky zákazníkem (a její potvrzení) prostřednictvím online webové aplikace dojde k jejímu předání do ERP systému. Následně je vytvořena zakázka a předána MES systému. Systém pro řízení výroby pak zakázku přebírá a provádí plánování výroby (tedy řazení objednávky do výrobní fronty). Pozice ve výrobní frontě může být dynamicky modifikována, a to až do okamžiku fyzického zahájení výroby konkrétního kusu na základě:

- doby čekání ve frontě



Obr. 4.1: Současný návrh mechanické části testbedu [31]

.

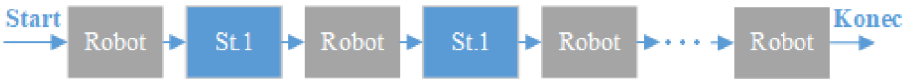
- možnosti agregace výroby (výroba podobných produktů v těsném sledu)
- možnosti výrobní linky

Co se týká samotného procesu výroby, každá sklenice obsahuje RFID čip s recepturou pro konkrétní nápoj (ten slouží také pro stanovení času dokončení výroby nápoje). Sklenice bude přesouvána pomocí robotu. V první fázi tedy robot vyzvedne čistou sklenici ze zásobníku sklenic. Tu umístí do zásobníku alkoholických nápojů. Zásobník zajistí napuštění požadovaného množství alkoholu. Sklenice bude následně přemístěna do další z buněk, kde se na základě požadavků receptury zajistí požadovaná akce. Takto sklenice postupně projde všemi potřebnými buňkami, aby nakonec byla robotickým manipulátorem posazena na jeden z vozíků dopravníkového pásu, na kterém bude distribuována zákazníkovi. Dopravníkový pás slouží rovněž pro odkládání prázdných sklenic, které jsou robotem umisťovány do druhého zásobníku [31].

4.2 Návrh modelu

Jak plyne z kapitoly 4.1.1. Každá objednávka se skládá z posloupnosti operací vykonávaných na různých stanovištích (autonomní buňky a dopravníkový pás). Nelze opomenout roli robota, který díky přesunu sklenice z určitého stanoviště na jiné zahrnuje každou lichou operaci zmíněné posloupnosti. Ilustrace sady takovýchto instrukcí je na Obr. 4.2. Nyní lze identifikovat známé parametry a neznámé proměnné pro optimalizační úlohu. Důležitými **známými** parametry jsou:

- Doba přechodů mezi každými dvěma stanovišti



Obr. 4.2: Diagram přípravy nápoje

.

- Nutná doba setrvání na každém stanovišti pro každý výrobek
- Priorita nápoje (určitelná pořadím objednávek, dobou čekání ve frontě)
- Aktuální stavy přípravy všech připravovaných drinků
- Aktuální volné pozice na dopravním pásu
- Pořadí procedur dané recepturou v RFID identifikátoru sklenice

Hlavní neznámou z pohledu zákazníka je čas dokončení výroby nápoje. Pokud jde o určení neznámých při plánování výrobních operací projektu Barman, jsou neznámými problému počáteční časy výrobku na jednotlivých stanovištích. Odtud lze potom pomocí známých parametrů dopočítat čas, kdy bude hotový nápoj připraven na dopravníkovém pásu.

4.2.1 Rozvrh práce problému Barman

Z kapitoly 2.3.3 je možno identifikovat problém jako rozvrh práce nebo také rozvrh úloh (*job-shop scheduling*). Každá úloha se může skládat z několika úkolů, které jsou prováděny na různých stanovištích (na každém je provedena nějaká operace).
Nejlépe si lze JSS představit, jako model kanceláře, kde pracuje např. 5 lidí (5 úloh). V rámci kanceláře jsou pracovní místa (stanoviště), např. kopírka, 3 počítače s různými programy, skartovačka a další. Každý z těchto lidí má sadu úkolů, které musí vykonat na určitých pracovištích, nesmí ale kolidovat s jiným člověkem. Rozvrh práce je optimalizační model, který hledá optimální rozvrh, aby byly dokončeny všechny úkoly těchto pěti lidí v co nejkratším čase.
Kriteriální funkcí je minimalizace času dokončení poslední operace (označovaný jako *makespan* [33]). Rozvrh práce má jistá omezení [34]:

- Žádný úkol nesmí začít dokud není dokončen ten předchozí v rámci jedné úlohy (operace se nesmí překrývat)
- Na každém stanovišti může v jednu chvíli probíhat pouze jeden úkol (jedna operace)
- Jakmile je úloha započata, musí proběhnout až do skončení její poslední operace

Předpoklady JSS problému jsou [36]:

- Každý úkol může být vykonáván pouze na jednom stanovišti

- Existuje pouze jedna sekvence operací pro každou úlohu

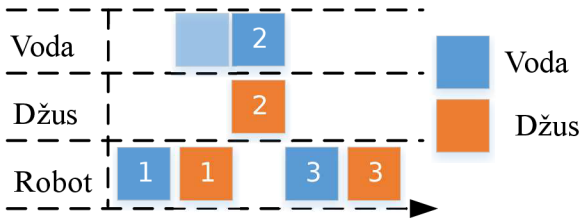
Existuje mnoho variací *job-shop schedulingu*, přičemž většinou zůstávají zmíněná omezení. Jedním z rozšíření původního JSS je práce s prioritami (každá úloha má svoji prioritu).
Nejbližším z hlediska instancí JSS, je projektu barman formulace *Body shop* , což je model karosárny. Problémem je, že ohledně této problematiky je [37] jediný zdroj. Hlavním důvodem, proč ale nebyl použit tento model jsou jistá omezení (např. všechny úlohy musí být vykonávány na všech stanovištích).

4.3 Analýza a popis problému

Základní problém byl specifikován výše, s tím, že jednotlivé úkoly představují operace na jednotlivých stanovištích. Stanoviště jsou všechny autonomní buňky, dopravníkový pás a „hlavním“ stanovištěm je robotický manipulátor. Vzhledem k tomu, že variant *job-shop schedulingu* (tedy rozšíření klasického pojetí popsaného v kapitole 4.2.1) je celá řada, je nutné správně definovat problém a následně přizpůsobit rozvrh práce. Proto musí být vybrán vhodný zdroj zabývající se touto problematikou nebo musí být odstraněna některá omezení z jiných dostupných zdrojů, která jsou součástí popisu problému rozvrhu práce JSSP a tvorby modelu. Jako příklad takovýchto omezení lze uvést:

× Každá úloha může být na jednom stanovišti provedena pouze jednou - je v rozporu se stanovištěm robota [35]

× Počet stanovišť je stejný jako počet operací každé úlohy - není pravidlem [33]



Obr. 4.3: Zobrazení příkladu rozvrhu JSS, který je nerealizovatelný u problému Barman

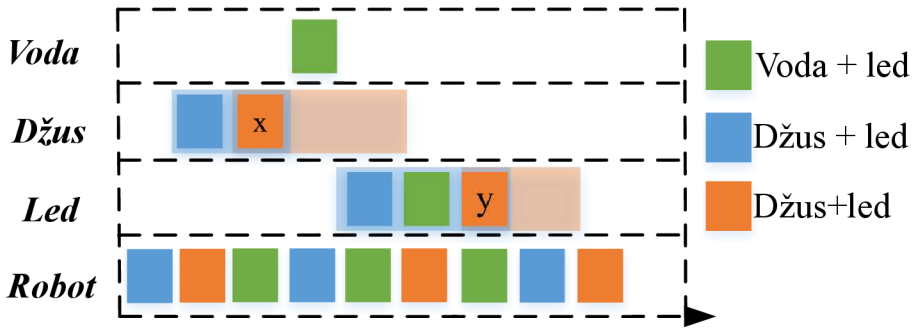
.

Přesto je nutné zavést další pravidla (které základní úloha JSSP neřeší), aby se optimalizační úloha projektu Barman stala realizovatelnou:

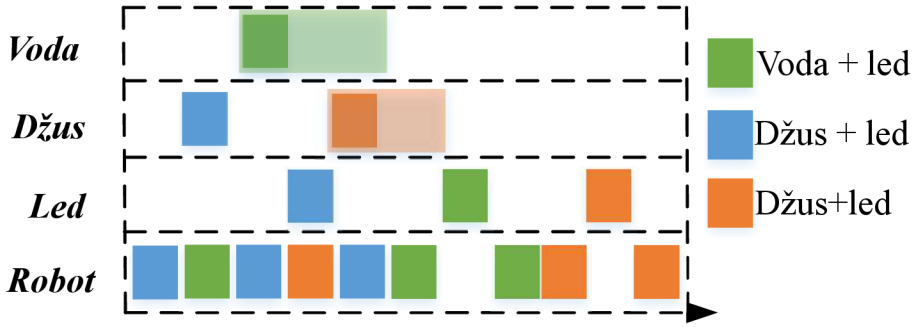
1. Musí se zajistit, aby operace robota byla vždy (mimo poslední operaci robota[1]) následována operací v rámci výroby stejného nápoje (naznačeno světlým odstínem barvy operace na Obr.4.3).

[1] viz kapitola 4.3.1

2. Plánování s ohledem na aktuální pozice sklenic. Přestože z hlediska rozvrhu je operace již dokončena, sklenice je pořád na daném stanovišti, dokud ji robotický manipulátor nepřesune na jiné stanoviště. V originálním problému rozvrhu práce JSSP tak může docházet ke hromadění (kolizi) sklenic na jednom stanovišti (viz operace `x` a `y` na Obr.4.4)



(a) Výrobní rozvrh 3 nápojů s kolizí sklenic na jednom stanovišti)



(b) Optimalizovaný výrobní rozvrh bez kolize sklenic.

Obr. 4.4: Příklad komplikace základní úlohy *job-shop schedulingu.*

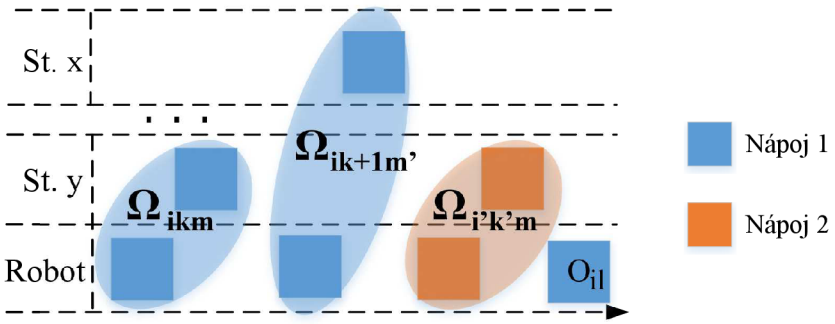
4.3.1 Zjednodušení

Přestože je problém detailně specifikován a parametrizován, je potřeba zavést jistá zjednodušení, která mají i praktická úskalí. Základním zjednodušením před formulací modelu je zanedbání poslední operace výroby každého nápoje (pohyb po dopravníkovém pásu) a tedy přesný odhad času dodání nápoje. Množina operací pro výrobu nápoje bude mít vždy lichý počet prvků (každá lichá operace je přesun sklenice robotem). To má za následek nejen redukci počtu operací každého nápoje, ale i snížení celkového počtu stanovišť na 6.
Dalším předpokladem řešení je konstantní počet objednaných nápojů během výpočtu řešení (nedochází k objednávkám během aktuální výroby). Pokud dochází k častým objednávkám, je nejprve nutné např. zpracovat prvních 10, provést optimalizaci atd. Vznikající fronta určí prioritu nápojů.

Omezujícím faktorem je také nemožnost zastavit probíhající výrobu nápojů za účelem přidání dalšího nápoje (nová optimalizační úloha).

4.3.2 Formulace

Je k dispozici množina nezávislých úloh J, kde každá úloha má prioritu p_j. O_i je množina operací odpovídající úloze i. Popis všech parametrů je v Tab.4.1. Aby bylo zabráněno kolizím sklenic a zároveň pro zajištění realizovatelnosti optimalizačního algoritmu, je nutné přinést rozšíření základní úlohy JSS. Jednou z možností je zavedení složených operací Ω, které slučují operaci na každém stanovišti s předchozí operací robota. Jejich význam ilustruje Obr.4.5. Praktický význam z pohledu zákazníka, který složené operace přináší je rozdělení výroby nápoje na výrobní činnosti, v nichž je zahrnutý i přesun sklenice robotem. Například zákazník vidí, že stav výroby nápoje ve webové aplikaci je míchání, přičemž prakticky teprve robot přesouvá sklenici ke stanovišti *shakeru*. Parametr `m` označující stanoviště složené operace tak vždy přísluší druhé operaci Ω. Důsledkem vzniku složených operací je osamostatnění poslední operace robota pro přesun sklenice na výdejní dopravníkový pás.

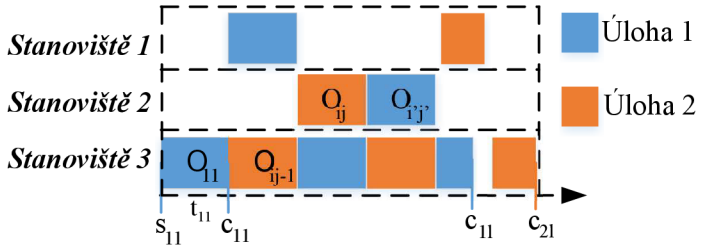


Obr. 4.5: Znázornění složených operací pro zajištění realizovatelnosti optimalizační úlohy Barman.

Kriteriální funkce určuje cíl optimalizace, kterým je v tomto případě minimalizace sumy časů dokončení každé úlohy (nápoje) vynásobených s prioritou. Jako kriteriální funkce může také sloužit minimalizace sumy časů dokončení / časů startu všech operací v rámci jedné úlohy / výroby jednoho nápoje vynásobených prioritou nápoje.

$$minimize \quad \sum_{i=0}^{n_j} p_i \cdot c_{il} \tag{4.1}$$

$$minimize \quad \sum_{i=0}^{n_j} \sum_{j=0}^{n_{oi}} p_i \cdot c_{ij} \tag{4.2}$$



Obr. 4.6: Zobrazení popisu parametrů na obecném příkladu

Omezení zajišťují realizovatelnost řešení. První podmínka (4.3) zamezuje překrývání operací v rámci jednoho stanoviště (např. nejprve se musí dokončit míchání jednoho nápoje O_{ij}, než se započne míchání dalšího $O_{i'j'}$, viz Obr.4.6). Omezení zajišťující realizovatelnost je (4.4). Řeší problematiku kolizí, jak je naznačeno na Obr.4.4. Označení jsou stejná jako na Obr.4.5. Jestliže složená operace $\Omega_{i'k'm}$ s počátečním časem $s_{i'k'm}$ a časem dokončení $c_{i'k'm}$ následuje operaci Ω_{ikm} s počátečním časem s_{ikm} a časem dokončení c_{ikm}, pak musí být nejdříve provedena složená operace $\Omega_{ik+1m'}$ (následující operaci Ω_{ikm} v rámci stejné úlohy/výroby stejného nápoje). Na pravé straně sjednocení omezení (4.4) jsou podmínky zajišťující, že stejná pravidla platí i obráceně (tedy pokud složená operace Ω_{ikm} následuje složenou operaci $\Omega_{i'k'm}$). Omezení (4.5) a (4.6) umožňují vykonávání určité operace až po tom, co je dokončena předchozí operace (O_{ij} a O_{ij-1}). Pokud se jedná o operace náležící do stejné složené operace, pak musí navazovat přímo (4.5), jinak stačí znaménko nerovnosti (4.6). Omezení (4.7) je podmínka pro konkrétní operaci, jejíž čas dokončení se skládá z času začátku operace a doby jejího trvání (na Obr.4.6 je to operace O_{11}). Poslední podmínka (4.8) určuje nezápornost všech proměnných a parametrů.

$$s_{i'j'm} - c_{ijm} \geq 0 \ \cup \ s_{ijm} - c_{i'j'm} \geq 0 \quad \forall i \in J - \{i'\}, \ \forall j \in O_i - \{j'\}, \ m \in M \tag{4.3}$$

$$s_{i'k'm} - c_{ikm} \geq 0 \ \cap \ s_{i'k'm} - c_{ik+1m'} \geq 0 \ \cup \ s_{ikm} - c_{i'k'm} \geq 0 \ \cap \ s_{ikm} - c_{i'k'+1m''} \geq 0$$
$$\forall i \in J - \{i'\}, \ \forall k \in \mathbb{O}_i, \forall k' \in \mathbb{O}_{i'} \ m, m', m'' \in M, m \neq m', m \neq m'' \tag{4.4}$$

$$s_{ij} - s_{ij-1} = t_{ij-1} \ \forall i \in J, \ \forall j \in O_i : \ O_{ij}, O_{ij-1} \in \mathbb{O}_{ik} \tag{4.5}$$

$$s_{ij} - s_{ij-1} \geq t_{ij-1} \ \forall i \in J, \ \forall j \in O_i : \ O_{ij} \notin \mathbb{O}_{ik} \tag{4.6}$$

$$c_{ij} \geq s_{ij} + t_{ij} \ \forall i \in J, \ \forall j \in O_i \tag{4.7}$$

$$c_{ij}, s_{ij}, t_{ij} \geq 0 \ \forall i \in J, \ \forall j \in O_i, \tag{4.8}$$

Uvedený matematický popis by měl být dostačující pro formulaci programu s omezujícími podmínkami.

