VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

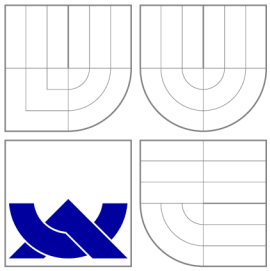
DISTRIBUOVANÝ SYSTÉM PRO ŘÍZENÍ A VIZUALIZACI PRŮMYSLOVÉHO PROCESU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

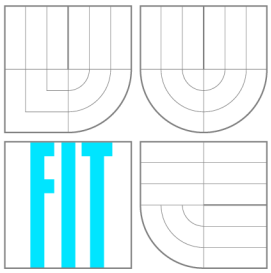
AUTOR PRÁCE
AUTHOR

MARTIN KLVAŇA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DISTRIBUOVANÝ SYSTÉM PRO ŘÍZENÍ A VIZUALIZACI PRŮMYSLOVÉHO PROCESU

DISTRIBUTED SYSTEM FOR CONTROL AND VISUALIZATION OF INDUSTRIAL PROCESS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KLVAŇA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2013

Abstrakt

Práce popisuje vývoj systému vizualizace a řízení výrobního procesu pro firmu vyrábějící křemíkové desky. Hlavním přínosem systému je zvýšení ergonomie práce operátorů výrobní linky pomocí navrženého uživatelského rozhraní. Proto jsou v práci i probrány zásady správného návrhu uživatelského rozhraní. Aplikace se snaží využít všech výhod nově navrženého uživatelského rozhraní ModernUI, které je zde použito. Práce popisuje návrh architektury spolu s implementací, hodnotí zpětnou vazbu od operátorů firmy a shrnuje nasazování aplikace a její budoucí rozšiřování.

Abstract

The thesis describes the development of system for control and visualization of industrial process created for a company producing silicon wafers. The main benefit of this system is improving operator working ergonomy with the designed user interface. This is why the thesis also discuss good principles of user interface design. The application tries to take full advantage of ModernUI user interface, which is used in this application. The thesis describes designing architecture and implementation, rates operator feedback and summarize application deploy and her future improving.

Klíčová slova

Uživatelská rozhraní, výroba křemíkových waferů, Modern UI (Metro), MVVM, Silverlight, C#, .NET, Databáze

Keywords

User interface, silicon wafer production, Modern UI (Metro), MVVM, Silverlight, C#, .NET, Databases

Citace

Martin Klvaňa: Distribuovaný systém pro řízení a vizualizaci průmyslového procesu, bakalářská práce, Brno, FIT VUT v Brně, 2013

Distribuovaný systém pro řízení a vizualizaci průmyslového procesu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Klvaňa
13. května 2013

Poděkování

Velice rád bych poděkoval vedoucímu mé bakalářské práce Doc. Ing. Adamu Heroutovi, Ph.D. za věcné rady a připomínky, které mi pomohly s vypracováním mé práce. Dále bych rád poděkoval konzultantovi z firmy ON Semiconductor Ing. Petru Barošovi za konstruktivní kritiku, která také ovlivnila finální podobu práce. Na místě je také poděkovat firmě ON Semiconductor, která mi zadala tuto práci k vypracování, stejně tak jako všem jejím zaměstnancům, kteří se podíleli na testování a hodnocení výsledného řešení, nebo poskytovali další věcné rady a doporučení.

Obsah

Kapitola 1

Úvod

Většina lidí si dnes nedokáže představit svůj život bez chytrých přístrojů usnadňujících práci v jakékoli myslitelné činnosti. Pračky, ledničky, auta, počítače, mobily. To je jen zlomek z rozsáhlé množiny zařízení, kde se každý prvek liší, jak funkcemi, tak i stavbou. Tyto prvky však spojuje to, že jsou řízeny polovodičovými čipy. Čipy bývají obvykle vyráběny z křemíkových desek, kde každá je vyráběna složitým technologickým procesem, hierarchicky rozděleným do více částí. Celý proces není plně automatizovaný a stále vyžaduje spoluúčast lidských operátorů. Právě lidský faktor tvoří nejkritičtější část celého výrobního procesu, protože je nejvíce náchylný k chybám.

Cílem této práce je zvýšit ergonomii práce operátora na jedné konkrétní části výrobního procesu. Za tímto účelem vznikla aplikace, která bude nejen vizualizovat postup operátorovy práce, ale případně i kontrolovat správné pořadí vykonávaných úkonů. Díky tomu se předpokládá i pokles jím produkovaných chyb a s tím související nárůst produktivity. Aplikace bude tvořena tenkým klientem implementovaným v technologii `Silverlight` a serverem, zajišťujícím veškerou obsluhu a implementovaným v technologii `.NET`.

Jelikož aplikace je navržena podle specifických požadavků firmy zabývající se výrobou křemíkových desek a v této firmě žádná podobná aplikace neexistuje, je možné tuto práci považovat za průkopnickou. Ergonomii uživatele při práci na počítači nejlépe zvýšíme intuitivním a přehledným uživatelským rozhraním zaměřeným na obsah. Tyto požadavky splňuje Microsoftem navržené uživatelské rozhraní `ModernUI`[1], do kterého bude celá aplikace stylizována.

Obecný postup výroby křemíkových desek ve firmě ON Semiconductor a popis výrobní části s analýzou práce operátora, kde bude nasazena aplikace, popisuji v kapitole 2. V kapitole 3 jsou popsány zásady pro správnou tvorbu uživatelských rozhraní a specifika uživatelského rozhraní `ModernUI`. Dále se v kapitole 4 zabývám návrhem aplikace, kde je postupně zmíněn návrh serverové části, schéma databáze a myšlenky při návrhu uživatelského rozhraní. Vlastní implementaci, kde je popsán použitý návrhový vzor `MVVM`, propojení serveru s klientem a implementace serveru a klienta, jsem popsal v kapitole 5. Kapitola 6 zmiňuje použití a testování aplikace operátory. A nakonec v závěrečné kapitole 7 popisuji výsledky celé práce a její zhodnocení.

[1] Popisem se zabývám v podkapitole 3.3

Kapitola 2

Výroba křemíkových desek v ON Semiconductor

Informace v této kapitole jsou čerpány z dokumentu Technologie růstu monokrystalů křemíku czochralskiho metodou dostupného ve sborníku Škola růstu monokrystalů [3]. Doplňující informace a obrázky z výroby jsou čerpány z interních zdrojů nebo ze stránek ON Semiconductor [8].

ON Semiconductor je nadnárodní korporace zabývající se vývojem a výrobou, jak různých integrovaných obvodů, tak křemíkových desek. Křemíkové desky tvoří základ pro výrobu integrovaných obvodů. Proces výroby křemíkových desek zahrnuje jak chemické tak i strojírenské operace. Při každé operaci je požadována vysoká přesnost a extrémní čistota celého procesu. V ON Semiconductor jsou výráběny křemíkové leštěné a epitaxní desky o průměrech 100 a 150mm. Tyto desky jsou vyráběné z Czochralského metodou připraveného monokrystalu křemíku.

Před samotnou výrobou monokrystalu křemíku je třeba získat polykrystal křemíku, který je vstupním prvkem v Czochralského metodě. Ten se získává z dokonale čistého křemenného písku. Společnosti se neoplatí vlastními prostředky vyrábět polykrystal křemíku, proto je nakupován od externích dodavatelů.

2.0.1 Růst monokrystalu

Počáteční operací je až růst monokrystalu. V této fázi je vstupní polykrystalický křemík nejprve smíchán s malým množstvím příměsi(dopant). Tento dopant určuje elektrické vlastnosti výsledného produktu. Podle množství a druhu dopantu získáme monokrystal s nevlastní vodivostí typu N nebo P. Následně je takto smíchaný polykrystal křemíku roztaven v křemenném kelímku a do taveniny je ponořen monokrystalický zárodek.

Regulováním rychlosti tažení, teploty taveniny, otáček a řady dalších technologických parametrů se docílí toho, že atomy křemíku se postupně zabudovávají do přesně definovaných poloh v krystalové mřížce a postupně tvoří monokrystal. Po vytažení těla monokrystalu následuje tvorba špice krystalu pomocí zvyšování rychlosti tažení, čímž klesá průměr monokrystalu. Zúžení zamezuje vzniku poruch krystalové mříže. K poruchám může dojít vlivem teplotního šoku po vytažení krystalu z taveniny. I proto se už hotový monokrystal křemíku chladí jen velmi pomalu.

2.0.2 Řezání monokrystalu

Po vychladnutí se z vytaženého krystalu odřeže nepotřebná hlava a špice. Válcové tělo se rozčlení na více částí pro snadnější manipulaci a z každého provedeného řezu se odebere kontrolní deska o tloušťce 1 mm. Na těchto deskách se testují základní fyzikální vlastnosti jako měrný odpor nebo koncentrace žádoucích a nežádoucích příměsí. Části krystalu splňující specifikaci se obrousí na požadovaný průměr.

Následující operací je řezání monokrystalu, při níž se krystal rozčlení na jednotlivé desky. K tomu se používá technologie řezání na pilách s vnitřním diamantovým bortem nebo řezání na tzv. drátových řezačkách, kde se zpracovává celý díl krystalu najednou. Aby se zabránilo vzniku výštipků a lomů při následných operacích, dochází k zaoblování ostrých hran desek po nařezání. Profilování okraje se děje pomocí diamantového kotoučku s drážkou požadovaného tvaru.

2.0.3 Lapování a leptání

Při řezání desek obvykle dochází k narušením povrchu. Většina těchto narušení bývá odstraňována pomocí oboustranného lapování desek, kdy jsou všechny desky ponořeny do jemné suspenze kysličníku hlinitého. Kromě toho dojde i ke zlepšení geometrických parametrů. Následné leptání ve směsi kyselin odstraňuje zbytkové narušení po lapování.

2.0.4 Leštění

Pro další vyhlazení povrchu desky dochází ještě k leštění. Leštění je kombinací chemických a mechanických procesů, které dávají přední straně desek zrcadlově lesklý povrch v kvalitě potřebné pro současnou výrobu polovodičových součástek. Po vyleštění desky procházejí v několika krocích sérií chemických a mechanických čistících operací. Zde se odstraňují poslední zbytky leštící suspenze, stopové kontaminace i prachové částice. V této části procesu bývá zapojen i přístroj `DewaxSpec`.

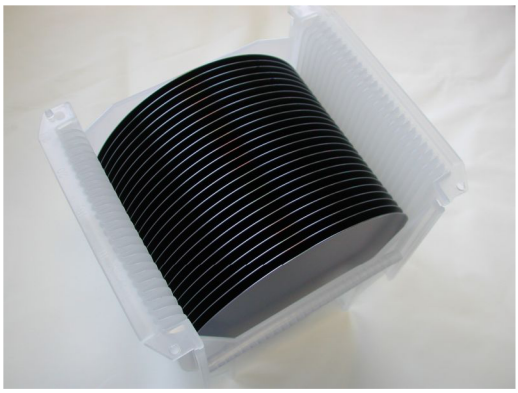
2.0.5 Závěrečná kontrola

Připravené desky nyní procházejí poslední kontrolou. Měření probíhají bezkontaktně a měří se nejen elektrické parametry desek(opět měrný odpor), ale i mechanické vlastnosti jako je zakřivení desky nebo tloušťka a její proměnnost. Po posledním čištění a vizuální kontrole je část leštěných desek zabalena a distribuována přímo zákazníkům, část pokračuje na operaci epitaxe.

2.1 Přístroj DewaxSpec

Při leštění křemíkových desek se nad jednotlivými deskami přilepenými k podložce pohybuje leštící kotouč. Desky jsou ke spodní podložce přilepovány vrstvou vosku, aby při pohybu kotouče nedocházelo k jejich odlepování. Protože je po leštění nežádoucí, aby na zadních stranách desek zůstalo jakékoli množství vosku či jiných nečistot, dochází postupně k několika chemickým čištěním. Všechna tato čištění probíhají v přístroji zvaném `DewaxSpec`.

`DewaxSpec` je zařízení několik metrů dlouhé, zasahující do dvou výrobních prostor a procházející zdí. Nachází se totiž na pomezí čistého a extrémně čistého prostoru, proto je nutné, aby jej obsluhovali alespoň 2 operátoři. Vstup `DewaxSpecu` se nachází v čistém prostoru. Jelikož zde desky ještě obsahují pozůstatky vosku a jiných nečistot, není potřeba



Obrázek 2.1: Desky v teflonové kazetě, ve které jsou nakládány do DewaxSpecu [8].

dodržovat maximálně přísné požadavky na čistotu. Výstup `DewaxSpecu` už ovšem zasahuje do extrémně čistého prostoru, kde platí nejpřísnější normy na čistotu celého prostoru. Po průchodu `DewaxSpecem` už desky čeká pouze měření fyzikálních vlastností a poslední mytí před finálním odesláním k zákazníkům.

Jednotlivé desky jsou při průchodu `DewaxSpecem` čištěny v několika fázích, během kterých se na křemíkové desky působí zvukem o frekvenci okolo 1 MHz. Tento krok pomáhá lépe uvolňovat usazeniny. V první fázi jsou odstraňovány voskové a jiné organické zbytky pomocí směsí kyseliny sírové a peroxidu vodíku. Následuje leptání kyselinou fluorovodíkovou, které z povrchu odstraní usazené oxidy a zbytky leštící vodní emulze. Ve třetí fázi dochází k odstraňování prachových částic pomocí čištění v roztoku hydroxidu amonného a peroxidu vodíku. Tento roztok prachové částice nejenom odnese, ale také zabrání jejich opětovnému usazování. V předposlední části na desky působí směs kyseliny chlorovodíkové s peroxidem vodíku, aby z povrchu desky odstranila usazené částice kovů. Nakonec se vyčištěná deska musí nechat několik desítek minut odstát než je považována za zpracovanou a je možno ji vybrat z `DewaxSpecu`.

Všechny fáze trvají dohromady něco málo přes hodinu. Nejdelší část zabere 28 minut. Pokud se čištěná várka přesune do další fáze, je možno naložit další várku, což vede ke zrychlení zpracování podobně jako při pipeliningu u procesorů. Desky mohou být v přístroji čištěny dvěma různými programy. Jednotlivé programy se v zásadě liší pouze časem jednotlivých kroků nebo množstvím působících směsí. Všechny kazety ve várce však musí být určeny pro stejný program zpracování, jinak by mohlo dojít nejen k poškození desek, ale i přístroje.

2.2 Analýza práce operátora

Protože určité části výroby křemíkových desek nelze plně automatizovat, jsou lidští operátoři také významným článkem celého výrobního procesu. Aby byla zachována čistota prostředí, ve kterém dochází při určitých výrobních úkonech k manipulaci s křemíkovými deskami, musí mít každý operátor oblečen ochranný oblek a latexové rukavice. Obvyklá pracovní doba operátora činí 12 hodin. Operátor pracující v čistých prostorách má tedy složitější podmínky než člověk pracující v kanceláři.

Úloha operátora jakéhokoli zařízení na celé výrobní lince spočívá v udržování vytíženosti přístroje na maximum a vyvarování se jakýchkoli chyb. Proto výroba obsahuje plejádu aplikací pro maximální usnadnění práce operátora.

Obrázek 2.2: Operátor v pracovním obleku určeném do čistých prostor [8].

Přístroj `DewaxSpec` je nutné obsluhovat dvěma operátory. První operátor působí u vstupu a druhý u výstupu `DewaxSpecu`. Oba operátoři pracují s umělohmotnými kazetami, kde bývají přenášeny křemíkové desky. Každá kazeta v ideálním případě obsahuje 25 křemíkových desek. Veškeré kazety ve výrobě (umělohmotné a teflonové) obsahují čárový kód, ke kterému se vážou veškeré dostupné informace o deskách v kazetě. Všechny operace s kazetami je nutné zaznamenávat do databází a systému `PROMIS`. Tak se obvykle děje přečtením čárového kódu kazety pomocí čtečky čárových kódů a uložením pomocí speciálního softwaru.