Tab. 4.1: Popis symbolů používaných při definici problému v textu.

Ozn.	***Obecný popis***	***konkrétní popis***
J	množina nezávislých úloh	objednané nápoje
i,i'	index úlohy	index nápoje
m,m',m''	čísla stanovišť	čísla stanovišť (shaker,..)
j/k	index operace/složené operace	index aktuální operace/činnosti (např. míchání)
p_i	priorita úlohy i	priorita nápoje určená pořadím ve frontě
$O_i/\mathbb{O}_i$	množina operací/složených operací odpovídající úloze i	dílčí operace/činnosti výroby nápoje (led, přesun robotem...)
$\mathbb{O}_{ik}$	k-tá složená operace úlohy i	k-tá činnost výroby nápoje
O_{il}	poslední operace úlohy i	poslední operace výroby i-tého nápoje (přesun nápoje na výdejní pás)
Ω_{ikm_y}	k-tá složená operace i-té úlohy mající druhou operaci na stanovišti m_y	k-tá činnost výroby i-tého nápoje, jejíž druhá operace je na stanovišti m_y
$\Omega_{i'km_y}$	k-tá složená operace jiné než i-té úlohy mající druhou operaci na stanovišti y	k-tá činnost výroby jiného než i-tého nápoje, jejíž druhá operace je na stanovišti y
Ω_{ik+1m_x}	Složená operace následující Ω_{ikm_y} stejné úlohy mající druhou operaci na stanovišti x	činnost výroby i-tého nápoje následující činnost k, jejíž druhá operace je na stanovišti x
O'_i	množina operací odpovídající úloze i'	dílčí operace výroby nápoje (přidání vody, led,...)
M	množina stanovišť	autonomní buňky, robot, dopravník
s_{ij}/s_{ik}	začátek j-tého časového slotu/k-té složené operace úlohy i	začátek j-té operace/k-té činnosti výroby i-tého nápoje
c_{ij}/c_{ik}	konec j-tého časového slotu/k-té složené operace úlohy i	konec j-té operace/k-té činnosti výroby i-tého nápoje
c_{il}	konec posledního časového slotu úlohy i	konec poslední operace výroby i-tého nápoje
t_{ij}	je délka j-tého časového slotu úlohy i	délka j-té operace výroby i-tého nápoje
n_j/n_{oi}	je počet úloh/počet operací úlohy i	poč. objednaných nápojů / poč. oper. výroby nápoje i

4.3.3 Formulace problému z hlediska lineárního programování

Jak plyne z kapitoly 4.3.3, musí být jak kriteriální funkce, tak všechna omezení lineární. Při bližším pohledu na matematickou formulaci omezení (4.3) a (4.4) , je zřejmé, že musí být modifikována do lineární podoby. Většinou se tato omezení řeší přidáním binární proměnné a dostatečně velké konstanty (jak uvádí například [36]). Pro (4.3) platí:

$$s_{ij} \geq c_{i'j'} - Y_{iji'j'} \cdot L \quad \forall i < i',\ \forall j \in O_i,\ \forall j' \in O'_i \tag{4.9}$$

$$s_{i'j'} \geq c_{ij} - (1 - Y_{iji'j'}) \cdot L \quad \forall i < i',\ \forall j \in O_i,\ \forall j' \in O'_i \tag{4.10}$$

kde $Y_{iji'j'}$ je binární proměnná rovna 1, pokud operace O_{ij} předbíhá operaci $O_{i'j'}$, jinak je rovna 0. L je dostatečně velká konstanta. Jestliže tedy operace O_{ij} je před operací $O_{i'j'}$, potom je na začátek časového slotu s_{ij} kladena pouze podmínka nezápornosti, zatímco $s_{i'j'} \geq c_{ij}$ a naopak.
Pro omezení (4.4) zajišťující realizovatelnost jsou lineární nerovnosti následující:

$$s_{i'k'm} - c_{ikm} \geq (X_{iki'k'} - 1) \cdot L \tag{4.11}$$

$$s_{i'k'm} - c_{ik+1m'} \geq (X_{iki'k'} - 1) \cdot L \tag{4.12}$$

$$c_{i'k'm} - s_{ikm} \leq X_{iki'k'} \cdot L \tag{4.13}$$

$$c_{i'k'+1m''} - s_{ikm} \leq X_{iki'k'} \cdot L \tag{4.14}$$

$$\forall i \in J - \{i'\},\ \forall k \in \mathbb{O}_i, \forall k' \in \mathbb{O}_{i'}\ m, m', m'' \in M, m \neq m', m \neq m''$$

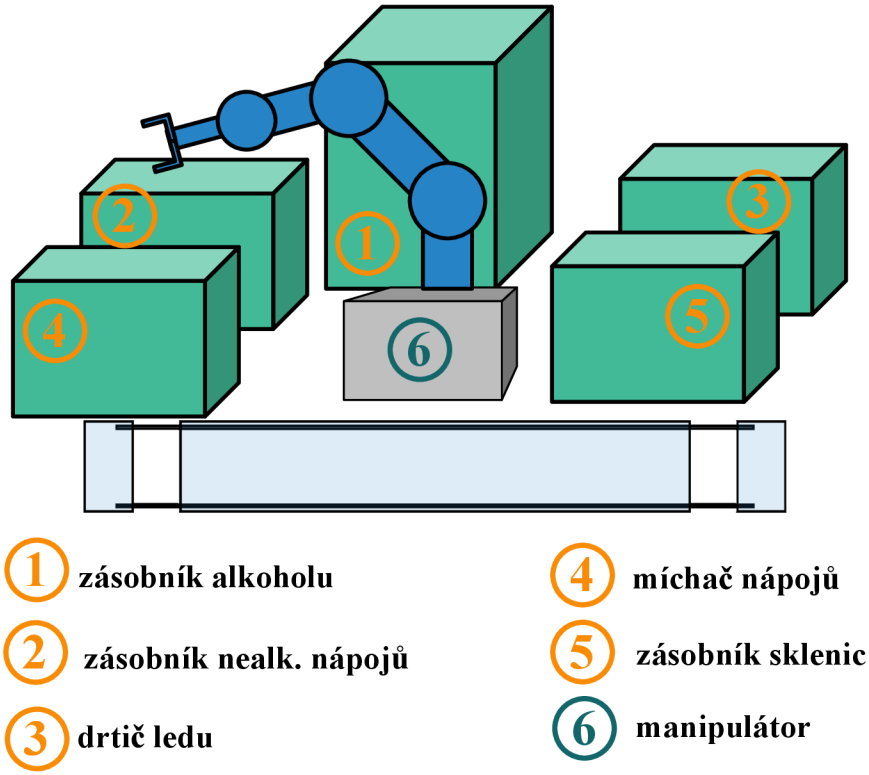
kde $X_{iki'k'}$ je binární proměnná rovna 1, pokud složená operace $\mathbb{O}_{ik}$ předbíhá složenou operaci $\mathbb{O}_{i'k'}$ na stanovišti `m`, jinak je rovna 0. L je opět dostatečně velká konstanta. Podmínky (4.11) a (4.12) musí být splněné na úkor podmínek (4.13) a (4.14). Z principu zavedení binárních proměnných je nutné, aby konstanta `L` byla násobně větší než výsledek levé strany každé nerovnice.
Je zřejmé, že tato formulace bude platit i pro smíšené celočíselné lineární programování s tím, že požadavek na celá čísla je zadán při formulaci problému pro výpočetní solver.

4.4 Demonstrační příklad

Než bude uveden příklad plánování operací pro projekt Barman, je nejprve nutné zavést společná označení stanovišť. Vezme-li se jako vzor aktuální konstrukce testbedu z Obr.4.1, lze stanoviště označit podle ilustračního Obr.4.7. Demonstrační příklad se skládá z 5 objednaných nápojů, přičemž každý lze rozložit do sady operací, které jsou zobrazeny na Obr.4.8. Priorita jednotlivých nápojů je opačná vůči jejich pořadí (nápoj 1 má prioritu 5, nápoj 5 má prioritu 1 atp.). Jak bylo popsáno dříve, každou

lichou operaci tvoří samotný manipulátor, který provádí přesun sklenice mezi stanovišti.

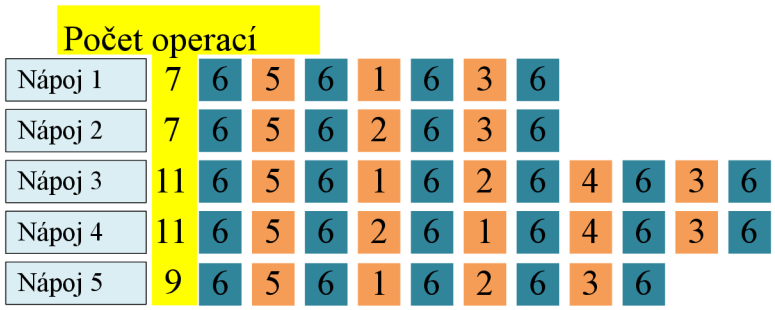
Každá operace je charakterizována dobou trvání (viz Tab.4.2) a pořadím. U kaž-



Obr. 4.7: Rozvržení testbedu pro demonstrační příklad

.

dého nápoje jsou pevně dány první 3 operace, stejně tak poslední operace. Jedná se o přemístění sklenice ze zásobníku na první stanoviště a z posledního stanoviště na výdejní pás. První operace robota (počáteční operace výroby každého nápoje) je delší než ostatní operace prováděné manipulátorem, protože může být konkrétní nápoj prvním objednaným po delší době a robot by musel nejprve přejít do aktivního módu. V praxi by tato situace pravděpodobně nastala pouze u první objednávky. Jak bylo nastíněno dříve, doba, kterou sklenice setrvává na dopravníkovém pásu, než se dostane k zákazníkovi, není započítávána do optimalizační úlohy. Pokud by u některého nápoje bylo více operací ze stejného stanoviště (např. více druhů alkoholu nebo při ředění džusu vodou apod.), lze tyto úkony shrnout do jedné finální operace, která bude mít větší dobu trvání.



Obr. 4.8: Posloupnost operací pro demonstrační příklad

.

Tab. 4.2: Doba trvání jednotlivých operací u demonstračního příkladu.

Index operace	0	1	2	3	4	5	6	7	8	9	10
Nápoj 1 [s]	3	1	2	3	2	2	2				
Nápoj 2 [s]	3	1	2	3	2	2	2				
Nápoj 3 [s]	3	1	2	3	2	3	2	3	2	2	2
Nápoj 4 [s]	3	1	2	3	2	3	2	3	2	2	2
Nápoj 5 [s]	3	1	2	3	2	3	2	2	2		

4.5 Řešení výpočetními prostředky

V této kapitole budou uvedena řešení pomocí solverů představených v kapitole 3.6. Aplikovány jsou postupy popsané v kapitole 4.3.

4.5.1 IBM ILOG CPLEX Optimization Studio

V IBM ILOG CPLEX Optimization Studiu vychází formulace z dostupných příkladů na *Job-Shop Scheduling* a *Flexible Job-Shop Scheduling* při použití programu s omezujícími podmínkami (CP). První příklad předpokládá stejný počet operací, jako stanovišť (nedostatek byl zmíněn v kapitole 4.3). Druhý je naopak rozšířením optimalizačního projektu Barman - pracuje i s možností, že lze jednu operaci provést na více stanovištích. Žádný ovšem neřeší omezení zajišťující realizovatelnost (popsané v úvodu kapitoly 4.3). Bylo tedy nutné vytvořit vlastní modely, jak program s omezujícími podmínkami, tak model pro CPLEX. Data jsou modelu předávána v datovém souboru `.dat`. Jeho formát a příklad je v příloze A.3.

Program s omezujícími podmínkami (CP) dosáhl optimálního řešení problému s 5 nápoji za 0,76 s při použití kriteriální funkce (4.1). Při použití kriteriální funkce

Tab. 4.3: Přehled časů dokončení jednotlivých nápojů od jejich uvedení do výroby při použití 2 různých kritérií v IBM ILOG CPLEX Optimalizačním studiu.

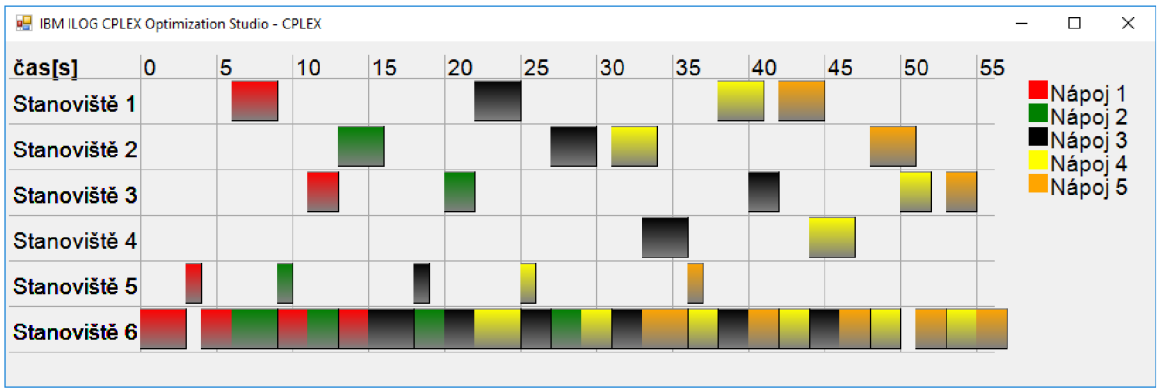
-	CPLEX - časy dokončení		CP - časy dokončení	
Objednávka	krit. (4.2) [s]	krit. (4.1) [s]	krit.(4.2) [s]	krit. (4.1) [s]
Nápoj 1	15	15	15	15
Nápoj 2	29	19	29	24
Nápoj 3	46	41	46	41
Nápoj 4	55	53	55	53
Nápoj 5	57	65	57	65

(4.2) bylo dosaženo optima za 1,11 s, přičemž výsledek byl celkově přínosnější (došlo k urychlení výroby 5 nápojů o 8 s). Počet proměnných byl 71 a počet omezení 173. Zdrojový kód modelu je v příloze A.1.

Program CPLEX optimalizátoru našel optimální řešení za 2,56 s při použití kriteriální funkce (4.1), tedy sumy konečných časů dokončení všech nápojů. Při použití druhé funkce (4.2), sumy koncových časů všech operací, bylo nalezeno optimální řešení za 6,13 s. Bylo zřejmé, že pro optimalizační problém Barman (a podobné problémy rozvrhu) nejsou pro tento typ solveru optimální, protože některé typy nejsou při použití CPLEX vůbec k dispozici, například `interval`, nebo `sequence`. Nejen, že tyto typy umožňují rychlé vytvoření modelu, ale nabízí přehledné zobrazení výsledků v Ganttově diagramu.
Výsledek optimalizace pro kriteriální funkci (4.2) je na Obr.4.9. Tab.4.3 ukazuje výsledky optimalizace pro obě kriteriální funkce. Některé časy se liší, což je pravděpodobně způsobeno různou formulací modelu (a tedy různou interní reprezentací proměnných) pro optimalizátory CPLEX a CP.
Pro všechny CPLEX simulace byla konstanta L (viz model v kapitole 4.3.3) nastavena na 5000.
Zdrojový kód modelu CPLEX je v příloze A.2.

4.5.2 LPSolve

Pro přípravu `.lp` skriptu byl vytvořen program v jazyce C# . Formulace řešení pro demonstrační příklad 4.4 v programu `LPSolve` obsahovala celkem 1665 omezení a 773 proměnných. Nalezení optimálního řešení trvalo 59,25 s. Z toho je zřejmé, že pro rozsáhlejší problémy, kterým tento příklad z hlediska optimalizace je, je klíčové použít soubory pro matematické modelování, se kterými pracuje i studio od IBM,



Obr. 4.9: Výsledný Ganttův diagram pro program CPLEX i CP optimalizátoru v IBM ILOG CPLEX OS při použití kritéria (4.2).

Tab. 4.4: Přehled časů dokončení jednotlivých nápojů od jejich uvedení do výroby při použití 2 různých kritérií v LPSolveIDE-5.5.2.5.