2.2.1 Práce operátora u vstupu

Práce operátora u vstupu spočívá ve vytváření várek z kazet obsahujících desky. Z předcházejícího kroku výroby se operátorovi postupně hromadí kazety. Jedna várka pro čištění může pojmout 3 kazety. Operátor vytváří várky z dostupných kazet a následně várky vkládá do přístroje `DewaxSpec`, kde probíhá čištění. Jelikož do `DewaxSpecu` je možné vkládat křemíkové desky pouze ve speciální teflonové kazetě, musí operátor všechny desky ze vstupní kazety nejprve přesunout do teflonové kazety. Až poté je kazeta umístěna do várky. Pokud je várka vytvořena a sady správně uloženy v `PROMISu`, je možno zahájit čištění.

2.2.2 Práce operátora u výstupu

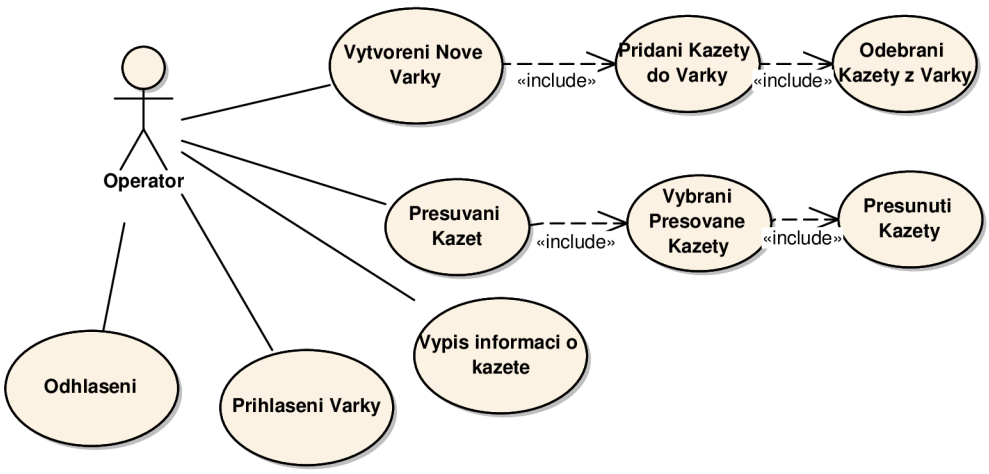
Operátor u výstupu vybírá zpracované várky z přístroje. Při vybírání opět přečte čárový kód. Při této operaci také dochází k vygenerování nového kódu pro další část výroby. Následně přesouvá desky z teflonových kazet do umělohmotných výstupních. Po přesunu nakonec nalepí nově vygenerovaný kód a zároveň jej i přečte, čímž vyvolá v aplikaci dialog pro odhlášení sady z `PROMISu`. Tím je sada zpracovaná a je možné ji přesunout dále.

2.3 Specifikace požadavků na funkčnost aplikace

Výsledkem práce by měl být distribuovaný systém sloužící operátorům obsluhujícím přístroj pro čištění křemíkových desek `DewaxSpec`. Systém by měl být dostupný ze všech počítačů firemního intranetu, proto je nutné rozdělení na serverovou část a tenkou klientskou část s uživatelským rozhraním. Serverová část musí být schopna při jakémkoli výpadku schopna obnovit předcházející stav, proto budou veškeré informace a stavy aplikace uchovávány v databázi. Komunikace s přístrojem `DewaxSpec` bude probíhat prostřednictvím firemního systému PROMIS. Je třeba implementovat rozhraní pro vstup přístroje `DewaxSpec` a rozhraní pro výstup přístroje `DewaxSpec`. Rozhraní budou sdílet některé prvky a budou mít společnou databázi a serverovou část. Rozhraní pro vstup umožní operátorům sestavování várek tímto způsobem:

1. Po spuštění aplikace se načtou informace z databáze a až do přihlášení, bude aplikace pouze v režimu čtení.
2. Pokud se uživatel autentizuje, bude moct vytvořit a začít sestavovat novou várku o maximálně třech kazetách.
3. Při sestavování várky musí být všechny tři kazety určeny pro stejný program zpracování, jinak dojde ke zvýraznění a aplikace nepustí uživatele k dalšímu kroku.
4. Po úspěšném sestavení várky, dojde k jejich přihlášení do `PROMISu`.
5. Následuje přesun várky do přístroje `DewaxSpec` a uložení informací do databáze.
6. Zahájení zpracování várky a návrat do stavu po autentizaci uživatele, pro sestavení další várky.

Use-Case diagram práce operátora DewaxInputSpec

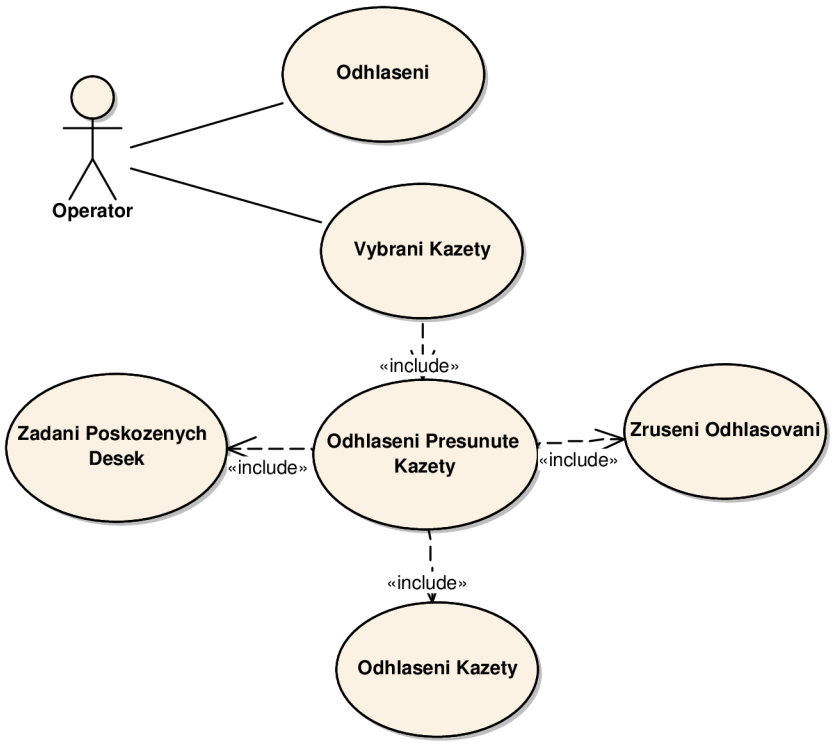


Obrázek 2.3: Diagram případů užití po přihlášení operátora. Bez přihlášení je aplikace v režimu čtení.

Rozhraní pro výstup umožní vybírat již zpracované várky a dále je předávat pro další zpracování:

1. Po spuštění aplikace se načtou informace z databáze a až do přihlášení, bude aplikace pouze v režimu čtení.
2. Pokud se uživatel autentizuje, může vybrat kazetu z várky již zpracované `DewaxSpecem`.
3. Po vybrání kazety dojde k vygenerování a tisku nového čárového kódu. Kód bude nalepen na výstupní kazetu, určenou pro další část výrobního procesu.
4. Dojde k přečtení nově vygenerovaného čárového kódu a aplikace vyvolá dialog, pro odhlášení kazety z `PROMISu`. Zde operátor zadá např. informace o poškozených deskách a odhlásí kazetu s deskami.
5. Po odhlášení je kazeta s deskami připravena pro následující krok výroby a operátor `DewaxSpecu` úspěšně zpracoval celou kazetu.
6. Operátor může pokračovat v odhlašování dalších zpracovaných sad.

Use-Case diagram práce operátora DewaxOutputSpec



Obrázek 2.4: Diagram případů užití po přihlášení operátora. Bez přihlášení je aplikace v režimu čtení.

Dále musí systém umožnit operátorům načítat čárové kódy jednotlivých kazet s deskami prostřednictvím čtečky čárových kódů a také generování nových čárových kódů pro zpracované desky a jejich tisk `ZEBRA`[2] tiskárnou.

[2]Více na http://en.wikipedia.org/wiki/Zebra_Technologies

Kapitola 3

Zásady tvorby uživatelských rozhraní

Uživatelské rozhraní je soubor postupů, jak pracovat s objektem [15]. Krom počítačů jej mají i auta, televize, pračky atd.. Ve výpočetní technice je uživatelské rozhraní obvykle zaměřené na uživatele. Uživatel díky tomu nemusí znát podrobnosti implementace programu a mít odborné znalosti vážící se k jeho obsluze. Naopak je mu předkládána obsluha programu jednoduše a efektivně, tak aby se mohl věnovat skutečnému účelu své práce. Pro předávání informací uživateli jsou využívány vjemy zachytitelné lidskými smyslovými orgány:

- Hlavním smyslovým orgánem je zrak, protože umožňuje předat velké množství informací najednou.
- Doplňkovým je potom sluch.

Naopak uživatel předává počítači a tím i rozhraní podněty pomocí:

- Pohybu, kdy se obvykle využívají ovládací zařízení typu myš, klávesnice nebo joystick.
- Řeči - například příkazy v terminálu.
- Hmatu - různá gesta a mimika například u zařízení `Kinect` u `Xboxu`.

3.1 Doporučené zásady návrhu uživatelských rozhraní

Návrh uživatelského rozhraní tvoří nejdůležitější část práce. Protože samotná příjemnost či efektivita práce s programem závisí výhradně na uživatelském rozhraní [11]. Každé uživatelské rozhraní je možno ohodnotit v pěti kategoriích:

- Dobou nutnou k naučení se práce s rozhraním.
- Rychlostí vykonávání požadované činnosti.
- Druhem a četností chyb, které může uživatel provádět.
- Uchováním znalostí o práci s programem.
- Subjektivním dojmem při práci s programem.

Obvykle se nepodaří uspět ve všech oblastech, protože některé se mohou navzájem vylučovat. Pro co nejlepší výsledek navrhovaného rozhraní je však možno dodržovat osm základních a obecných postupů:

1. **Konzistence:** Dodržování konzistentní terminologie, pravidel pro tvorbu daného prostředí, směřování k tvorbě stereotypů.
2. **Respektování široké skupiny uživatelů:** Musí být jasné, jaké skupiny uživatelů budou aplikaci používat(začátečníci, profesionálové, handicapovaní...) a podle toho tvořit rozhraní. Styly práce u různých skupin se mohou značně lišit, proto musí existovat alternativy v ovládání aplikace.
3. **Zpětná vazba systému:** Uživatele je nutné informovat o výsledcích práce s aplikací(zda akce proběhla úspěšně či nikoliv). Zpětná vazba však nesmí uživatele obtěžovat. Rozlišujeme silnou zpětnou vazbu a slabou zpětnou vazbu.
4. **Navigování uživatele:** Je lepší rozdělit složitý problém na posloupnost menších jednodušších podproblémů(například objednávání zboží v internetovém obchodě se děje v několika krocích namísto jednoho složitého formuláře). Proto by rozhraní akcí mělo být rozděleno na jednoduché, logicky rozčleněné kroky respektující pracovní postup.
5. **Prevence chyb:** Rozhraní by mělo minimalizovat možné chyby uživatele. Pokud chybě nelze předejít, je nutno uživatele o chybě výstižně informovat a přitom také sdělit možné příčiny a možná řešení problému.
6. **Možnost opravy či vrácení akcí:** Je vhodné, aby uživatel mohl všude, kde je to možné, vrátit akci zpět.
7. **Předvídatelné uživatelské rozhraní:** Uživatel musí být řídícím prvkem rozhraní a ne plnit rozkazy aplikace.
8. **Minimalizace nároků na krátkodobou paměť uživatele:** Rozhraní by mělo být přehledné, bez nutnosti si jednotlivé věci pamatovat. Podobné je to u řízení auta, kdy řidiči stačí pouze přehledová informace namísto toho, aby znal přesný obraz jednotlivých zrcátek.

Pokud existuje závažný důvod, je možné jakékoliv z osmi pravidel porušit. Například u specificky zaměřených výrobních aplikací.

3.2 Základní ovládací prvky a jejich uspořádání

Ovládací prvek je základní element uživatelského rozhraní, který slouží pro interakci s uživatelem. Ovládací prvky mohou být rozděleny do několika kategorií [14]:

- **Okna** jsou asi nejdůležitějším komunikačním prvkem uživatelského rozhraní, dělí se na modální a nemodální nebo dialogová.
- **Výběrové a zobrazovací prvky** jsou asi nejrozsáhlejší skupinou ze všech ovládacích prvků. Patří sem například Tlačítka, Seznamy, Ikona, Lišta nabídek nebo Zaškrtávací pole.

- **Navigační prvky** jako Panel nebo Posuvník slouží pro pohyb ve stránce.
- **Textový vstup** většinou slouží pro zadávání textových údajů pro uživatele. Patří sem Textové pole, Adresní řádek a Kombinované pole.
- **Textový výstup** kam patří prvky jako Stavový a Informační řádek, Popisek nebo Indikátor průběhu slouží jako výstup programu pro zobrazování různých informací.

3.2.1 Uspořádání prvků

Abychom vytvořili přehledné, produktivní a pro uživatele příjemné uživatelské prostředí, je třeba dbát na správné uspořádání jednotlivých prvků. Při rozmísťování prvků je nutné brát v úvahu, že uživatelé jsou zvyklí číst zleva-doprava a shora-dolů a že rozmístění a posloupnost ovládacích prvků by mělo respektovat tok informací. Lidé také kladou důraz na uspořádanost, systém a jednoduchost.

Při návrhu uživatelského rozhraní hrají roli také rozměry a vzdálenosti mezi ovládacími prvky. Protože systém může být provozován na různých zařízeních s různými parametry nebo nastaveními, je nevhodné specifikování rozměrů a vzdálenosti pomocí fyzických rozměrů. Typickými vlastnostmi, které se mohou měnit jsou například rozlišení nebo tvar a rozměry systémového fontu. V dnešních systémech je zpravidla využíváno tzv. relativních pixelů, které zajišťují nezávislost na rozlišení daného zařízení.

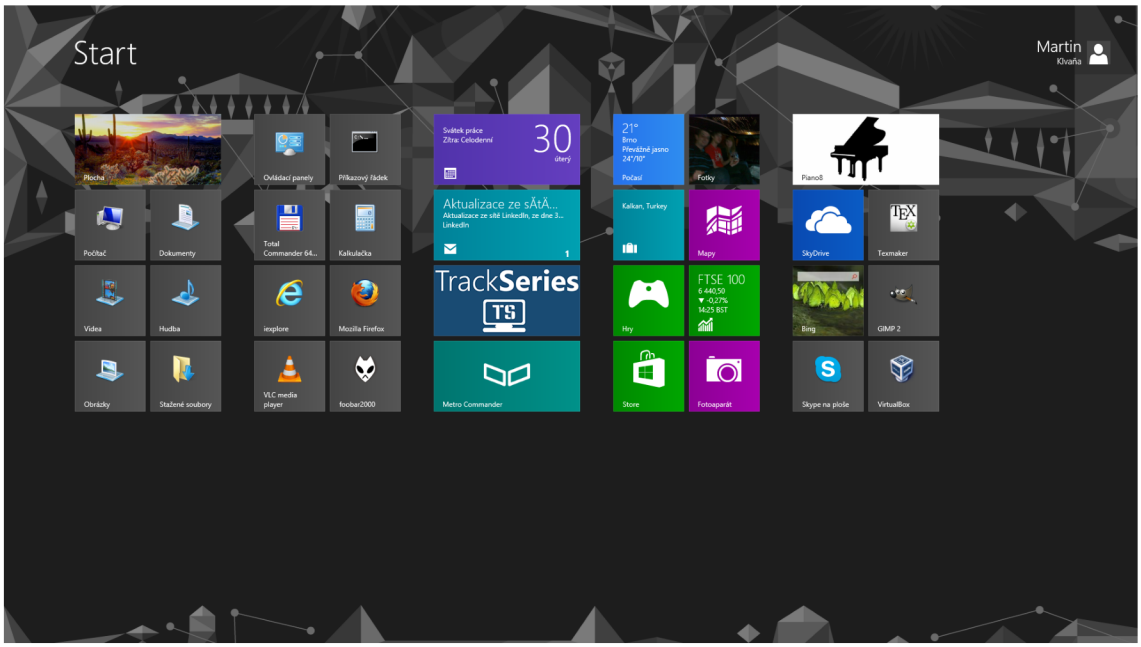
3.3 Principy návrhu uživatelského rozhraní ModernUI (Metro)