Objednávka	krit. (4.2) [s]	krit. (4.1) [s]
Nápoj 1	18	15
Nápoj 2	34	24
Nápoj 3	47	41
Nápoj 4	53	53
Nápoj 5	60	65

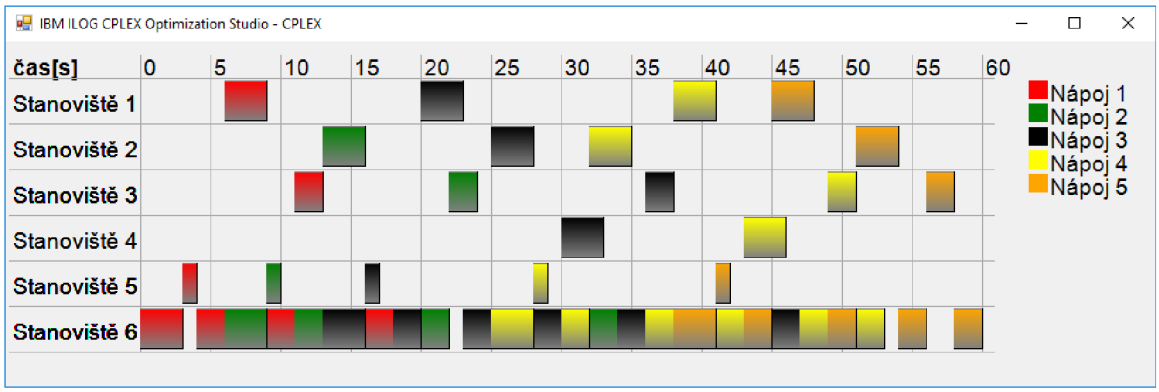
tedy `.mod`[2] nebo `.mps`[3] namísto prosté formulace v souboru `.lp`. Tabulka výsledků optimalizace pro obě kriteriální funkce je v Tab.4.4. Odlišnosti časů oproti Tab.4.3 jsou způsobeny mírně odlišnou formulací problému.

4.5.3 Rozdílnost modelů

Vytvářením matematického modelu `.mod` v optimizačním studiu od IBM je složité zajistit úplnou shodu s `.lp` modelem. Přestože CPLEX optimalizátor umožňuje export do `.lp` souboru (opět s odlišnou syntaxí), ani po úpravě syntaxe skriptem nebylo možné při spuštění v `LPSolveIDE` dostat ani jedno proveditelné řešení.
Pro všechny simulace byla konstanta L (viz model v kapitole 4.3.3) nastavena na 5000.

[2]Nutno uvést, že syntaxe pro tvorbu modelu je odlišná od syntaxe `.mod` souborů v IBM CPLEX OS. Tvorba `.mod` modelu pro LPSolveIDE není v rámci práce řešena.

[3]Nejméně uživatelsky přívětivé. Formulace lineárního problému maticově. Tvorba `.mps` modelu není v rámci práce řešena.



Obr. 4.10: Výsledný Ganttův diagram pro program v prostředí LPSolveIDE 5.5.2.5 při použití kritéria (4.2).

4.6 Porovnání optimalizací pro různá vstupní data

Přestože demonstrační příklad sám ukazuje výhody a možnosti optimalizačního algoritmu, je na místě provést srovnání výsledků optimalizace pro různé konfigurace vstupních dat (objednaných nápojů). Jak vyplynulo z kapitoly demonstračního příkladu 4.4, je „silnějším“ kritériem pro optimalizační problém Barman kritérium (4.2). Proto je toto kritérium aplikováno při vytváření přehledu.
Vytvořený přehled výstupů z optimalizačního studia IBM ILOG CPLEX je v Tab.4.6

Tab. 4.5: Porovnání časů výpočtu pro Tab.4.6.

	Počet operací na nápoj	CPLEX [s]	CP [s]	LP Solve
5 nápojů	7	1,2	0,8	5,7
	11	18,6	0,9	180[4]
	mix	3,9	0,8	61,6
10 nápojů	7	40[4]	2,00	180[4]
	11	40[4]	2,2	180[4]
	mix	40[4]	3,1	180[4]

a Tab.4.5. Tab.4.6 ukazuje změny efektivity využití robotického manipulátoru a celkové časy výroby pro sadu 5 a 10 nápojů. Obě skupiny jsou zkoumány ve třech různých konfiguracích, kdy mají všechny nápoje 7 stejných operací (např. alkohol s ledem), 11 stejných operací (míchaný nápoj s ledem) nebo se jedná o mix různých nápojů. Tabulka ukazuje nejprve výsledky bez optimalizace (3. sloupec) - prosté řazení nápojů za sebe, následně pro optimalizaci se solverem CPLEX a s CP optimalizací. Na první pohled je možné ověřit, že efektivita robota je ve všech případech

[4]Jedná se o časový limit nastavený v optimalizačním studiu/prostředí LPSolveIDE, protože čas řešení byl příliš velký (minuty). V těchto případech se tedy jedná o suboptimální řešení

přes 90 %. Při bližším prozkoumání je možné zjistit, že efektivita je větší, jestliže je předmětem optimalizace sada nápojů se stejným počtem operací. Tento rozdíl se zvětšuje s množstvím nápojů v rámci jednoho optimalizačního problému. V Tab.4.5 jsou potom uvedeny doby výpočtu pro solvery CPLEX a CP. Podle hodnot je opět dokázáno, že CPLEX optimalizátor není příliš vhodný pro typ problémů jako je Barman (problémy rozvrhu, *scheduling problems*).
Při výpočtu stejných příkladů v prostředí LPSolve jsou časy opět řádově vyšší, proto byl celkový časový limit nastaven na 3 minuty. Výsledky programu LPSolve jsou v Tab.4.7. Srovnáním s Tab.4.6 lze dojít k závěru, že v případech, kdy se během časového limitu podařilo nalézt optimální řešení (sada 5 různých nápojů a sada stejných 5 nápojů o 7 operacích) jsou výsledky pro prostředí LP Solve a program CPLEX stejné. Tato situace by měla platit při absolutní shodě modelů pro všechny případy. Ve všech případech simulací CPLEX a LPSolve (tabulky Tab.4.6 a Tab.4.7)je konstanta L = 5000. Zdrojové soubory pro optimalizaci CP a CPLEX studia IBM jsou stejné jako u demonstračního příkladu (pouze se měnily datové soubory) - v příloze A.

Tab. 4.6: Porovnání výsledků problému Barman s různými vstupními konfiguracemi v optimalizačním studiu IBM ILOG CPLEX.

	Počet operací na nápoj	Bez opt.	CPLEX		CP	
		Dokončení [s]	Využití robota[%]	Dokončení [s]	Využití robota[%]	Dokončení [s]
5 nápojů	7	75	92	49	92	49
	11	125	93	70	90	72
	mix	100	92	60	90	61
10 nápojů	7	150	95	95	96	94
	11	250	94	138	95	137
	mix	199	90	122	92	119

4.6.1 Vliv pořadí jednotlivých operací

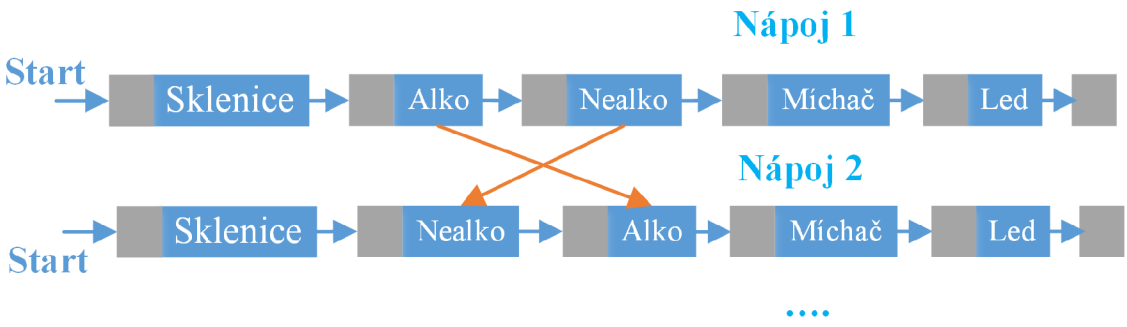
Až doposud byly prováděny veškeré simulace s pevně danou posloupností operací v rámci výroby nápoje. Při činnosti Barmana ale na pořadí jednotlivých operací[5] vždy nezáleží, např. u míchaného nápoje lze zaměnit pořadí operací. Tab.4.8 ukazuje porovnání výsledných časů zpracování 5 a 10 stejných míchaných nápojů, přičemž každý druhý nápoj v pořadí měl přehozené složené operace na stanovišti alkoholických a nealkoholických nápojů (Obr.4.11). Pro výsledky z LPSolve byl opět nastaven

[5]V tomto kontextu se jedná samozřejmě o operace složené.

Tab. 4.7: Porovnání výsledků problému Barman s různými vstupními konfiguracemi v optimalizačním studiu IBM ILOG CPLEX.

	Počet operací na nápoj	Bez opt.	LP Solve	
		Dokončení [s]	Využití robota[%]	Dokončení [s]
5 nápojů	7	75	92	49
	11	70	89	73
	mix	60	92	60
10 nápojů	95	95	92	98
	11	138	86	153
	mix	122	90	126

časový limit 3 min. Jedná se tedy o suboptimální řešení. Při porovnání výsledků lze dojít k překvapivému závěru, že po záměně byly výsledky horší u většiny případů, kromě situace s 5 nápoji při řešení solverem LPSolve a situace s 10 nápoji při řešení CPLEX.



Obr. 4.11: Ukázka záměny složených operací na prvních dvou nápojích.

4.6.2 Nedokonalosti

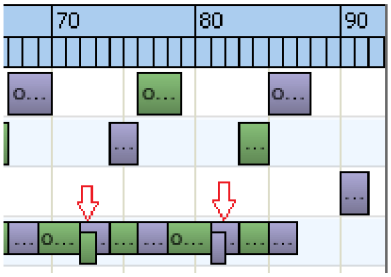
V rámci provádění porovnávacích simulací v IBM ILOG Optimalizačním studiu i výsledků z LPSolveIDE byly výsledné počáteční časy jednotlivých operací ilustrovány v Ganttově diagramu pomocí programu v jazyce C# [6]. Přestože studio od IBM umožňuje zobrazení výsledků v Ganttově diagramu pro programy s omezujícími podmínkami, bylo nutné ověřovat informace ve vytvořeném programu, protože v některých případech dával výsledný diagram z IBM optimalizačního studia zkreslené

[6]Program byl převzat z codeprojects.com a upraven. Je licencován otevřenou licencí CPOL.

Tab. 4.8: Porovnání výsledků pro míchané nápoje (11 operací) s různými a pevně danými uspořádáními operací.

	CPLEX [s]		CP [s]		LP Solve	
	různé	stejné	různé	stejné	různé	stejné
5 nápojů	73	70	77	72	72	73
10 nápojů	137	138	147	137	163	153

informace (viz Obr. 4.12). Vzniklá chyba je způsobena tím, že nejsou během optimalizace využita všechna stanoviště (není využit zásobník nealk. nápojů a míchač). Pro korektní zobrazení by bylo nejprve nutné definovat pouze použitá stanoviště (změnit datový soubor - tj. z použitých stanovišť 1,3,5,6 budou stanoviště 1 - 4).



Obr. 4.12: Chyba zobrazení výsledků optimalizace programu s omezujícími podmínkami v Ganttově diagramu (případ 10 stejných nápojů o 7 operacích).

4.7 Ukázková aplikace

V rámci této kapitoly optimalizačního problému Barman bylo provedeno množství simulací a porovnání dat z různých příkladů. Zatím ovšem nebyly popsány detailnější podrobnosti týkající se kódu, konkrétní formulace problémů pro každý použitý optimalizační software včetně souborů vstupních dat (IBM ILOG). Tyto podrobnosti jsou skryty za popsanou teorií. Dílčí programy v jazyce c# , které byly použity při tvorbě nebo vyhodnocení modelů jsou zahrnuty jako funkce v ukázkové aplikaci. V rámci této aplikace jsou popsány některé části týkající se převážně datových souborů studia IBM ILOG [7].

Tato ukázková aplikace demonstruje ucelení všech informací týkajících se optimalizačního problému barman (ale také rozvrhování obecně) do určitého modulárního celku, který bude použitelný při realizaci plánovacího modulu jako subsystému MES.

[7] Operace právě v tomto formátu jsou vstupním souborem aplikace

4.7.1 Použití LP solve API

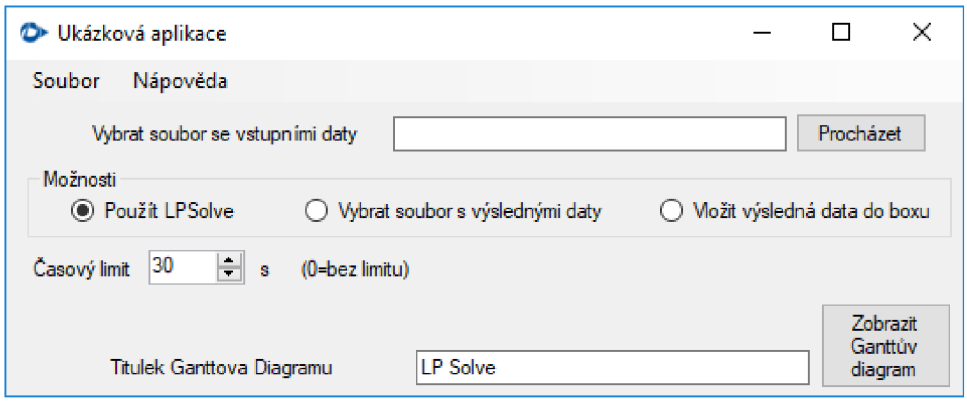
Pro použití jsou potřeba 2 soubory: externí knihovna `lpsolve55.dll` a c# wrapper `lpsolve55.cs` se statickými metodami. Pro použití je nutné zkopírovat knihovnu `lpsolve55.dll` a `lpsolve55.cs`, které jsou v adresáři instalace LPSolveIDE), do složky projektu (knihovnu ideálně k ostatním binárním souborům). Aby bylo možné projekt s `lpsolve55.cs` spustit, je potřeba u konkrétního projektu nastavit v *Properties* projektu na záložce *build* povolit *allow unsafe code* aby nedocházelo ke spouštění vyjímek. C# tímto zajišťuje indikaci, že jde o „nebezpečný kód“, kterým je v tomto případě použití ukazatelů a operací s nimi při přístupu k externí `.dll` knihovně. Dále, aby bylo možné `lpsolve` použít, je nutné ještě v kartě *build* ve vlastnostech projektu nastavit *Platform Target* na `x86` (*All configurations*), protože knihovna je 32 bitová.

4.7.2 Hlavní funkce ukázkové aplikace

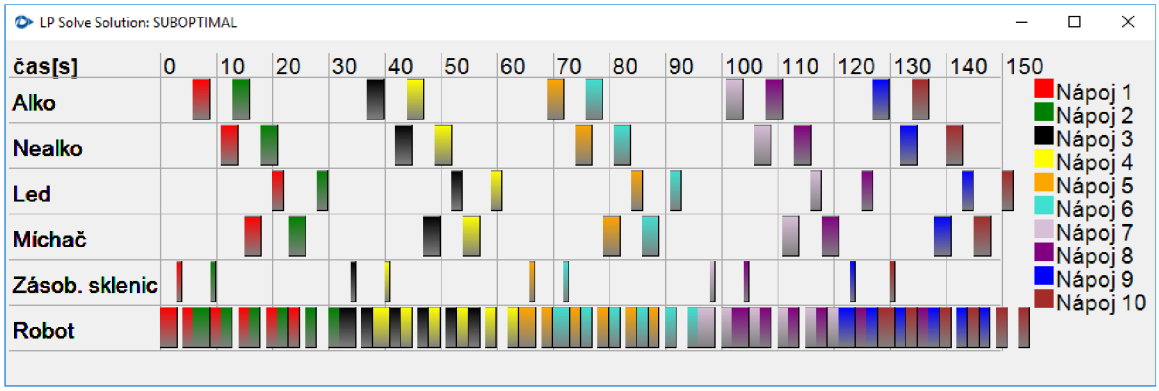
Po spuštění aplikace (Obr.4.14) je potřeba zadat cestu k souboru se vstupními daty (`OpsSingle`), jejichž formát je popsán v příloze A.3 nebo v nápovědě programu. Dále jsou na výběr možnosti, jestli použít LPSolve API (default) nebo vybrat soubor s výslednými daty (startovní časy všech operací v pořadí jejich identifikátoru - viz `OpsSingle`) nebo data vložit ručně do TextBoxu.
Co se týká použití LPSolveAPI, je dále možné nastavit časový limit simulace (*timeout*). Ten je ve výchozím stavu nastaven na 30 s (jak bylo popsáno dříve, optimalizace je časově náročná i pro 5 nápojů). Při adekvátním nastavení *timeoutu* může solver vrátit suboptimální řešení, jinak se zobrazí hláška, že je nastavená doba výpočtu pro zvolený model (vstupní data) příliš malá. Během nastaveného času (provádění optimalizace) je zobrazen čekací kurzor, tj. aplikace neočekává od uživatele žádné akce.
Další možností je vložení dat formou prostého textu nebo textového souboru (při použití IBM ILOG nebo LPSolveIDE), formát je popsán v nápovědě[8]. Pak už jen stačí modifikovat titulek Ganttova diagramu a zobrazit Ganttův diagram[9]. Výsledek pro příklad se stejnými 10 nápoji (při timeoutu nastaveném na 30 s) je na Obr.4.14. Jak je zřejmé z obrázku, je v titulku diagramu také uvedeno řešení - tedy jestli se jedná o optimální nebo suboptimální. V rámci jednoho spuštění aplikace je možné zobrazení více diagramů z různých zdrojů.
Další funkcí programu je (menu Soubor) export dat pro IBM ILOG. Přestože jsou

[8]Modely z IBM ILOG studia v rámci postprocessingu exportují výsledné časy v požadovaném formátu do souboru `modelRun.txt`

[9]Současný program umožňuje pouze zobrazení 10 nápojů. Rozsáhlejší optimalizace by vyžadovali zásah do zdrojového kódu.



Obr. 4.13: Hlavní okno ukázkové aplikace.



Obr. 4.14: Ganttův diagram pro příklad 10 stejných nápojů s 11 operacemi.

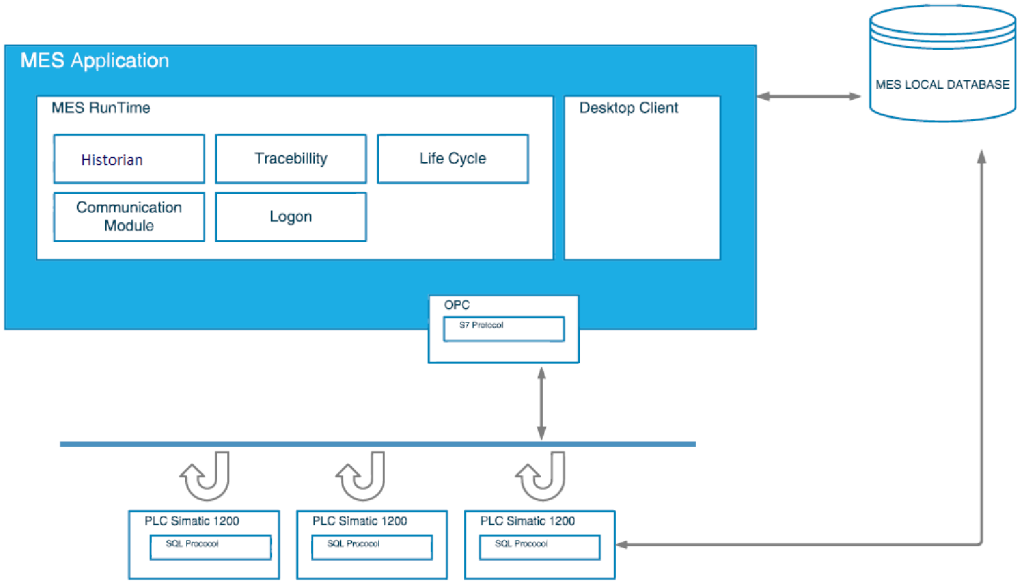
všechny operace nápojů (vstupní data) požadována právě ve formátu pro studio od IBM, jsou zde další struktury, které zajišťují správnou funkci modelu v IBM ILOG CPLEX optimalizačním studiu. Jednou z nich je struktura *OpsCompound*, která plní funkci složené operace (popsáno v teorii při tvorbě modelu). Další je sada parametrů popisující počet operací apod. a nakonec struktura priorit, která přiřazuje nejvyšší prvnímu nápoji v pořadí a nejnižší poslednímu.
Poslední funkcí je export `.lp` modelu, tedy dat pro LPSolveIDE. Tu lze nalézt opět přes položku menu Soubor.

5 Současný stav poskytnutého MES systému

V této kapitole je popsána původní verze MES systému, která byla vytvořena v rámci diplomové práce [44]. Tato aplikace je napsána v jazyce C#.
Při návrhu byla použita databáze MSSQL od firmy Microsoft. Aplikace je založena na komerčním frameworku Telerik. Tento nabízí oproti standardním komponentám (např. MVVMLight) mnoho výhod, a to jak z hlediska uživatele (interaktivní komponenty - viz Obr.5.2), tak z hlediska programátora (způsob práce s daty).

5.1 Architektura aplikace

Původní architektura je na Obr.5.1. Díky protokolu třetí strany umožňovala implementovat ukládání dat do databáze přímo ve vývojovém prostředí PLC.
Program je vytvořen v prostředí Visual Studia 2015 jako WPF aplikace. To znamená, že umožňuje jednoduchou adaptaci programu na návrhový vzor MVVM, který byl při návrhu použit. Ten slouží k oddělení uživatelského grafického rozhraní od zbytku programu. Podrobnější popis je ve zmíněné diplomové práci [44].
Jak je zřejmé z Obr.5.1, aplikace obsahuje modul pro komunikaci s PLC (*Communication Module* potažmo OPC), dále modul pro správu a zobrazení profilu uživatele (*Logon*), případně entit (bude popsáno dále).

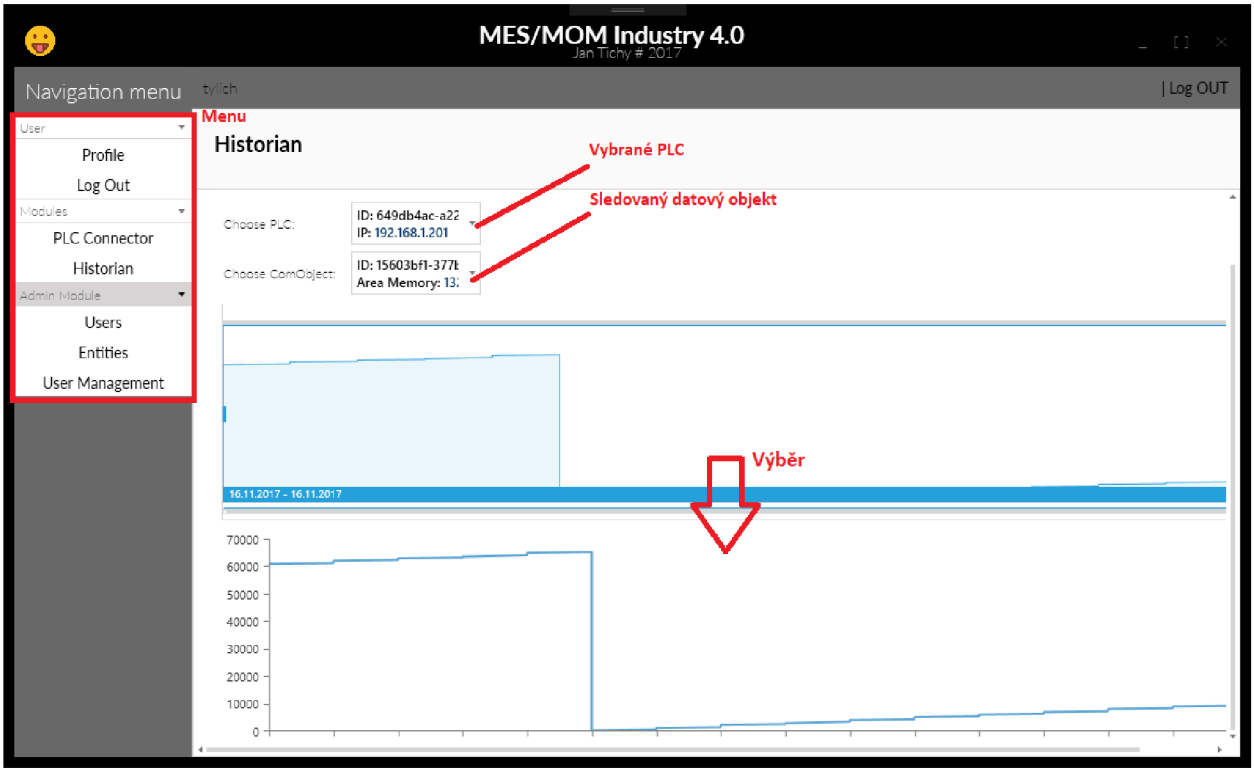


Obr. 5.1: Architektura původního MES systému [44]

5.2 Funkce aplikace a programu[1]

Možnosti aplikace (menu) a přehled hlavního modulu je na Obr.5.2. Program po spuštění zobrazí přihlašovací formulář. Po úspěšném zadání přihlašovacích údajů se zobrazí profil uživatele v hlavním okně programu. Přes položku *Profile* nebo *User Management* může uživatel upravit svůj profil. Po výběru *PLC Connector* z menu *Modules* je možné přidat PLC. Po přidání lze přiřadit objekty, které mají být sledovány (např. celé číslo z Data bloku 1 s offsetem 0). V menu *PLC Connectoru* a stisknutí tlačítka *start* u přidaného PLC dojde k vyčítání dat z PLC s nastavenou periodou. Tato data je potom možné sledovat v modulu *Historian* (při výběru přidaného PLC a sledovaného paměťového objektu v rozbalovacích lištách okna). Po výběru konkrétní oblasti z grafu pomocí kurzoru myši se zobrazí detail pod grafem (Obr.5.2).

Z pohledu programátora je aplikace psána jako „těžký klient“, tedy kdy většina operací běží na straně klienta a server funguje jen jako úložiště dat.



Obr. 5.2: Modul historianu původního MES systému

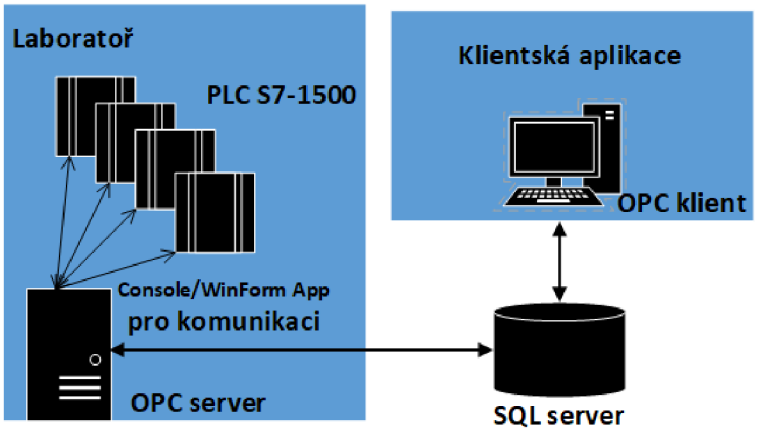
[1]Program byl testován s databází MySQL, jelikož byl tento DB stroj vybrán pro budoucí aplikaci.

6 Návrh systému MES/MOM

Nejprve je potřeba určit architektura systému korespondující s databázovým modelem. Dále je nutné stanovit požadavky na softwarové nástroje.

6.1 Architektura programu

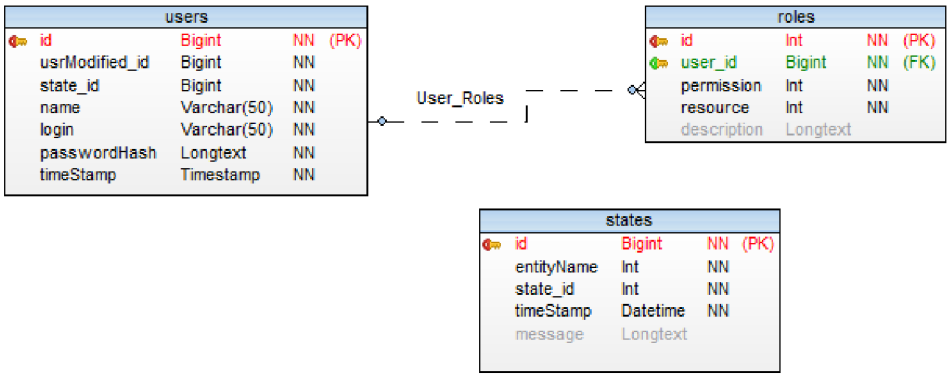
Na rozdíl od původního programu popsaném v kapitole č.5, bude aplikace tvořena jako tenký webový klient, kde veškerý sběr dat z PLC bude proveden z OPC serveru, který bude přímo ukládat data do databáze. Klientská aplikace bude určovat konkrétní objekty, které jsou z PLC vyčítány. Pokud dojde ke změně, OPC server změnu zaregistruje při vyčítání dat z databáze a ihned aktualizuje seznam objektů.



Obr. 6.1: Architektura navrhovaného MES systému

6.2 Navržený datový model

Při návrhu databáze je klíčovým modelem tzv. Entitně - relační (ER) model (někdy také označovaný jako ER diagram). Při pohledu na tento diagram jsou ihned zřejmé vazby mezi jednotlivými entitami. Při podrobnějším pohledu by diagram sám o sobě měl dát základní představu o funkci systému. Model byl zpracován v programu *TOAD Data Modeler* 6.3.
U navrhovaného systému MES je pro zjednodušení oddělena část pro správu uživatelů a část pro správu stavů od jádra aplikace. Chybějící databázové propojení pomocí cizích klíčů bude realizováno v programu.



Obr. 6.2: Datový model části pro správu MES systému

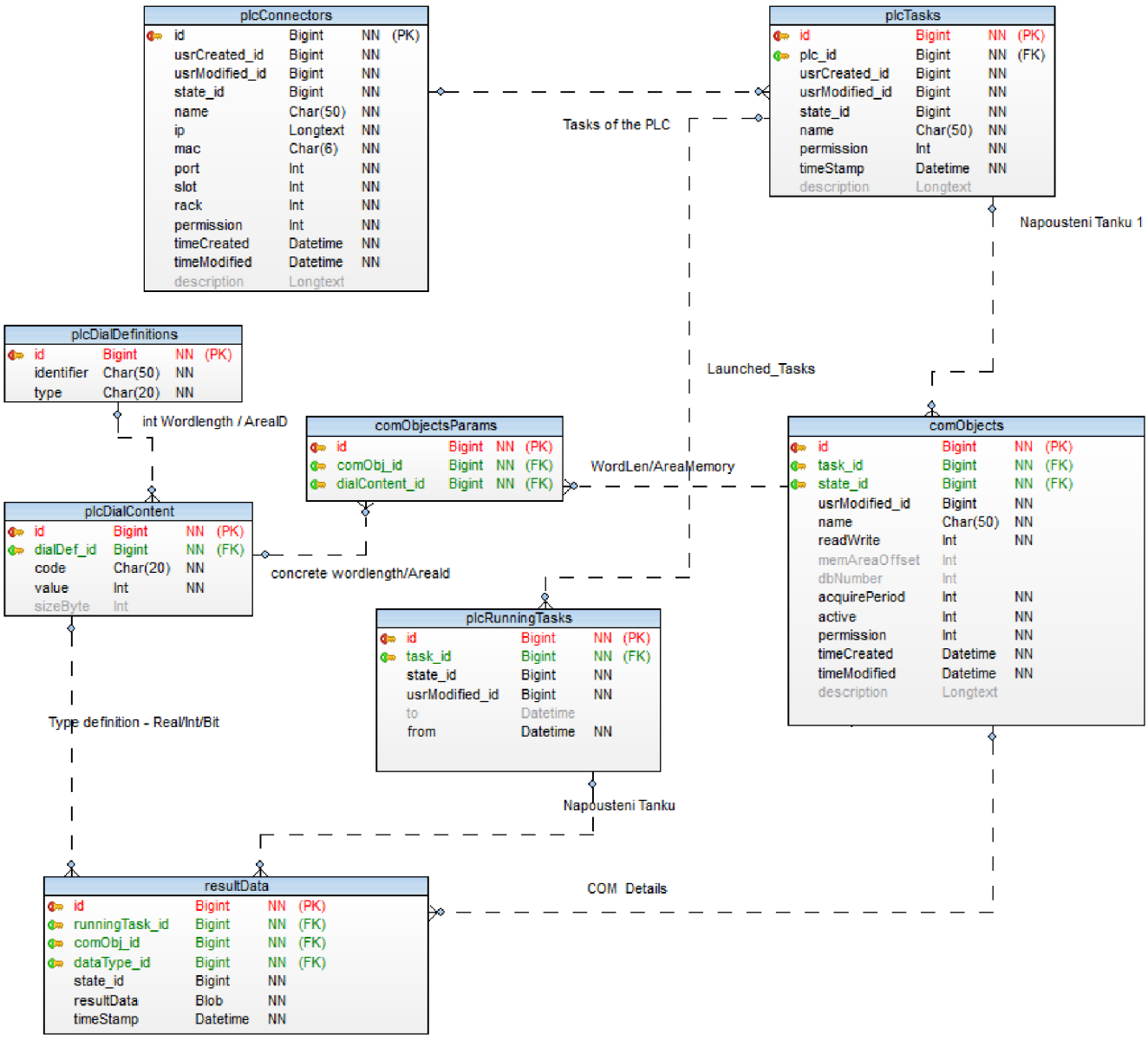
6.2.1 Část pro správu jednotlivých modulů

Do této části se řadí část pro správu uživatelů a správu stavů.
Tabulka uživatelů `users` obsahuje pouze základní údaje potřebné k přihlášení uživatele. Tato tabulka je nadřazená tabulce rolí. Ta obsahuje kromě klíčů dvě důležité položky - samotnou roli a povolení (`resource` a `permission`). Tyto celočíselné proměnné jsou v programu reprezentovány výčtovým typem Enum.
Aktuální řešení přiřadí každému uživateli tři role (`useradmin`, `plcadmin` a `comadmin`), přičemž každá role má přiřazeno povolení (žádná práva, právo čtení/ právo čtení a zápisu). Každá z těchto rolí se váže k určité skupině objektů. Za administrátora je považován uživatel s právem čtení a zápisu.
Další důležitou součástí pro administraci funkčního jádra programu je informace o stavech (`states`). Tabulka reprezentuje stavy všech potřebných entit (tedy ostatních tabulek). Díky časové značce je možné dohledat a seřadit postupně stavy např. konkrétního PLC. Samotný stav je opět reprezentován výčtovým typem v programu (`entityName`). `state_id` např. k indikaci, jestli je konkrétní stav ještě aktivní nebo ne (realizováno dalším typem Enum). Tabulka stavů je z hlediska databázového modelu jedináček. Jedná se především o zjednodušení modelu a větší otevřenosti ke změnám. Záznamy stavů zatím nejsou implementovány (signalizace stavů pomocí typu Enum).