`ModernUI` je název pro nové uživatelské rozhraní od Microsoftu kódově nazývané `Metro`. Toto rozhraní je postupně zaváděno do všech Microsoftem vydaných produktů jako `Windows 8`, `Windows Phone 8` a jiných. Kódový název `Metro` nebyl použit jen náhodou. V metru, na letištích a vlakových nádražích našel Microsoft inspiraci při návrhu. Na těchto místech se totiž vykytují přehledné informační tabule pomáhající k rychlejší orientaci.

Základem `ModernUI` jsou dlaždice. Ty se dělí na aplikační a sekundární dlaždici. Aplikační dlaždice tvoří zástupce programu na hlavní obrazovce systému. Sekundární se zobrazují až při spuštění programu. Všechny dlaždice jsou animované.

Pro podporu vývoje nových aplikací s `ModernUI` uživatelským rozhraním Microsoft vydal publikaci, kde jsou shrnuty zásady pro návrh [6]. Hlavními zásadami pro návrh takto stylizovaných aplikací jsou:

- **Ukázat kus poctivé práce:** Používání aplikace by pro uživatele mělo být vždy zážitkem. Proto by se měl návrhář aplikace věnovat každému detailu u všech prvků aplikace. Aplikace by také měla být pro uživatele předvídatelná a přístupná co nejširší možné základně potenciálních uživatelů.
- **Rychlé a plynulé používání:** Uživatelé oceňují přímou a rychlou interakci s obsahem, který si prohlížejí. Proto je důležité, aby reakce aplikace působily živě, mohly se ovládat intuitivními a snadno přístupnými gesty a reakce na následující uživatelskou akci byla vždy ihned připravená.
- **Využít sílu virtuálního světa:** Aplikace by měla využívat veškeré přednosti digitálních médií. Návrhář aplikace by měl odhodit veškerá omezení reálného světa a



Obrázek 3.1: Metro obrazovka ve Windows 8.

vytvářet aplikace, které díky využití moderních technologií budou mnohem efektivnější a jednodušší na používání. Stav aplikace a uživatelská data by měla být uložena v `cloudu`, přesto by aplikace měla být funkční i bez internetu.

- **Méně je někdy více:** Čisté rozhraní, kde bude do popředí vytažen samotný obsah nad různými oku lahodícími prvky, může ve finále vypadat i sloužit uživatelům lépe. Je lepší být vynikající v jedné činnosti, než být průměrný v mnoha oblastech. Prohlížeč obrázků uživatelé zpravidla otevírají proto, aby si prohlédli obrázky, ne proto aby obdivovali ovládací prvky prohlížeče.

- **Pracovat jako jeden:** Aplikace by měla být schopna spolupráce s ostatními aplikacemi, zařízeními a systémem. Například používáním kontraktů při vyhledávání a sdílení. Důležité je i využívání prvků, které lidé už znají jako různá doteková gesta a animace, namísto vymýšlení nových. Chování aplikace by mělo být konsistentní se systémem a uživatelské rozhraní by nemělo obsahovat redudantní prvky.

Kapitola 4

Návrh uživatelského rozhraní a celkové architektury systému

Při návrhu systému bylo potřeba vycházet ze specifikace požadavků firmy, protože aplikace musela být přizpůsobena stávájícímu ekosystému ve firmě. Proto už od počátku došlo k prostudování patřičných detailů o výrobě a prostudování specifických komponent, se kterými bude muset výsledná aplikace spolupracovat. Tyto informace posloužily při návrhu serverové části. Pro návrh uživatelského rozhraní posloužilo analyzování práce operátorů, pro které bude výsledný systém určen. Veškerá fakta potřebná k návrhu systému jsou popsána v kapitole 2.

4.1 Návrh struktury systému

Už ve specifikaci bylo uvedeno, že celý systém bude typu klient-server. Celá výroba ve firmě ON Semiconductor běží vždy online, aby bylo možné komunikovat s informačním systémem PROMIS, databázemi nebo bylo možno dálkově kontrolovat jednotlivá výrobní zařízení. V případě výpadku sítě se obvykle zastaví celá výroba. V druhém podstatném požadavku na aplikaci stálo, aby ji bylo možné spustit z kteréhokoli počítače firemního intranetu. Aby klesly požadavky na hardware, je klientská část aplikace navržena jako tenká a veškerý výkonný kód běží na serveru.

Při návrhu struktury aplikace byl použit návrhový vzor `MVVM` více popsaný v kapitole kapitole 5, který umožnil oddělit vývoj uživatelského rozhraní a serveru.

Moduly serveru

Výkonnou část celého systému provádí server, pro lepší strukturování je server rozdělen do čtyř modulů. První a zřejmě nejdůležitější modul zajišťuje komunikaci mezi klientem a ostatními moduly. Následuje databázový modul, který využívá databázi uloženou na serveru. Třetí modul se stará o komunikaci s informačním systémem PROMIS. Tento modul hlavně využívá metody z již implementované knihovny pro práci s tímto informačním systémem. Poslední pomocný modul má hlavně na starost komunikaci se `ZEBRA` tiskárnou, čtečkou čárových kódů případně jinými hardwarovými zařízeními.

4.2 Návrh databáze

Systém je navržen tak, aby si i v případě vypnutí pamatoval poslední zpracovávaný stav. Proto jsou veškeré informace potřebné k běhu ukládány do databáze běžící na serveru a díky tomu není třeba řešit lokální ukládání dat u jednotlivě spouštěných klientů. Pro vytvoření tříd v aplikaci je vhodné znát alespoň elementární strukturu a vztahy mezi daty. Celé databázové schéma je zobrazeno na obrázku 4.1. Diagram byl vytvořen podle specifikace požadavků a obsahuje pouze tabulky nezbytně nutné pro chod aplikace. Databáze se skládá ze sedmi tabulek:

- **Tabulka PscEquipments**
  `PscEquipments` obsahuje především dva atributy: `EquipmentID` a `Name`. `EquipmentID` je primární klíč tabulky a atribut `Name` obsahuje název výrobního zařízení. Tabulka uchovává veškerá zařízení výrobní linky a v databázi identifikuje kazety `DewaxSpecu`.

- **Tabulka PscEquipWaitingCassettes**
  `PscEquipWaitingCassettes` slouží pro získání kazet čekajících před `DewaxSpecem`. Její primární klíč `CassetteUsageId` identifikuje každou kazetu, cizí klíč `EquipmentId` se odkazuje na ID výrobního zařízení a sloupec `ProgramName` uchovává program, jakým má být deska zpracována.

- **Tabulka PscCassetteUsage**
  `PscCassetteUsage` udržuje používané kazety na zařízení. Umožňuje přístup ke dvěma pomocným tabulkám uchovávajícím historii přesunu kazet. Obsahuje primární klíč `CassetteUsageId`, cizí klíč `CassetteId` identifikující kazetu ve várce a taky sloupec `UsageTime` určující, kdy naposledy byla kazeta zpracována na `DewaxSpecu`. Pokud je `UsageTime` rovno `NULL`, nachází se kazeta před `DewaxSpecem`.