6.2.2 Funkční část

Jedná se o jádro celé aplikace. Datový model je použit samotnou klientskou aplikací a dílčí část je sdílena s OPC serverem pro informace o objektech a zápis výsledných dat.
Tabulka `plcConnectors` reprezentuje informace o jednom konkrétním PLC. Kromě jména a adres (fyzické MAC a síťové IP) obsahuje identifikátory umístění (slot a

rack). V budoucnu bude položka `permission` zajišťovat např. zobrazení konkrétního PLC pouze určité skupině uživatelů.
V rámci konkrétního PLC se může vytvářet spousta úloh (`plcTasks`), z nichž může být některá spuštěná (`plcRunningTasks`). V rámci každé úlohy si může uživatel (student) nadefinovat, které objekty (proměnné z DB, merkery) chce vyčítat. Každý takový objekt je definován v tabulce `comObjects`. Obsahuje název (`name`), informaci, zda je ještě aktivní (`active`), povolení a časovou značku (v případě objektu z DB také offset a číslo datového bloku). Ke `comObjectu` se ale váží další informace, jako délka dat v bytech a konkrétní datový typ. Tyto informace ke konkrétnímu `comObjectu` jsou pomocí klíčů uchovávány v tabulce `comObjectsParams`. Všechny možnosti jsou v tabulce `plcDialContent`. Jestli se jedná o délku nebo konkrétní typ zajišťuje vazba k `plcDialDefinition`. Jedná se vlastně o tabulky číselníku, které mohou být použity i pro další parametry (např. chybová hlášení komunikační knihovny apod.).
K ukládání výsledných dat slouží tabulka `ResultData`. Ta nese informace, ke které úloze data patří (`runningTask_id`), o jaký se jedná `ComObject` (`comObj_id`). Protože jsou výsledná data uchovávána v databázovém typu blob, je nutné zjistit, kolik bytů data mají pro zpětnou konverzi (pole bytů není jako databázový typ příliš rozšířené).



Obr. 6.3: Datový model funkční části MES systému

7 Realizace systému MES/MOM

Budou stručně předvedeny vybrané technologie, frameworky případně další řešení použité pro výslednou aplikaci. Aplikace je psána v jazyce C#, především proto, že knihovna pro komunikaci s PLC byla napsána v jazyce C#.

7.1 Práce s daty

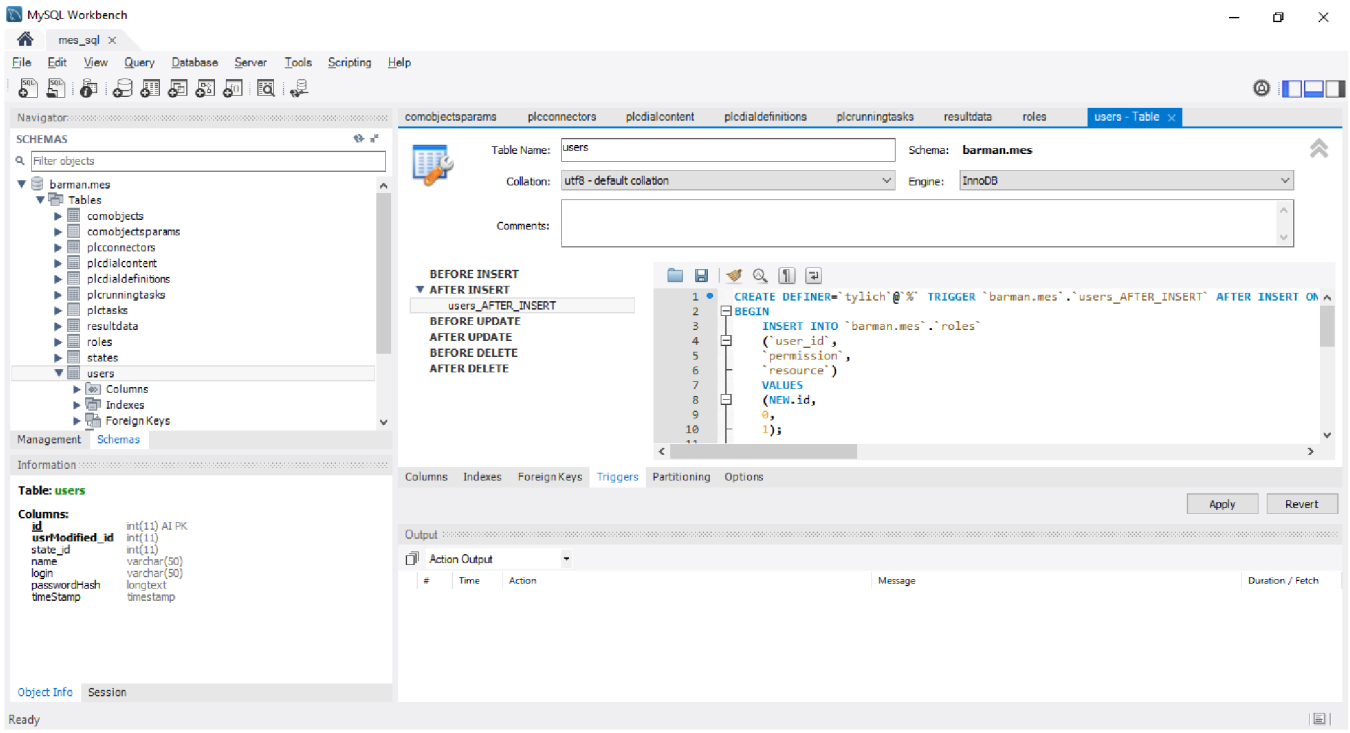
V této kapitole budou popsána použitá řešení pro práci s daty v navrhovaném systému MES.

7.1.1 MySQL

Pro ukládání dat na server byl vybrán databázový stroj MySQL ve verzi 5.7. Byl vybrán především díky rovnocenné podpoře tří nejrozšířenějších operačních systémů - Windows, Linux a Mac OS X. Z dalších výhod lze uvést větší flexibilita co se týká podpory ukládacích strojů [46]. Hlavním přínosem pro práci s MySQL bylo použití MySQL Workbench, tedy grafického rozhraní pro práci s databází. Po vytvoření databáze nabízí, kromě funkcí exportu dat a průvodce migrací, také export přehledného a editovatelného ER diagramu. Vkládání dat do tabulek, editace tabulek, přidávání triggerů (Obr.7.1) a další jsou samozřejmostí.
MySQL nabízí také oproti konkurenčním systémům (např. MSSQL) výhody vyplývající z různorodosti syntaxe (např. zobrazení x-tého a y-tého řádku v rámci jednoho příkazu), ale i z aplikace schémat (např. innoDb - pro zachování integrity).

7.1.2 ADO.NET

Jako technologie pro práci s daty, tedy pro komunikaci s databází byla zvolena architektura ADO.NET. Hlavním důvodem výběru byl přístup tzv. `database-first`, tedy „první-kód“, kdy se nejpve musí vytvořit databáze, od které se potom odvíjí modely a repozitáře v programu. Přestože je tvorba modelu znatelně pracnější než u přístupů `code-first` nebo `model-first`, není spolehlivost systému odkázána na softwarový framework, jako je tomu u ostatních dvou přístupů.
ADO.NET představuje množinu tříd nabízejících služby pro přístup k datům (informace v DB) a tvorbu databázových aplikací. Mezi jeho přednosti patří především jednoduchý způsob použití, rychlost při zpracování a další. Stačí vytvořit spojení se serverem, se kterým se pracuje, pomocí zvoleného adaptéru a zadaného dotazu získat z databáze data a ty pak načíst do některé z připravených konstrukcí pro práci s daty z tabulek [45] (v programu reprezentováno jako `List<Udt>`, kde `Udt`



Obr. 7.1: Editace triggeru v MySQL Workbench editoru.

reprezentuje vytvořený typ pro práci).
ADO.NET může být použit i pro jiné poskytovatele než je Sql (tzv. *ADO.NET providers*) jako Oracle, OleDb, Odbc a další.
Konkurenčním přístupem může být například EntityFramework (jako `code-first` - „první-kód“), který byl využit u původního programu.
V rámci tvorby rozhraní nad databází byl použit již hotový bázový repozitář, který byl v průběhu vývoje modifikován.

7.2 Softwarové řešení

Nejprve budou posány frameworky a doplňky použité pro tvorbu aplikace, potom budou popsány komunikační a klientská aplikace.

7.2.1 DotVVM

Jedná se o Open-source framework založený na .NET frameworku od české firmy Riganti. Hlavní výhodou je, že umožňuje psaní graficky propracovaných stránek bez znalosti JavaScriptu. Je založený na návrhovém vzoru *Model–view–viewmodel* (MVVM). Ten se ve stručnosti skládá ze tří částí, tedy **Model** (symbolizující data v databázi) **ViewModel** (prostředník mezi daty a tím co vidí uživatel) a **View**

(Pohled - to, co vidí uživatel). MVVM využívá možnosti *Windows Presentation Foundation* (WPF), tedy tzv. databinding, což je vazba dat (obvykle z ViewModelu) na zobrazované prvky.
Přestože je DotVVM OpenSource, některé komponenty jsou pouze v placené verzi (např styly pro stránkování, modální formuláře atp). Tyto komponenty jsou implementovány pomocí javascriptu.

7.2.2 Bootstrapious

Vhodná šablona pro tvorbu webových stránek je opět součástí placeného balíčku DotVVM. Bootstrapious je opět projekt českého původu. Autoři využívají knihoven bootstrapu a vytváří šablony pro webové stránky s otevřeným kódem. Jediným omezením je logo ve spodní části stránky.

7.2.3 Visual Studio 2015 a doplňky

Jako vývojářský editor je použito Visual Studio 2015. Jako většina vývojových nástrojů disponuje pokročilými možnostmi ladění (vrátit se o pár řádků kódu zpět, přidávání breakpointů za běhu a další). Jeho předností je doplněk ReSharper od firmy JetBrains, který nabízí spoustu funkcí, především pro refaktoring, ale i další funkce například zobrazování stisknutých zkratek v prezentačním módu a další. Tento doplněk je placený, nicméně pro studenty je zdarma.
Další výhodou Visual Studia je softwarová báze tzv. *NuGet*, kde je možné stáhnout spoustu nástrojů a doplňků, které šetří čas a mnohdy také spoustu práce.

AutoMapper Je to doplněk, který je schopen namapovat jednu entitu (třídu) na jinou. Pokud jsou stejné názvy vlastností, stačí pouze jedna řádka kódu a lze získat např. místo `PlcConnector` s více než pěti záznamy `PlcConnectorMin`, který má pouze adresu a jméno PLC.

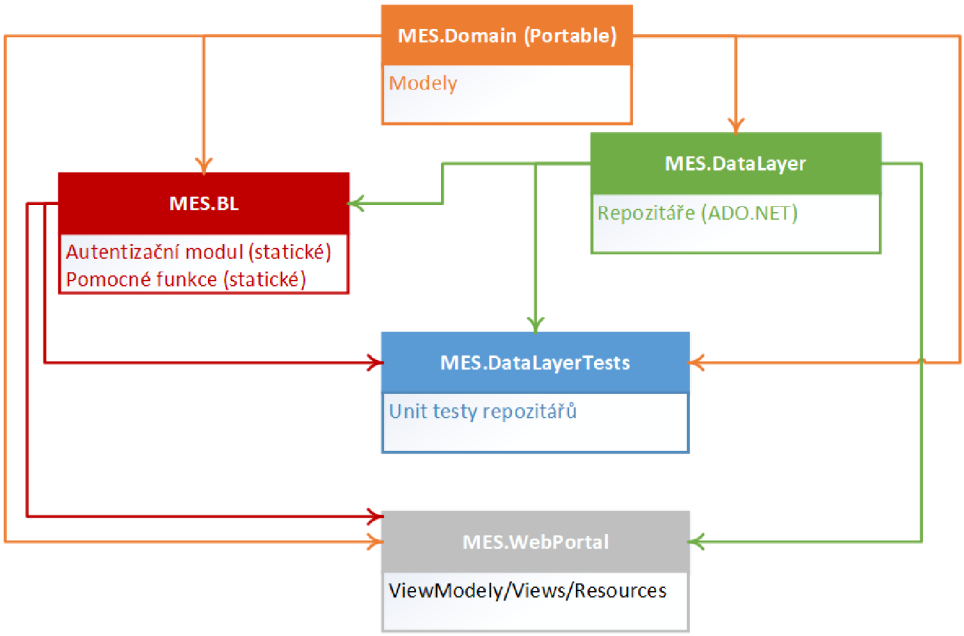
7.2.4 Komunikace s PLC

Za tímto účelem byla použita knihovna Sharp7, tedy stejná knihovna jako v původním systému MES [44]. Nespornou výhodou knihovny je OpenSource licence *GNU General Public License*. Přestože byla původně navrhovaná pro CPU řady S7-300, S7-400, je komunitou zajišťována podpora i nových PLC firmy Siemens (CPU S7-1200, CPU S7-1500). Pro zajištění správné funkce je nutné mít ve vývojovém prostředí PLC S7 (TIA Portal) programu ve vlastnostech PLC povolenu funkci *PUT/GET* a mít vypnutou optimalizaci datových bloků (ve výchozím stavu je tento parametr povolený, ve vlastnostech příslušného databloku je možnost jej vypnout).

7.2.5 Realizace klientské aplikace

Klientská aplikace je webová aplikace, prostřednictvím které si uživatel zobrazí data z libovolného PLC v laboratoři kdekoliv na světě. Struktura programu (všechny projekty a jejich vzájemné reference) je na Obr.7.2. Následuje popis dílčích částí klientského programu systému MES, některé obrazovky jsou v příloze B.

Po spuštění aplikace se zobrazí přihlašovací formulář. Pokud jde o nového uživatele,



Obr. 7.2: Projekty klientského programu a jejich vzájemné reference.

který nemá vytvořený účet, musí se nejprve zaregistrovat. Po zadání jména, hesla a vygenerování loginu je uživateli vytvořen účet a je přesměrován opět na úvodní stránku. Po přihlášení má uživatel základní práva, tedy editaci uživatelských údajů, zobrazení, spouštění a přidávání PLC úloh. Dále zobrazení historických dat v modulu Historianu. Menu MES systému je na Obr. 7.3.

Uživatelská práva jsou rozdělena do 3 kategorií. Práva uživatelů, práva pro práci s PLC a práva pro práci s komunikačními objekty. Změna pravidel pro nového uživatele (práva `NONE` nebo `READ`) je možná pouze uživatelem s většími právy (`READWRITE`). V rámci dalších dvou kategorií má opět uživatel se základními právy omezené možnosti (nemůže přidávat ani editovat PLC / komunikační objekty). Administrátor uživatelů (`READWRITE` v kategorii *UserAdmin*) může editovat práva všech ostatních uživatelů stejně tak jejich profilové informace. Všechny tyto položky týkající se správy jsou v položce menu *Uživatelé*.

PLC a jejich správa je pod záložkou Administrace PLC. Administrátor může přidávat, měnit a odstraňovat PLC. Základními parametry je Název, IP adresa, MAC adresa a série parametrů Port, Slot, Rack, je směrodatná pro použitou komunikační knihovnu Sharp7.

PLC Úlohy je položka menu, která umožňuje pro uživatele s příslušnými právy přidávání, editaci i mazání jednotlivých úloh. Spuštěním úlohy ji již nelze editovat ani mazat. To platí i pro vlastnosti příslušného PLC a pro všechny ostatní úlohy příslušící k tomuto PLC, včetně komunikačních objektů.