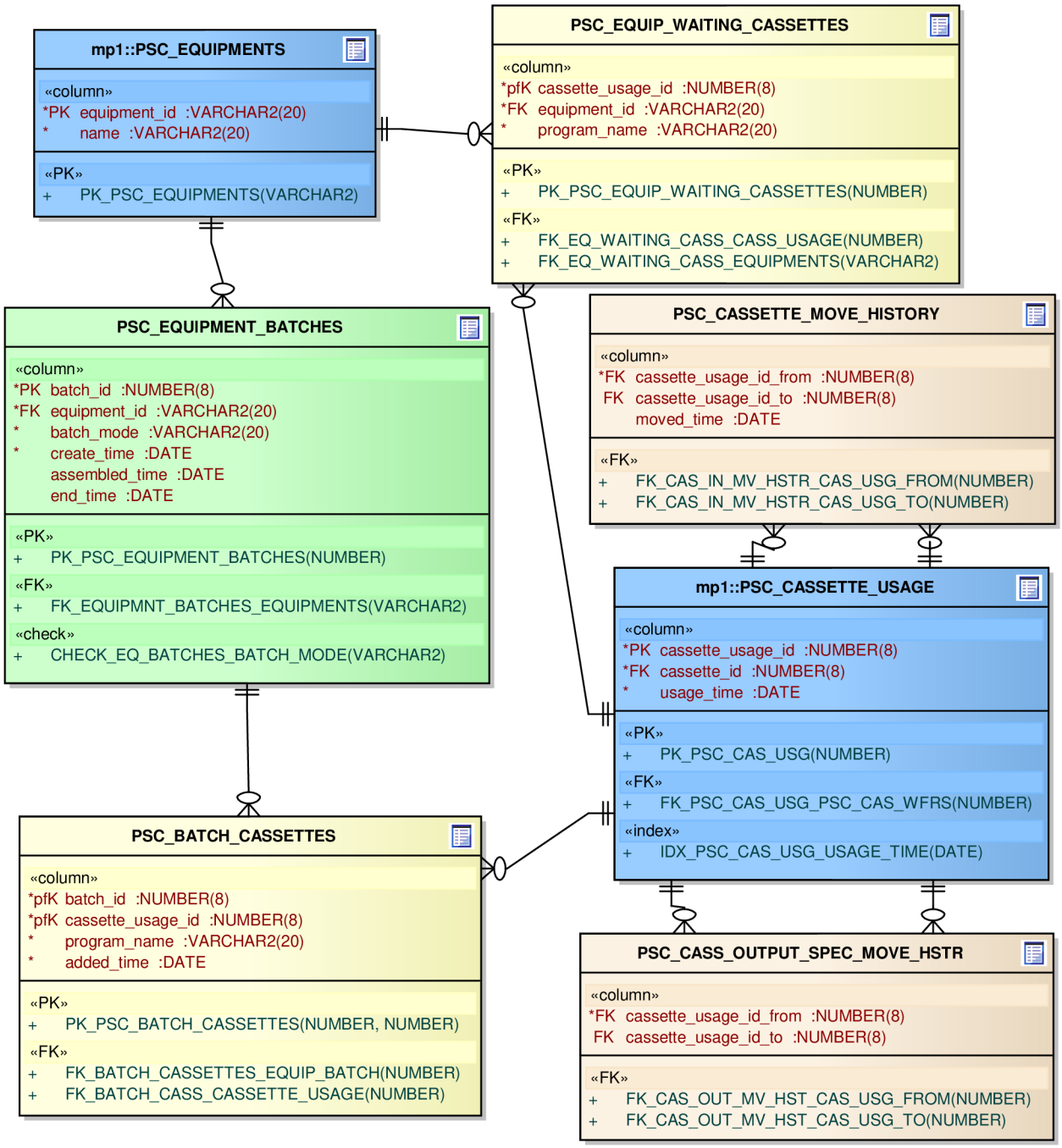
- **Tabulka PscEquipmentBatches**
  `PscEquipmentBatches` uchovává údaje o zpracovávaných várkách. Vždy při vytvoření nové várky je vytvořen nový záznam v této tabulce. Primárním klíčem je sloupec `BatchId` a cizím klíčem je `EquipmentId`. Sloupec `BatchMode` uchovává stav vstupní části `DewaxSpec` aplikace. Ta může být buď ve stavu, kdy se neděje nic, nebo může být sestavována várka a nebo dochází k přesunu kazet. Časové sloupce určují vytvoření várky `CreateTime`, zpracování várky přístrojem `AssembledTime` a kompletní ukončení zpracovávání `EndTime`.

- **Tabulka PscBatchCassettes**
  `PscBatchCassettes` identifikuje jednotlivé kazety ve várce. `BatchId` určuje várku a `CassteUsageId` identifikuje danou kazetu z tabulky `PscCassetteUsage`. Sloupec `ProgramName` uchovává program a `AddedTime` značí čas přidání kazety do várky.

- **Tabulka PscCassetteMoveHistory**
  Pomocná tabulka slouží pouze pro `DewaxInputSpec`, kde uchovává, jaká kazeta byla přesunuta do jaké teflonové kazety. Doplňující je sloupec `MoveTime`, který když je `NULL`, tak kazeta ještě nebyla přesunuta.

- **Tabulka PscCassOutputSpecMoveHstr**
  Pomocná tabulka slouží pouze pro `DewaxOutputSpec` a je analogická s tabulkou `PscCassetteMoveHistory`. Jsou zde pouze zaznamenávány přesuny u výstupní aplikace `DewaxSpecu`.

4.3 Databázové schéma



Obrázek 4.1: Databázové schéma používaných tabulek. Schéma využívá Information Engineering(Crow's Foot) notaci popsanou v knize [2].

4.4 Návrh uživatelského rozhraní

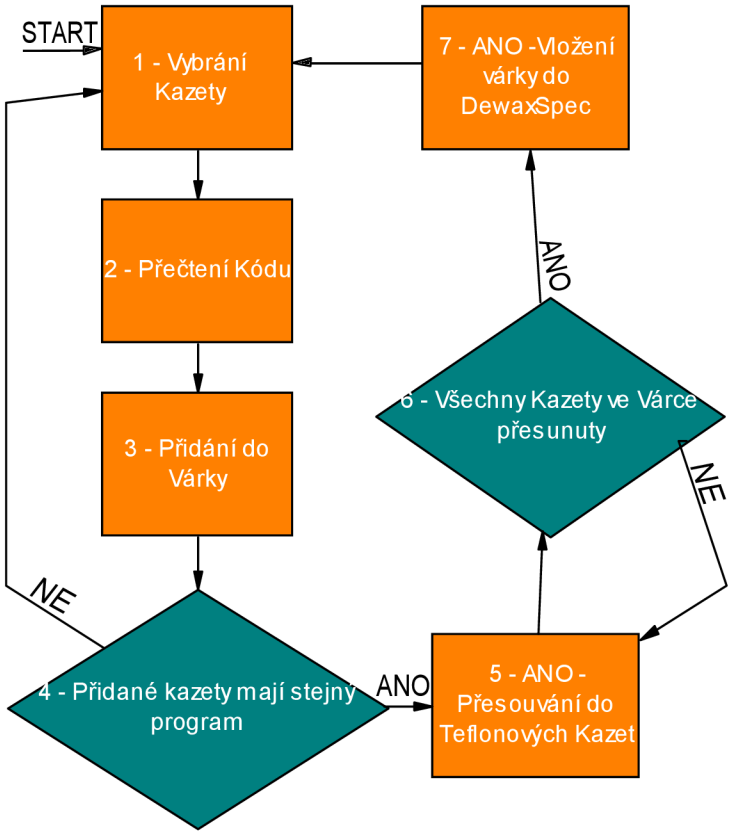
Podkapitola se zabývá návrhem dvou uživatelských rozhraní pro vstup a výstup aplikace pro vizualizaci řízení přístroje `DewaxSpec`. Obě uživatelská rozhraní budou sloužit hlavně operátorům u přístroje. Rozhraní by měla být navržena tak, aby nejen přehledně vizualizovala průběh operátorových kroků, ale i zabránila případným chybným krokům. Zároveň by měly být dodrženy všechny požadavky na aplikaci zmíněné ve specifikaci.

Při návrhu aplikace jsem se snažil splnit nejen praktické požadavky ze strany firmy, ale zároveň se i řídit doporučenými zásadami pro tvorbu uživatelských rozhraní popsanými v kapitole 3. Tento můj dílčí cíl nebylo úplně jednoduché splnit.

4.5 Uživatelské rozhraní pro vstupní část

4.5.1 Myšlenky při tvorbě rozhraní

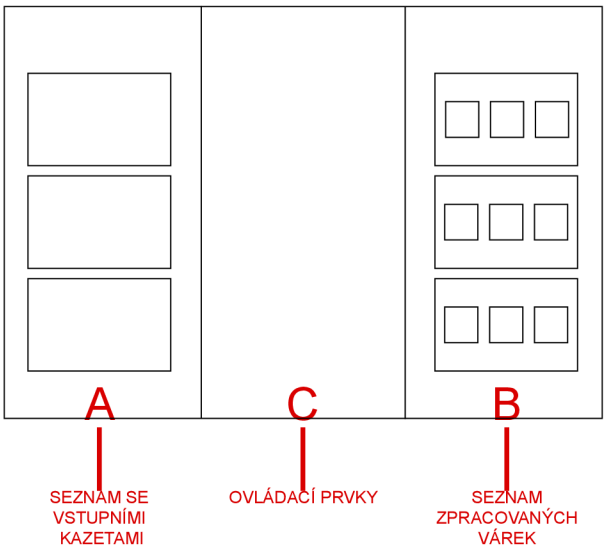
Při navrhování rozhraní bylo třeba se hlavně řídit pracovními úkony operátora a jeho potřebami. Operátor zjednodušeně bere nashromážděné desky, přesune je do nových a ty poté vkládá do přístroje.



Obrázek 4.2: Ilustrační obrázek jednoho cyklu pracovního postupu operátora aplikace DewaxInputSpec.

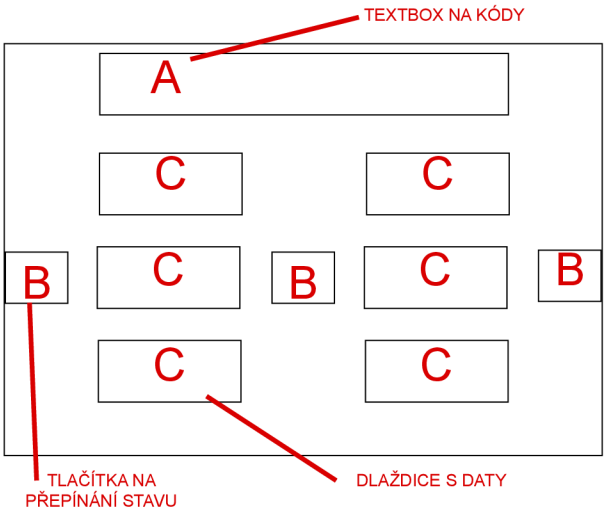
Z tohoto postupu vyplývá jeden zásadní požadavek, že operátor hlavně potřebuje mít přehled v množině příchozích a následně i zpracovávaných desek. Proto jsem se rozhodl,

že po obou stranách aplikace budou dva informační seznamy. Nalevo budou zobrazeny podrobné informace o kazetách. A napravo budou už zpracovávané várky o maximu třech kazet. Oba seznamy budou barevně zvýrazňovat zpracovávaný program.



Obrázek 4.3: Návrh obrazovky se seznamy bez ostatních ovládacích prvků.

Mezi oběma seznamy budou umístěny ovládací prvky aplikace, kde bude vizualizováno vytváření várky a přesouvání jednotlivých kazet(mohou být přesunuty až tři kazety). Protože aplikace bude hlavně ovládána čtením kódů kazet, není třeba vymýšlet žádné speciální ovládací prvky. Pouze bude třeba vizualizovat přesun desek a přehledně informovat v jakém stavu se aplikace nachází.



Obrázek 4.4: Návrh prostřední části obrazovky.

4.5.2 Elementy a jejich použití

Rozhraní je rozděleno do tří hlavních částí. Ve dvou částech nacházejících se na okrajích obrazovky jsou umístěny informační seznamy o várkách respektive o kazetách. Tyto seznamy mají pouze informativní a vizualizační funkci, proto jsou jakékoliv ovládací prvky u těchto seznamů nežádoucí. Pokud by například v seznamu bylo možné kliknutím nebo pomocí `DragNDrop` přesouvat jednotlivé kazety, mohlo by dojít k závažné chybě v pracovním procesu. Operátor by mohl fyzicky nechat zpracovat jednu kazetu a přitom v uživatelském rozhraní do databází a `PROMISu` uložit nezávisle úplně jinou. Proto byly tyto jinak obvyklé funkce účelově zavrženy.

Prostřední část nebude pouze informovat a vizualizovat, ale také bude obsahovat i ovládací prvky. Jelikož celá aplikace bude ovládána hlavně prostřednictvím čárových kódů, bude dominantním ovládacím prvkem Textové pole. Do Textového pole budou v ideálním případě čtečkou čárových kódů vepisovány čárové kódy. V nestandardních situacích, při nedostupnosti čtečky bude moct operátor vložit kód klávesnicí.

Druhým ovládacím prvkem budou tři tlačítka, která budou sloužit pro přepínaní aplikace mezi třemi stavy, kdy aplikace bude v návaznosti na aktuální stav reagovat na aktuálně zadaný kód. Tlačítka budou podle stavu aktivována či nikoliv.

Třetím prvkem v prostřední části bude šest `ModernUI` dlaždic. Ty budou umístěny po třech ve dvou různých sloupcích a budou vizualizovat přesouvání kazet nebo sestavování várek. Každý ze dvou sloupců bude značit, zda je aplikace ve stavu sestavování várky nebo přesouvání kazet.

Nakonec v místě pod Textovým polem budou vypisovány případné chybové a informační pokyny aplikace.

4.6 Uživatelské rozhraní pro výstupní část

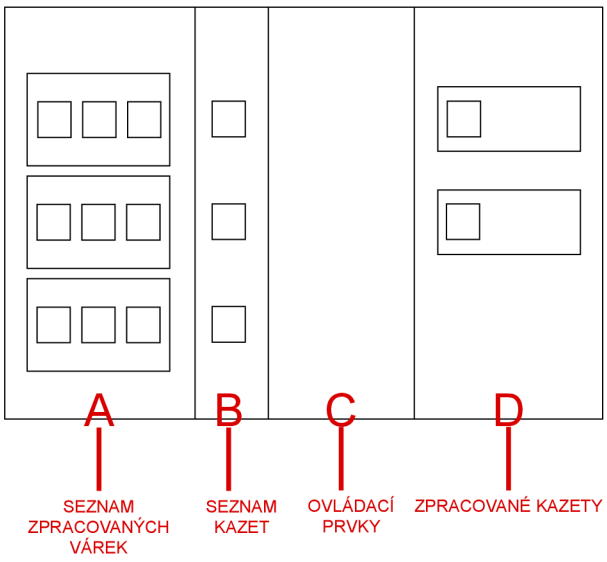
4.6.1 Myšlenky při tvorbě rozhraní

U výstupu je operátorova práce jakoby zrcadlově otočená ke vstupu. Operátor z přístroje bere zpracované várky. Z várek vybírá jednotlivé kazety a z kazet vytváří množinu zpracovaných kazet, kde už dále přechází odpovědnost za kazety na další úsek celé výrobní linky.