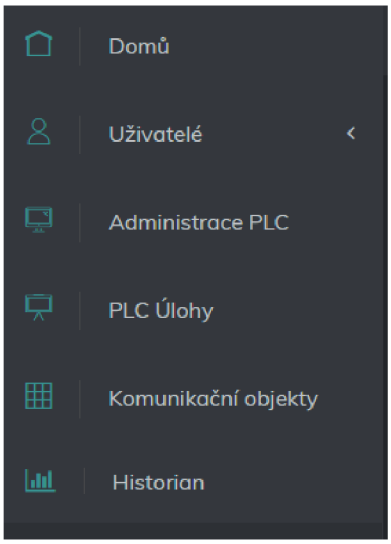
Komunikační objekty jsou všechny datové struktury v paměti PLC. Ať už jde o bitové, celočíselné nebo reálné merkery, datobloky, timery, procesní vstupy nebo procesní výstupy. Při přidávání objektu je potřeba zadat jeho název, vybrat příslušnou úlohu zvolit datový typ a místo v paměti. V případě, že jde o datový blok, je potřeba nastavit jeho číslo a offset[1].
Nutno uvést, že výsledná data jsou v tabulce `resultData` uložena v binární struktuře BLOB ve formě bytového pole. Podle typu proměnné se mohou vyčítat 1 - 4 B. Omezením jsou zde bitové proměnné, které se vyčítají po 1 B, je tedy nutné tento fakt zohledňovat při návrhu programu. Dalším omezením jsou datové typy `Udt`, ze kterých se nepodařilo získat data (např. Real, Int apod.), přestože příslušné databloky byly korektně nastavené.

Historian umožňuje zobrazení vybraných komunikačních objektů. Pro výběr konkrétního úseku slouží textboxy, u kterých ale není ošetřeno špatné zadání vstupu. Příklad zobrazení vývoje hladiny tanku v čase je v příloze B.4.

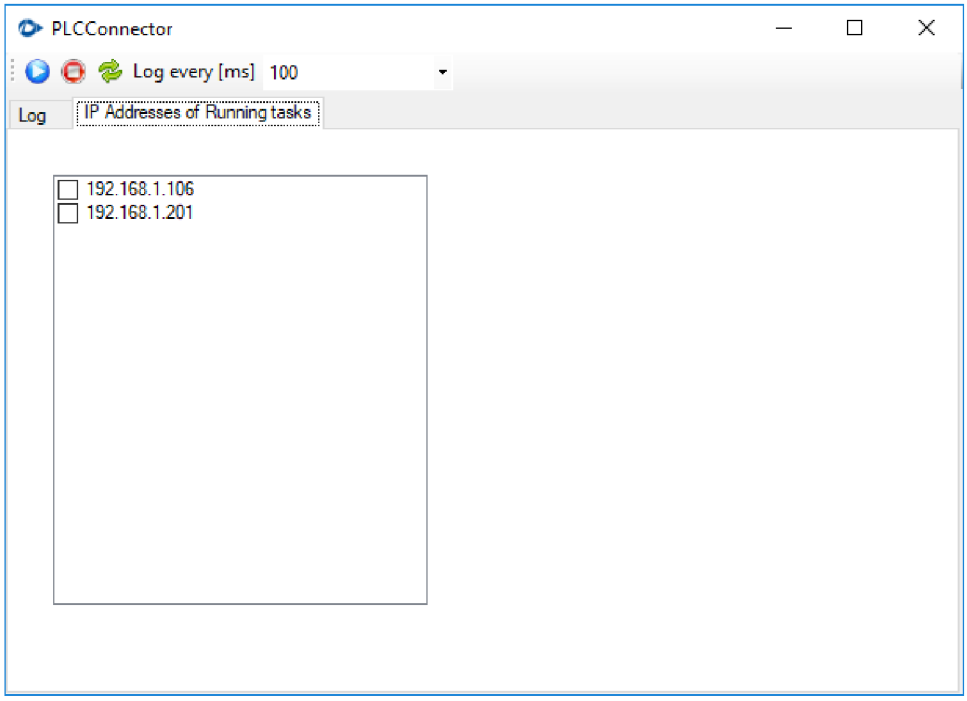
7.2.6 Realizace aplikace pro komunikaci

Aplikace pro vyčítání dat z PLC musí být spuštěna v laboratoři. Struktura projektů aplikace a jejich vzájemné reference jsou na Obr.B.3. Bude periodicky kontrolovat spuštěné úlohy (`plcRunningTasks` viz Obr.6.3). Všechny komunikační objekty (`ComObject`), které přísluší ke konkrétnímu `plcTasku` se začnou vyčítat do databáze. Aplikace *Windows forms*, která bude dávat administrátorům k dispozici log dat a seznam PLC, ze kterých jsou právě vyčítaná data. Těmto funkcím ale předchází autorizace uživatele přihlašovacím dialogem. Okno aplikace po přihlášení je na Obr.7.4. Okno obsahuje 2 záložky, přičemž jedna je určena pro logování dat a ve druhé jsou na výběr IP adresy spuštěných úloh. Po volbě IP adres zatrhnutím

[1] Ten je viditelný jako sloupec při otevření konkrétního databloku, jestliže je vypnuta optimalizace - viz 7.2.4.

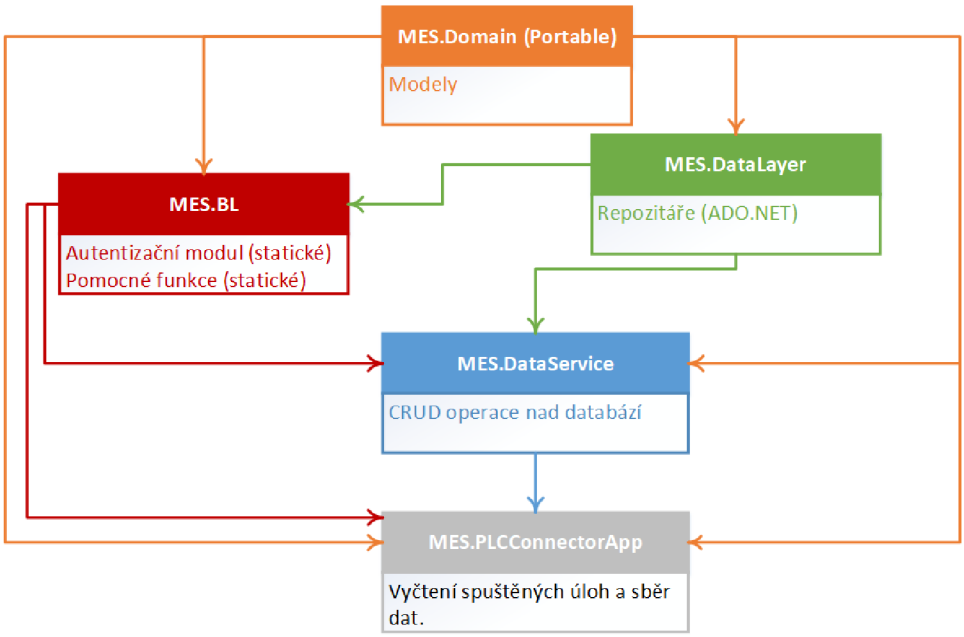


Obr. 7.3: Menu klientské aplikace MES systému.



Obr. 7.4: Dialogové okno aplikace pro komunikaci sběr dat z PLC.

(jestliže jsou dosažitelné v rámci sítě) a spuštění logování modrým tlačítkem začnou být vyčítána data z komunikačních objektů nastavených v klientské aplikaci. Nastavený čas nastavitelný rozbalovací lištou vedle ovládacích tlačítek určuje frekvenci logování dat, perioda vyčítání dat z PLC je nastavena v klientské aplikaci jako atribut komunikačního objektu.



Obr. 7.5: Projekty komunikačního programu a jejich vzájemné reference.

7.3 Zhodnocení MES systému a porovnání

Z porovnání obou systémů plyne, že funkce systému byly zachovány, bylo změněno rozhraní a architektura aplikace. Pokud jde o vzhled, je uživatelsky přívětivější současná verze. Nelze ale opomenout nesporné výhody komerčního řešení Telerik, které přináší do uživatelkého rozhraní komfort (interaktivní komponenty), který místy kompenzoval celkový vzhled. Tab.7.1 srovnává původní a současné řešení jádra MES systému.

Tab. 7.1: Srovnání současné a původní aplikace MES

Přidávání a editace uživatelů	Původní MES	Současný MES
Kontrola vstupních dat (formát, přihlašovací údaje)	✓	✓
Interaktivní komponenty	✓	×
Přidělování práv uživatelům	×	✓
Šifrované operace s hesly	×	✓
Vyčítání dat z PLC S7	✓	✓

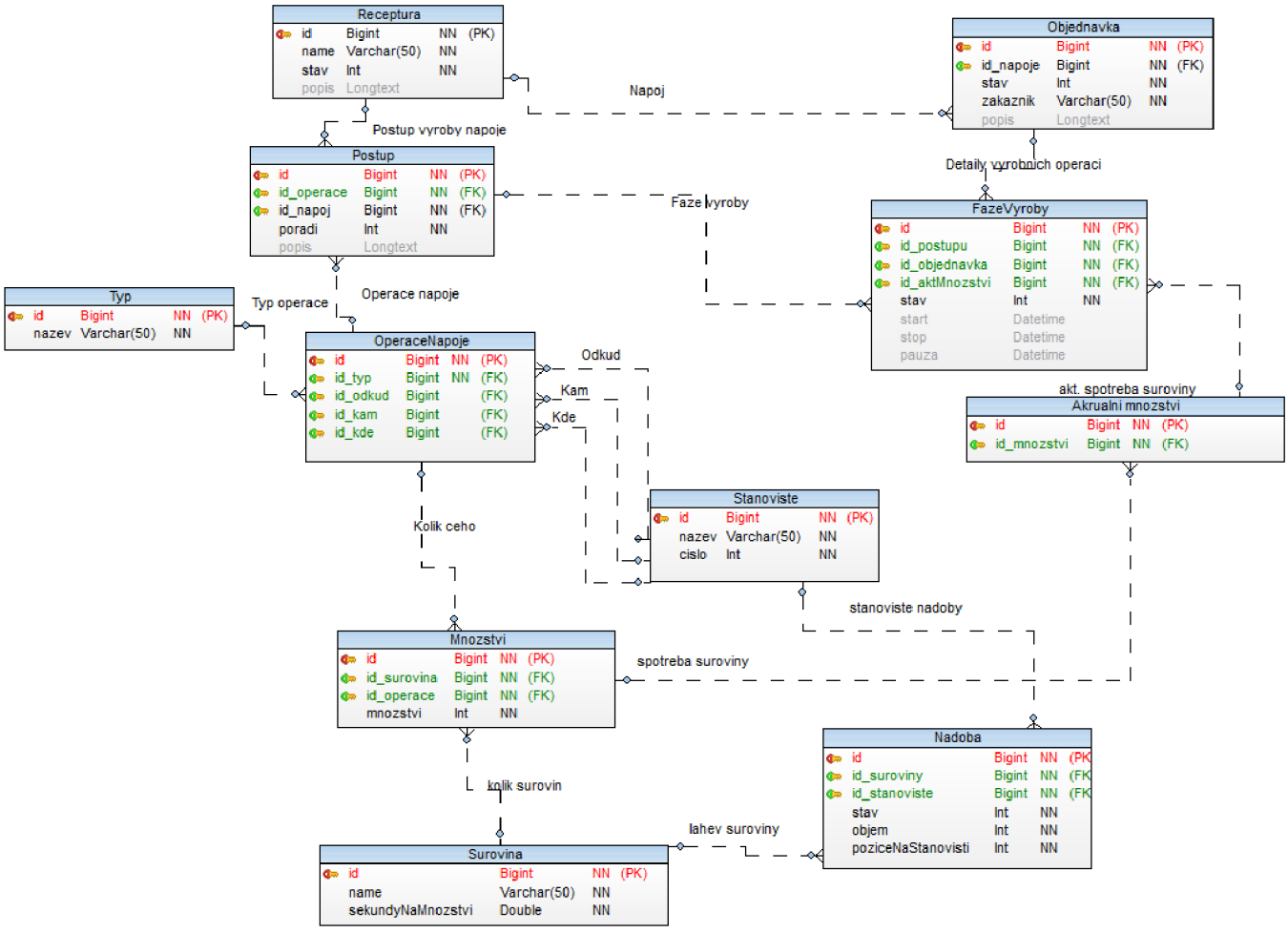
8 Modul pro plánování výroby v systému MES

Tato kapitola osvětluje začlenění systému pro plánování výrobních operací jako subsystému MES.

8.1 Rozšíření datového modelu

Základním kamenem pro vytvoření plánovacího subsystému v systému MES je soubor datových struktur pro ukládání veškerých potřebných dat, tedy návrh datového modelu. Datový model pro pro navrhovaný subsystém je na Obr.8.1. Lze jej rozdělit na dvě sekce - dynamickou a statickou. Dynamickou částí jsou tabulky `Objednavka`, `FazeVyroby` a `Aktualni mnozstvi`. Ostatní tabulky jsou statické, jejich záznamy jsou směrodatné pro výrobu nápojů podle objednávek, ale nejsou závislé na počtu objednávek nebo jejich dalších parametrech.

Nejvýše v hierarchii je tabulka `Receptura`, Tam je uložen konkrétní nápoj (např.



Obr. 8.1: Datový model subsystému pro plánování výrobních operací.

Mojito), který se skládá ze sady operací, které jsou agregovány v tabulce `Postup`, kde také každá operace získá pořadí (index), který určuje řazení při výrobě. `Operace napoje` obsahuje informaci, o jaký typ operace se jedná (přesun robotem nebo výrobní činnost), a dále jakého stanoviště se operace týká (`kde` v případně výroby nebo `odkud` a `kam` v případě operace přesunu robotem). Odtud vyplývá, proč nemají cizí klíče `id_ odkud` , `id_ kam` , `id_ kde` nastaven příznak `NN` (*Not Null*). Tabulka `Operace napoje` je potom nadřazená tabulce `Mnozstvi`, ze které je možné zjistit odebrané množství suroviny (to se získá součinem atributů `mnozstvi` ze stejnojmenné tabulky a `sekundyNaMnozstvi` z tabulky `Surovina`). Díky dynamické tabulce `Aktualni mnozstvi`, která má informace o stavu výroby z `FazeVyroby`, je možné zjistit aktuální množství suroviny v nádobě[1].
Konkrétní diagramy životních cyklů objednávky a výroby nápoje by byly předmětem konkrétní implementace. Nyní lze pouze nastínit průběh. Po přijetí objednávky je ve stavu *pořízená* a čeká na zpracování. Před uvedením do výroby jsou nastaveny počáteční časy všech operací receptury objednaného nápoje (atributy `start` a `stop` tabulky `FazeVyroby`) Následně dochází k výrobě, kdy každá operace v rámci postupu přechází ze stavu *čekající* přes stav *v procesu* . Jestliže následně přejde operace do stavu *přerušeno* , potom je nastaven čas `pauza` jako atribut tabulky `FazeVyroby`. Pokud dojde k přerušení systému výroby v čase, kdy jsou všechny operace dokončené nebo čekají na zpracování, může být provedena nová optimalizační úloha. K původní optimalizační úloze se přidají nově objednané nápoje a u objednávek, které mají stav *vyrábí se*, ale nemají hotové všechny operace, se v nové optimalizační úloze objeví pouze ještě neprovedené operace.
Popsaný datový model byl stejně jako při návrhu datového modelu jádra MES vytvořen v programu *TOAD Data Modeler* 6.3.

8.2 Modifikace MES

Modifikace pro výrobní informační systém obsahuje implementaci navrženého datového modelu na 8.1. Jednalo by se tedy o tvorbu všech potřebných formulářů a následně definici všech statických dat do databáze (Receptury jednotlivých nápojů).

8.3 Modifikace aplikace rozvrhu pro použití v praxi

Jak vyplývá z výsledků simulací v kapitolách 4.6 a 4.4, dělají časy výpočtu úlohu optimalizačního problému Barman, při objednávce několika desítek nápojů, nepo-

[1]Pro tuto strukturu je předpoklad *on/off* plnění suroviny do sklenic bez možnosti jakékoliv regulace.

užitelnou. Za tolerovatelnou dobu výpočtu lze považovat dobu do 10 s. V kapitole 4.7 ukázkového příkladu je naznačena možnost využití suboptimálního řešení, které lze pro většinu problémů do 10 nápojů získat ve zmíněném časovém limitu. Přestože byla při tvorbě modelu využita jistá zjednodušení (4.3.1), lze problém rozdělit a zpracovávat např. prvních 10 nápojů. Jakmile budou první nápoje hotové, proběhne nová optimalizační úloha s čekajícími operacemi[2] původních nápojů a všemi operacemi nových nápojů.

V rámci kapitoly 4.6.1 bylo zjištěno, že vliv záměny operací při několika stejných objednávkách nemá žádný časový přínos, proto tento fakt ani nebyl zohledněn při návrhu datového modelu. Jediným východiskem pro úsporu času může být změna receptury nápojů. Například míchaný nápoj s obsahem alkoholu i nealkoholického nápoje nemusí přejít přes stanoviště míchače, protože lze předpokládat, že se obsah promíchá navzájem.

Co se týče implementace, lze jako předlohu využít Ukázkový program použitý při provádění simulací na optimalizačním problému Barman. Ten je volně dostupný i pro komerční aplikace (licence GNU).

[2]Opět nutno podotknout, že jsou v tomto kontextu myšleny operace složené.

9 Závěr

V rámci první kapitoly byly posány funkce MES systému od raných fází jeho vývoje. Následně byly uvedeny hranice systému MES v kontrastu s dalšími systémy zajišťující fungování podniku jako celku, jejich možné překrývání a vstupně/výstupní toky dat. V kapitole 2 byla popsána problematika plánování výrobních operací. Kapitola vytvořila čtenáři povědomí o plánování a rozvrhování ve výrobě, zároveň ale uvedla plánování výrobních operací do souvislosti se systémem MES a dalšími systémy v historickém kontextu. Následně byla provedena rešerše matematických metod pro plánování výrobních operací. V rámci rešerše byly aplikovány základní kroky algoritmů na jednoduchých příkladech. Následně byly uvedeny možnosti aplikace vybranými výpočetními prostředky. Z těch byly vybrány 2 nástroje pro tvorbu matematických modelů, IBM ILOG Optimalizační studio a LP Solve.
Kapitola čtyři byla zaměřena na rozbor projektu Barman (vyvíjeném na ústavu ÚAMT) jako optimalizačního problému. V rámci této kapitoly byly využity informace z rešerše a byl uveden vzorový příklad, rozvrhu práce (*job-shop schedulingu*). Přesnou definicí problému a aplikací struktur použitých pro vzorový problém rozvrhu práce, byl vytvořen matematický model pro optimalizační problém Barman. Byly provedeny simulace na jednoduchém demonstračním příkladu. Dále byl vytvořen přehled a srovnání výsledků simulací pro různá vstupní data.
V rámci kapitol 5 až 7 je zadokumentována transformace současného jádra MES systému na nekomerční platformu. V kapitole pět byly shrnuty přínosy a stav poskytnuté verze MES, v kapitole šest byl pak proveden návrh systému. Jednalo se o ucelení datového návrhu systému, tedy jeho funkční části a části správy. Kapitola sedm popisuje zprovoznění systému, převedení do webového rozhraní postaveném na open-source frameworku, využití datových struktur a popis funkce.
V kapitole osm byl uveden návrh a kroky vedoucí k implementaci plánovacího modulu do systému MES. Byl vytvořen datový model a s využitím výsledků simulací a informací o návrhu modelu rozvrhu byly popsány kroky vedoucí ke korektní implementaci modulu.

Literatura

[1] MESA Model Evolution. MESA International. 2011, 11/4/2011(4), 21.

[2] MESA - základem je komunikace. AUTOMA. 2009, 01/2009, 2. Dostupné z: <http://automa.cz/Aton/FileRepository/pdf_articles/38463.pdf>.

[3] SCHOLTEN, Bianca. Integrating ISA-88 and ISA-95. ISA EXPO 2007. 2007, 2007(10/2007), 13. Dostupné z: <https://www.isa.org/pdfs/integrating-isa-88-and-isa-95/>.

[4] MES vs. MOM Panel Webcast [online]. MESA, 2014 [cit. 2018-02-10]. Dostupné z: <https://services.mesa.org/resourcelibrary>.

[5] MES Explained: A High Level Vision. MESA International - White Paper number 6. 1997.

[6] MESA Metrics Guidebook: ROI and Justification for MES. A MESA International Guidebook. 5/5/2014.

[7] PÁSEK, Jan. Automatizace procesů část 1: Řízení výroby MES - Standard S95. 2017. Přednáška. UAMT FEKT VUT v Brně.

[8] SWANTON, Bill a Alison SMITH. MES for long-term revenue and market benefits: AMR Research, USA. Elektron. 2005, 2005(02), 4.

[9] Critical Manufacturing - Download White Paper [online]. Critical Manufacturing, 2015 [cit. 2018-02-21]. Dostupné z: <http://www.criticalmanufacturing.com/en/resources/documents/mes-performance-scalability/download>.

[10] Operations Scheduling [online]. Supplement J [cit. 2018-03-06]. Dostupné z: <http://wps.prenhall.com/wps/media/objects/7117/7288732/krm9e_SuppJ.pdf>.

[11] Production Scheduling [online]. [cit. 2018-03-06]. Dostupné z: <http://mcu.edu.tw/~ychen/op_mgm/notes/scheduling.html#top>.

[12] Chapter 17: Operations Scheduling [online]. [cit. 2018-03-06]. Dostupné z: <web4.uwindsor.ca/users/b/.../Lecture7_scheduling_f06_604.ppt>

[13] GROSSMANN, Ignacio E. Overview of Optimization Models for Planning and Scheduling [online]. Center for Advanced Process Decision-making [cit. 2018-03-07]. Dostupné z: <http://egon.cheme.cmu.edu/ewo/docs/

EWOSchedulingGrossmann.pdf>. Department of Chemical Engineering Carnegie Mellon University Pittsburgh, PA.

[14] WILSON, Shellyanne. Sequencing in Process Manufacturing: The Product Wheel Approach [online]. In: . [cit. 2018-03-07]. Dostupné z: <https://www.pomsmeetings.org/confpapers/051/051-1171.pdf>.

[15] BAŠTINEC, Jaromír. Statistika, stochastické procesy, operační výzkum [online]. Brno, 2014 [cit. 2018-03-22]. Dostupné z: <http://matika.umat.feec.vutbr.cz/inovace/texty/DMA1/CZ/DMA1_plna_verze_CZ.pdf>. Skriptum. UMAT FEKT VUT v Brně.

[16] JENSEN, Paul A. a Jon BARD. Linear Programming Terminology. Dostupné z: <http://www.me.utexas.edu/~jensen/or_site/models/unit/lp_model/lp_terms/lp_terms.html>. University of Texas.

[17] LUENBERGER, David G. a Yinyu. YE. Linear and nonlinear programming. 3rd ed. New York, NY: Springer, c2008. ISBN 978-0-387-74502-2.

[18] HLADÍK, Milan. Celočíselné Programování [online]. Praha, 2017 [cit. 2018-03-23]. Dostupné z: <https://kam.mff.cuni.cz/~hladik/CP/text_cp.pdf>. Text k přednášce. Univerzita Karlova v Praze.

[19] DVOŘÁK, Jiří. Celočíselné programování [online]. [cit. 2018-03-23]. Dostupné z: <http://www.uai.fme.vutbr.cz/~jdvorak/vyuka/tsoa/Pred08.ppt>. Přednáška. Vysoké učení Technické v Brně.

[20] What Is Constraint Programming?. Constraint.org [online]. 2013 [cit. 2018-03-25]. Dostupné z: http://<www.constraint.org/en/intro.html>

[21] RUDOVÁ, Hana. Constraint Programming and Scheduling [online]. [cit. 2018-03-25]. Dostupné z: https://www.fi.muni.cz/~hanka/konstanz09/slides01_bw.pdf>

[22] HLADÍK, Milan. Základy Nelineární Optimalizace [online]. Praha, 2016 [cit. 2018-03-26]. Dostupné z: https://kam.mff.cuni.cz/~hladik/ZNO/text_zno.pdf>. Text k přednášce. Univerzita Karlova v Praze.

[23] BRUCKER, Peter. Scheduling algorithms. 5th ed. New York: Springer, c2007. ISBN 978-3-540-69515-8.

[24] LIMA, Ricardo. IBM ILOG CPLEX: What is inside of the box? [online]. [cit. 2018-03-27]. Dostupné z: http://egon.cheme.cmu.edu/ewo/docs/rlima_cplex_ewo_dec2010.pdf>. Prezentace. Carnegie Mellon University.

[25] CPLEX Optimizer. Ibm.com [online]. [cit. 2018-03-27]. Dostupné z: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer>

[26] Linear Programming [online]. [cit. 2018-04-05]. Dostupné z: https://developer.ibm.com/docloud/documentation/optimization-modeling/lp/>

[27] Introduction to lp_ solve 5.5.2.5. Lpsolve.sourceforge.net [online]. [cit. 2018-03-27]. Dostupné z: http://lpsolve.sourceforge.net/5.5/>

[28] Microsoft Solver Foundation 3.1. Lpsolve.sourceforge.net [online]. [cit. 2018-03-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff524509(v=vs.93).aspx>

[29] Simplex Method and Non-Linear Programming. International Journal of Computational Science and Mathematics [online]. 2012, 4(3), 5 [cit. 2018-03-25]. ISSN 0974-3189. Dostupné z: https://www.ripublication.com/irph/ijcsm/ijcsmv4n3_15.pdf>

[30] LUENBERGER, David G. a Yinyu. YE. Linear and nonlinear programming. 3rd ed. New York, NY: Springer, c2008. ISBN 978-0-387-74502-2.

[31] KACZMARCZYK, Václav, Ondřej BAŠTÁN, Zdeněk BRADÁČ a Jakub ARM. Industry 4.0 Testbed (Self-acting barman): principles and design (do 1.5.2018 nevydaný).

[32] HLOSKA, Jiří. Národní centrum Průmyslu 4.0: éra průmyslu 4.0 v ČR byla oficiálně zahájena. Automa. 2017, 2017(8-9), 2.

[33] ABREU, Pedro. [online]. Porto [cit. 2018-03-31]. Dostupné z: https://paginas.fe.up.pt/~prodei/dsie09/docs/pedroabreu/apresentacao.pdf>. Přednáška. Faculdade de Engenharia da Univ. Porto.

[34] The Job Shop Problem [online]. Google Optimization Tools, 2017 [cit. 2018-02-11]. Dostupné z: https://developers.google.com/optimization/scheduling/job_shop>

[35] KUHPFAHL, Jens. Job Shop Scheduling – Formulation and Modeling. KUHPFAHL, Jens. Job Shop Scheduling with Consideration of Due Dates [online]. Wiesbaden: Springer Fachmedien Wiesbaden, 2016, 2016-6-25, s. 9-23 [cit. 2018-03-31]. DOI: 10.1007/978-3-658-10292-0_ 2. ISBN 978-3-658-10291-3. Dostupné z: http://link.springer.com/10.1007/978-3-658-10292-0_2>

[36] ÖZGÜVEN, Cemal, Lale ÖZBAKIR a Yasemin YAVUZ. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Applied Mathematical Modelling. 2009, 2010(34), 10.

[37] SCHAUER, Joachim a Cornelius SCHWARZ. Job-shop scheduling in a body shop. Journal of Scheduling [online]. 2013, 16(2), 215-229 [cit. 2018-04-14]. DOI: 10.1007/s10951-012-0300-2. ISSN 1094-6136. Dostupné z: http://link.springer.com/10.1007/s10951-012-0300-2>

[38] RONDEAU, Patrick a Lewis LITTERAL. The evolution of manufacturing planning and control systems: From reorder point to enterprise resource planning. Lacy School of Business, 2001. Butler University.

[39] MCKAY, Kenneth N. a Gary W. BLACK. The evolution of a production planning system: A 10-year case study. Computers in Industry [online]. 2007, 58(8-9), 756-771 [cit. 2018-02-18]. DOI: 10.1016/j.compind.2007.02.002. ISSN 01663615. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S0166361507000206>.

[40] OLHAGER, Jan. Evolution of operations planning and control: from production to supply chains. International Journal of Production Research [online]. 2013, 51(23-24), 6836-6843 [cit. 2018-02-12]. DOI: 10.1080/00207543.2012.761363. ISSN 0020-7543. Dostupné z: <http://www.tandfonline.com/doi/full/10.1080/00207543.2012.761363>.

[41] Focus improvement on the manufacturing constraint [online]. WHAT IS THE THEORY OF CONSTRAINTS?, 2017 [cit. 2018-02-11]. Dostupné z: <https://www.leanproduction.com/theory-of-constraints.html>

[42] Michal Křen: Real-Time optimalizace operací v průmyslové výrobě, diplomová práce, Brno, FIT VUT v Brně, 2012 [cit. 2018-02-11]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=118408>. . Diplomová práce. Vedoucí práce Ing. Martin Hrubý, Ph.D.

[43] The Job Shop Problem [online]. Google Optimization Tools, 2017 [cit. 2018-02-11]. Dostupné z: <https://developers.google.com/optimization/scheduling/job_shop>

[44] TICHÝ, Jan. SYSTÉMY MES/MOM V PROSTŘEDÍ INDUSTRY 4.0 [online]. Brno, 2017 [cit. 2017-11-17]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=146025>. . Diplomová práce. Vedoucí práce Ing. Jan Pásek CSc.

[45] 1. ADO.NET [online]. 2007 [cit. 2018-01-04]. Dostupné z: <http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch07s01.html>.

[46] A Comparison between MySQL vs. MS SQL Server. Medium.com [online]. 10.4.2017 [cit. 2018-05-01]. Dostupné z: <https://medium.com/@mindfiresolutions.usa/a-comparison-between-mysql-vs-ms-sql-server-58b537e474be>

Seznam symbolů, veličin a zkratek

MES Manufacturing Execution System - Výrobní řídicí systém

MOM Manufacturing operations management - Řízení výrobních operací

ERP Enterprise Resource Planning - Plánování podnikových zdrojů

MRP Material Requirements Planning - Plánování potřeb materiálu

MCS Manufacturing Control System - Výrobní řídicí systém

MESA Manufacturing Enterprise Solution Association - Sdružení výrobních podnikových řešení

ISA International Society of Automation - Mezinárodní společnost automatizace

ROP Reorder point

EOQ Economic Order Quantity

SFC Shop Floor Control - Dílenské ovládání

CPS Cyber-Physical System - Kyberneticko-fyzický systém

JIT Just-In-Time

TOC Theory of constraints

MPC Manufacturing Planning and Control - Plánování a řízení výroby

MRP2 Manufacturing Resource Planning - Plánování potřeb materiálu

JSSP Job-shop scheduling problem - problém rozvrhu práce

JSS Job-shop scheduling - rozvrh práce

CP Constrained Programming - Programování s omezujícími podmínkami

LP Linear Programming - Lineární programování

MILP Mixed Integer Linear Programming - Smíšené celočíselné lineární programování

NL NonLinear Programming - Nelineární/matematické programování

MINLP Mixed Integer NonLinear Programming - Smíšené celočíselné nelineární programování

IBM ILOG zkratka pro IBM ILOG CPLEX Optimization Studio používaná v textu

WPF Windows Presentation Foundation

MVVM Model–view–viewmodel

OPC OLE for Process Control

WCF Windows Communication Foundation

API Application Programming Interface - rozhraní pro programování aplikací

Seznam příloh

A Matematické modely problému Barman pro IBM ILOG CPLEX Optimalizační studio

V rámci *postprocessingu* obou modelů jsou výsledné počáteční časy všech operací uloženy do souboru *modelRun.txt*. Odtud mohou být vloženy do Ukázkové aplikace (kapitola 4.7) jako výstupní data a může být zobrazen Ganttův diagram.

A.1 Model optimalizátoru CP

```
using CP;


tuple paramsT{
  int nbJobs;
  int nbMchs;
  int nbOpsSingle;
  int nbOpsCompound;
};
paramsT Params = ...;

int nbJobs = Params.nbJobs;
int nbMchs = Params.nbMchs;

range Jobs = 1..nbJobs;
range Mchs = 1..nbMchs;

int Priorities[j in Jobs] = ...;


tuple OperationSingle {
  int id;     // Operation id
  int jobId; // Job id
  int pos;    // Position in job
  int mch;  // Machine
  int pt; // processing time
};
tuple OperationCompound {
  int id;
  int job;
  int oper1Id;     // Robot
  int oper2Id; // another stage
  int mch2;  // Machine
```

```
};
{OperationSingle} OpsSingle   = ...;
{OperationCompound} OpsCompound = ...;

// Position of last operation of job j
int jlast[j in Jobs] = max(o in OpsSingle: o.jobId==j) o.pos;

dvar interval ops  [op in OpsSingle] size op.pt;
dvar interval opsCompound [op in OpsCompound];
dvar sequence mchs[m in Mchs] in all(op in OpsSingle: op.mch == m) ops[
   op];


execute {
      cp.param.FailLimit = 10000; // set fail limit
}

dexpr int all_ops = sum(j in Jobs, o in OpsSingle: o.jobId == j)
   startOf(ops[o]) * Priorities[j];
dexpr int end_ops = sum(j in Jobs, o in OpsSingle: o.pos==jlast[j] && o
   .jobId == j) startOf(ops[o]) * Priorities[j];

minimize all_ops;//end_ops;//

subject to {
 compoundOperation:
    forall (o1 in OpsSingle, o2 in OpsSingle, o in OpsCompound: o1.id==
       o.oper1Id && o2.id==o.oper2Id)
      {
      startAtStart(opsCompound[o],ops[o1]);
      endAtEnd(opsCompound[o],ops[o2]);
      }
 c01:
    forall (j in Jobs, o1 in OpsSingle, o2 in OpsSingle: o1.jobId==j &&
       o2.jobId==j && o2.pos==1+o1.pos)
      endBeforeStart(ops[o1],ops[o2]);
 c02:
  forall (m in Mchs)
      noOverlap(mchs[m]);
 c04: // compounded intervals
    forall (o1 in OpsSingle, o2 in OpsSingle, o in OpsCompound: o1.id==
       o.oper1Id && o2.id==o.oper2Id && o2.pos==1+o1.pos)
      endAtStart(ops[o1],ops[o2]);
 limitMachine:
     forall (o1a in OpsCompound, o2a in OpsCompound, o1 in OpsSingle,
        o2 in OpsSingle:
      o1a.id != o2a.id && o1a.mch2 == o2a.mch2 &&
```

```
        o1.id != o2.id && o1.id == o1a.oper2Id+1 && o2.id == o2a.
            oper2Id+1 &&
        o1.jobId == o1a.job && o2.jobId == o2a.job)
    {
     startOf(opsCompound[o2a]) - endOf(opsCompound[o1a]) >= 0 && startOf
        (opsCompound[o2a]) - endOf(ops[o1]) >= 0 ||
     startOf(opsCompound[o1a]) - endOf(opsCompound[o2a]) >= 0 && startOf
        (opsCompound[o1a]) - endOf(ops[o2]) >= 0;
    }

}

execute{  // postprocessing
  // start times
  var ofile = new IloOplOutputFile("modelRun.txt");
  for(var i in thisOplModel.OpsSingle){
     ofile.writeln(thisOplModel.ops[i].start+",");
  }
  ofile.close();
  // processing times
  var ofile1 = new IloOplOutputFile("modelProcessTimes.txt");
  for(var i in thisOplModel.OpsSingle){
     ofile1.writeln(thisOplModel.ops[i].size);
  }
  ofile1.close();
}
```

A.2 Model optimalizátoru CPLEX

```
using CPLEX;

execute PARAMS{
  cplex.tilim = 40;    // simulation time limit
  cplex.mipemphasis = 4;     // stop when 4 solutions are found
  cplex.nodealg = 0;     // algorithm selection (0=automatically chosen/
     default)
}

tuple paramsT{
  int nbJobs;
  int nbMchs;
  int nbOpsSingle;
  int nbOpsCompound;
};
paramsT Params = ...;
int L = 5000;
```

```
int nbJobs = Params.nbJobs;
int nbMchs = Params.nbMchs;
int nbOpsSingle = Params.nbOpsSingle;
int nbOpsCompound = Params.nbOpsCompound;

{int} operSingle = asSet(1..nbOpsSingle);
{int} operCompound = asSet(1..nbOpsCompound);

range Jobs = 1..nbJobs;
range Mchs = 1..nbMchs;
//range OpersCompound = 1..nbOpsCompound;
range OpersSingle = 1..nbOpsSingle;

int Priorities[j in Jobs] = ...;

tuple OperationSingle {
  int id;     // Operation id
  int jobId; // Job id
  int pos;    // Position in job
  int mch;  // Machine
  int pt; // Processing time
};
tuple OperationCompound {
  int id;
  int job;
  int oper1Id;     // Robot
  int oper2Id; // another stage
  int mch2;  // Machine
};
{OperationSingle} OpsSingle  = ...;
{OperationCompound} OpsCompound = ...;

// Position of last operation of job j
int jlast[j in Jobs] = max(o in OpsSingle: o.jobId==j) o.pos;
int compoundlast[j in Jobs] = max(o in OpsCompound: o.job==j) o.oper2Id
    ;

dvar int+ startTimes  [op in OpsSingle];
dvar int+ y[operSingle][operSingle] in 0..1;
dvar int+ x[operCompound][operCompound] in 0..1;

dexpr int all_ops = sum(j in Jobs, o in OpsSingle: o.jobId == j)
   startTimes[o] * Priorities[j];
dexpr int end_ops = sum(j in Jobs, o in OpsSingle: o.pos==jlast[j] && o
    .jobId == j) startTimes[o] * Priorities[j];
```

```
minimize all_ops;//end_ops;//
subject to {
EndBeforeStart:
  forall (o1 in OpsSingle, o2 in OpsSingle: o1.jobId==o2.jobId && o2.
     pos==1+o1.pos) //, o in OpsCompound && o1.id != o.oper1Id)
      startTimes[o2] - startTimes[o1]  >= o1.pt ;
NoOverlap:
  forall (m in Mchs, o1 in OpsSingle, o2 in OpsSingle: o1.mch==m && o2.
     mch==m && o1 != o2)
      {
      startTimes[o1] >= startTimes[o2] + o2.pt - y[o1.id][o2.id]*L;
      startTimes[o2] >= startTimes[o1] + o1.pt - (1 -y[o1.id][o2.id])*L
          ;
      }
compoundOpers:
  forall (o1 in OpsSingle, o2 in OpsSingle, o in OpsCompound: o1.id==o.
     oper1Id && o2.id==o.oper2Id)
      startTimes[o2] - startTimes[o1]  == o1.pt;
limitMachine:
     forall (o1a in OpsCompound, o2a in OpsCompound,
   o1 in OpsSingle, o2 in OpsSingle, o3 in OpsSingle, o4 in OpsSingle,
      o5 in OpsSingle, o6 in OpsSingle:
   o1.id == o1a.oper2Id && o2.id == o2a.oper1Id && o3.id == o1a.oper2Id
      +1 &&
   o4.id == o2a.oper2Id && o5.id == o1a.oper1Id && o6.id == o2a.oper2Id
      +1 &&
   o1a.job != o2a.job && o1a.job == o3.jobId && o2a.job == o6.jobId &&
   o1a.mch2 == o2a.mch2 && o1.jobId == o3.jobId && o2.jobId == o6.jobId
      )
   {
    startTimes[o2] - startTimes[o1] - o1.pt  >= (x[o1a.id][o2a.id] - 1)
       *L ;
    startTimes[o2] - startTimes[o3] - o3.pt  >= (x[o1a.id][o2a.id] - 1)
       *L ;
    startTimes[o4] + o4.pt - startTimes[o5]  <= x[o1a.id][o2a.id] *L ;

    startTimes[o6] + o6.pt - startTimes[o5]  <= x[o1a.id][o2a.id] *L ;
   }
}
main {
  thisOplModel.generate();
  //cplex.setParam(TiLim,120);
  cplex.solve();
  cplex.getSolnPoolNsolns();
  var ofile = new IloOplOutputFile("modelRun.txt");
  for(var i in thisOplModel.OpsSingle)
  {
```

```
    ofile.writeln(thisOplModel.startTimes[i]+",");     // write
        startTimes to file
  }
  ofile.close();
  writeln("Number of integer variables " + cplex.getNintVars());
  writeln("Number of bin variables " + cplex.getNbinVars());
  writeln("Number of iterations " + cplex.getNiterations());
  writeln("Number of constraints " + cplex.getNrows());
  writeln("Number of nodes " + cplex.getNnodes());
  cplex.exportModel("model.lp");
}
```

A.3 Formát datových souborů pro IBM ILOG studio

Následuje popis formátu datového souboru `.dat` pro IBM ILOG CPLEX Optimalizační studio. Ten je zdrojem vstupních informací jak pro CPLEX model, tak pro model CP.
Datový soubor obsahuje 4 struktury. Sadu parametrů *Params*, pole priorit (*Priorities*) pro každý nápoj, strukturu operací všech nápojů a strukturu složených operací. Pro vytvoření datového souboru stačí vytvořit strukturu `OpsSingle`, tu dát jako vstupní soubor Ukázkovému programu (kapitola 4.7), ve kterém se vytvoří export do IBM ILOG studia.

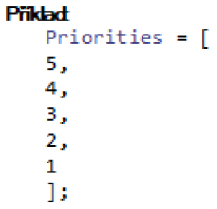
Parametry jsou počet nápojů (`mn_ napoj`), počet stanovišť `poc_ st` (obvykle 6), celkový počet operací `poc_ oper` a počet složených operací `poc_ operComp` . Struktura se v datovém souboru nazývá *Params* a příklad je na Obr.A.1.

```
Format:

<mn_napoj, poc_st, poc_oper, poc_operComp>

Příklad:
    Params = <5, 6, 35, 15>;
```

Obr. A.1: Sada parametrů pro model.

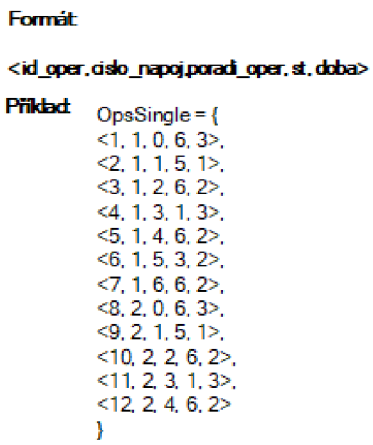
Priority jsou definovány polem, obvykle sestupně po sobě jdoucích hodnot (1. nápoj má největší prioritu). Pole je nazvané *Priorities* a příklad je na Obr.A.2.

Sada operací jako struktura nazvaná *OpsSingle* je složena z 5 hodnot. Identifikátoru operace `id_ oper` (označení pořadí v rámci všech operací), čísla nápoje `cislo_ napoj`, pořadí operace počítané od 0 (`poradi_ oper`), stanoviště `st` a doba trvání



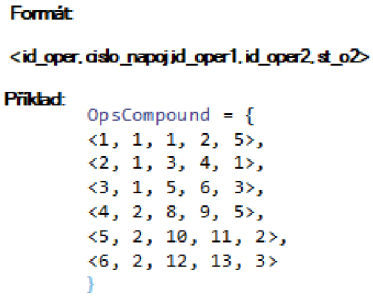
Obr. A.2: Ukázka příkladu priorit.

operace `doba`. Jedná se také o strukturu sloužící jako vstupní data pro Ukázkový příklad. Formát a příklad viz Obr.A.3.



Obr. A.3: Sada operací.

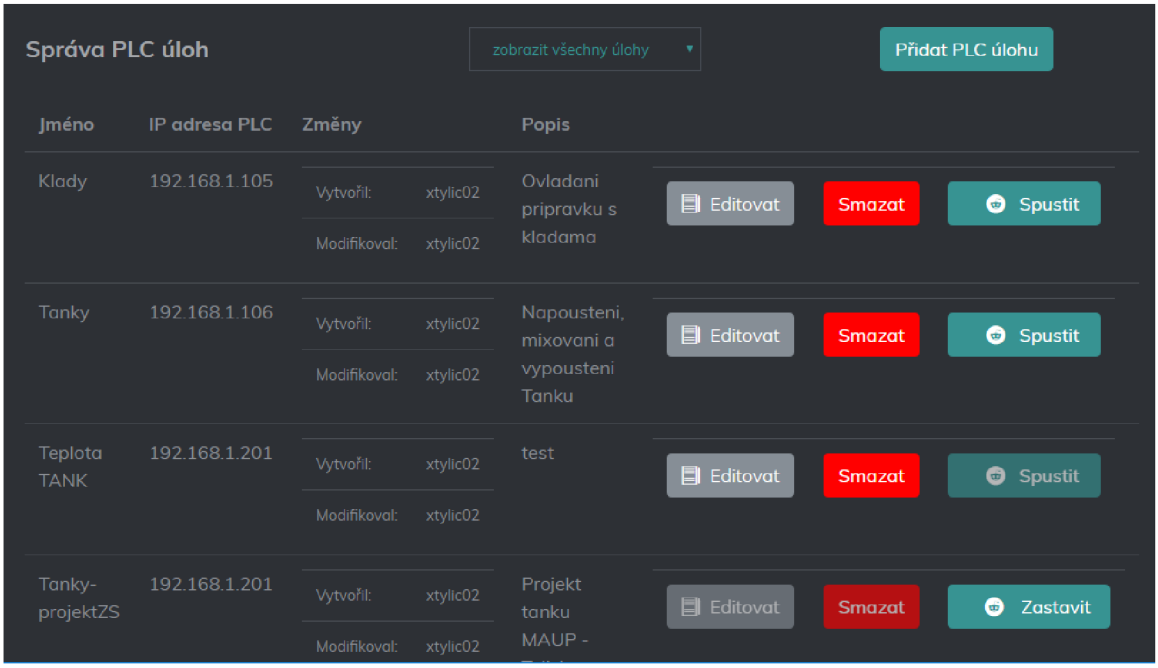
Sada složených operací jako struktura nazvaná *OpsCompound* je složena také z 5 hodnot. Identifikátoru operace `id_ oper` (označení pořadí v rámci všech operací), čísla nápoje `cislo_ napoj`, identifikátor operace robota (`id_ oper1`), identifikátor stanoviště, kam je sklenice přesouvána `id_ oper2` a číslo jejího stanoviště `st_ o2`. Formát a příklad viz Obr.A.4.



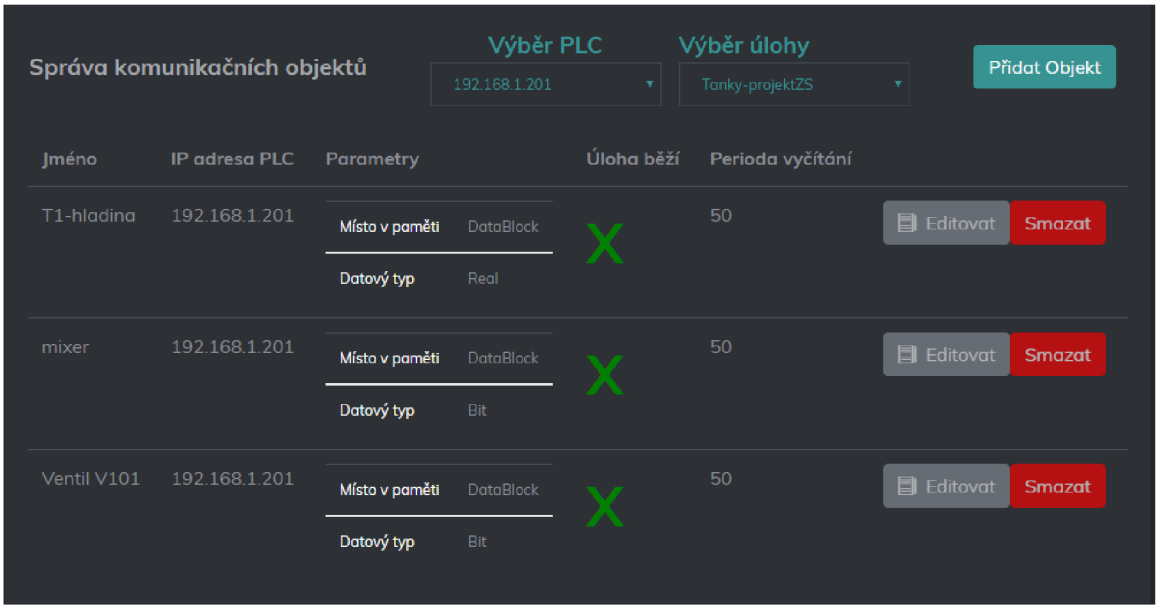
Obr. A.4: Sada operací.

B Obrazovky klientské aplikace systému MES

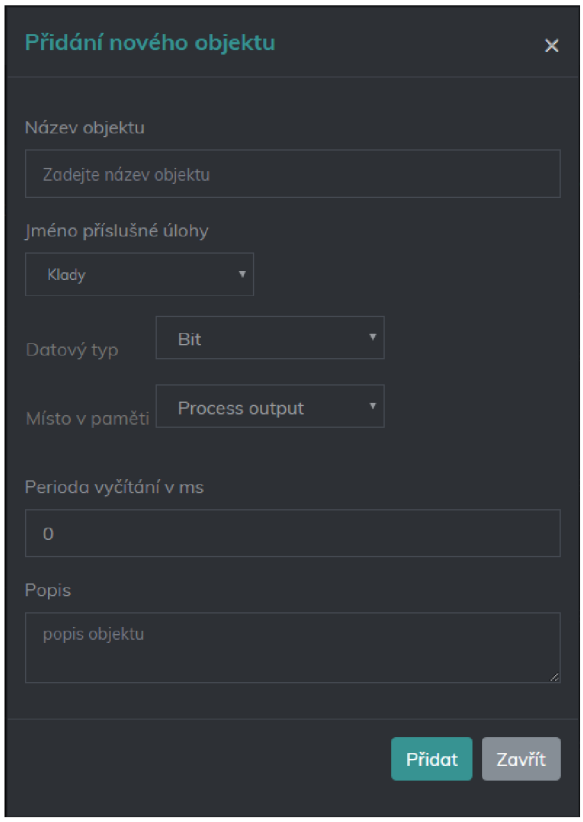
V této části jsou některé obrazovky systému MES.



Obr. B.1: Obrazovka PLC úloh.



Obr. B.2: Obrazovka komunikačních objektů patřících k jedné, spuštěné úloze.



Obr. B.3: Obrazovka komunikačních objektů patřících k jedné, spuštěné úloze.

Obr. B.4: Obrazovka zobrazení historických dat.

C Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `CPLEX-modely-bez-optimalizace` ..... model výroby nápojů jeden za druhým
  - `CPLEX-modely-porovnáni` ..... modely IBM ILOG použité pro simulace
  - `CPLEX-modely-porovnáni1` ..... modely s přehozenými operacemi
  - `LPSolve-porovnani` ..... modely LPSolve použité pro simulace
  - `MES` ..... projekty MES systému/demo programu pro PLC/export DB
  - `Projekty-IbmIlog` ..... kompletní složky projektů pro IBM ILOG
  - `UkazkovaAplikace` ..... zdrojové soubory k ukázkové aplikaci
  - `UkazkovaAplikace-Release` ..... soubory pro spuštění aplikace na PC bez MSVS
  - `Prace` ..... dokument a zdrojové soubory této práce