Obrázek 4.5: Ilustrační obrázek jednoho cyklu pracovního postupu operátora DewaxOutputSpec aplikace.

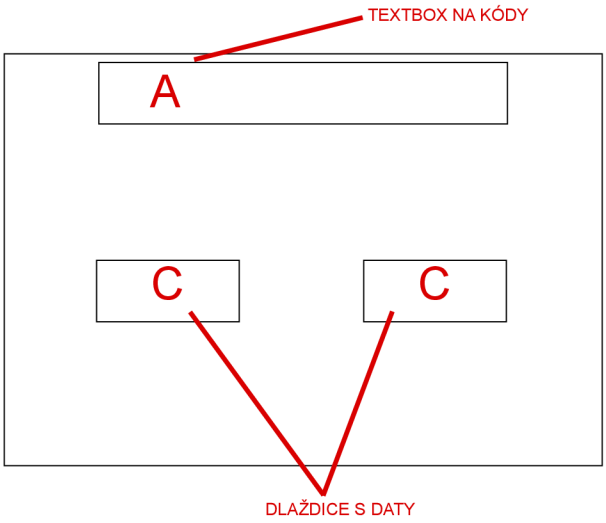
Nyní je proto pravý seznam u vstupu zobrazen nalevo, jelikož jsou vstupními kazetami ty, co vyjedou z přístroje. Napravo potom bude velký seznam, který bude ukazovat už kompletně zpracované a přesunuté kazety. Protože však při vybírání kazet z várek zůstanou kazety, které nepatří do seznamu zpracovávaných ani do seznamu již zpracovaných a přesunutých, bylo nutno přidat ještě třetí seznam, který obsahuje informace o jednotlivých zpracovaných, avšak nevyexpedovaných kazetách. U zpracovaných desek už není nutné

barevně zvýrazňovat program.



Obrázek 4.6: Návrh obrazovky se seznamy bez ostatních ovládacích prvků.

Mezi seznamy bude nyní zjednodušené ovládání ze vstupní aplikace. Zde je potřeba pouze vizualizovat přesun a následné vygenerování nového čárového kódu pouze u jedné kazety místo tří. Bude však třeba dodělat odhlašovací dialog z `PROMISu`, který vyskočí po přečtení nově vygenerovaného čárového kódu.



Obrázek 4.7: Návrh prostřední části obrazovky.

4.6.2 Elementy a jejich použití

Stejně jako rozhraní pro vstupní část bude i výstupní rozděleno do tří částí. V první levé části obrazovky se budou tentokrát nacházet dva seznamy. První bude stejný jako pravý seznam ve vstupním rozhraní. Druhý menší seznam bude obsahovat jednotlivé kazety ze zpracovaných várek. V pravé části obrazovky se bude nacházet nový seznam. Tento seznam bude zobrazovat už kompletně zpracované kazety, které bude shromažďovat k větším celkům, nazývaným `ParentLot`. Jeden `ParentLot` může být složen z více (až padesáti) kazet s deskami. V seznamu budou zobrazovány `Expandery`, barevně odlišující již přidané `ParentLoty`. Po rozkliknutí `Expanderu` dojde k zobrazení již hotových kazet, případně i zpracovávaných kazet a barevně zvýrazněn jejich stav. Obě krajní části budou mít opět pouze informační a vizualizační funkci. Nebudou tedy obsahovat žádné ovládací prvky.

Prostřední část bude opět stejně jako u vstupního uživatelského rozhraní obsahovat i ovládací prvky. Tentokrát nebude třeba přepínat mezi několika stavy, proto se v této části nebudou nacházet žádná tlačítka. Pouze nahoře bude opět velké Textové pole na zadávání kódů a pod ním dvě `ModernUI` dlaždice. Zde budou vizualizovat pouze přesun z teflonové kazety do umělohmotné a generování nového kódu.

Po přečtení nově vygenerovaného kódu dojde k vyvolání dialogového okna. V tomto modálním okně budou odhlašovány zpracované kazety z `PROMISu`. Bude se zde nacházet výpis nejdůležitějších informací, dále Textové pole pro zadání komentáře při odhlašování. A také `ComboBox` pro výběr čísla případné poškozené desky.

Místo pod Textovým polem bude opět sloužit pro zobrazování výpisů aplikace.

Kapitola 5

Výsledná implementace, testování a použité technologie

5.1 Návrhový vzor MVVM

`Model-View-ViewModel` je návrhový vzor pro `WPF` a `Silverlight` aplikace, který byl použit i v tomto projektu a nastudován v knize [12]. `MVVM` umožňuje oddělit aplikační logiku od uživatelského rozhraní. Díky tomu je možné vyvinout kratší, přehlednější kód, který je flexibilnější a případné pozdější úpravy neznamenají drastický zásah do celé aplikace.

`MVVM` odděluje data, stav aplikace a uživatelské rozhraní. Stav aplikace je uchováván v samostatné třídě nazývané `ViewModel`. `ViewModel` je nejdůležitější třída celého návrhového vzoru. Poskytuje všechna data pro uživatelské rozhraní, které se nazývá `View`. Data jsou poskytována v datových strukturách, které vyvolávají události při jejich změně. To umožňuje uživatelskému rozhraní zobrazit nová data ihned, jak se ve `ViewModelu` změní. `ViewModel` se skládá ze dvou základních prvků. Prvním je kolekce *`ObservableCollection`*, která vyvolá vždy událost při přidání a odebrání prvku. Druhým je rozhraní *`INotifyPropertyChanged`*. To popisuje událost, která nastane při změně vlastnosti ve `ViewModelu`. `MVVM` tvoří 3 vrstvy:

Model

Popisuje vlastní zobrazovaná nebo editovaná data. Může se jedna o databázová data, třídy kontraktů používaných `WCF` služeb nebo o libovolné vlastní objekty v závislosti na struktuře aplikace. `MVVM` neklade na model žádné specifické požadavky. Pouze model nesmí být nijak informován a stavu ovládacích prvků.

View

Reprezentuje uživatelské rozhraní popsané v deklarativním jazyce `XAML`. Obvykle se jedná o hlavní okna, stránky nebo ovládací prvky, u kterých se používá `DataBinding` a obousměrný `DataBinding` na `ViewModel`. Prvky uživatelského rozhraní nebývají zpravidla pojmenovávány a místo událostmi řízeného obslužného kódu se používá architektura `Commandů`.

ViewModel

Třída spojující `Model` a `View` a uchovávající stav celé aplikace. Obsahuje kód logiky prezentační vrstvy. Na jeho veřejné vlastnosti se `View` odkazuje pomocí `DataBindingu` a také obsa-

huje definice Commandů, což jsou akce, které můžeme z View volat. Aby se veškeré vlastnosti ViewModelu propagovaly do View, musí implementovat rozhraní *INotifyPropertyChanged*.

Porovnání MVVM s MVC a MVP:

- U MVVM tvoří jádro aplikace ViewModel a zároveň řídí celou aplikaci, kdežto View tvoří pouze uživatelské rozhraní, se kterým může uživatel interagovat.
- U MVC tvoří jádro aplikace View, ale je řízeno Controllerem, který řídí celou aplikaci.
- MVP je podobné MVC, jen Presenter může také řídit Model.

5.2 Implementace komponent serverové části

Volba programovacích prostředků serveru souvisela s volbou prostředků u uživatelského rozhraní. Protože u uživatelského rozhraní padla volba na technologii Silverlight, zvolil jsem jako implementační platformu serveru jazyk C# a platformu .NET. Pro volbu tohoto prostředí hrál i fakt, že pro tuto platformu byla dostupná knihovna pro spolupráci s firemním informačním systémem PROMIS. Jako poskytovatel dat slouží relační databázový systém od firmy Oracle. Zde došlo pouze k přizpůsobení se stávajícím technologiím dostupným v infrastruktuře firmy.

Vývoj serveru probíhal odděleně od vývoje uživatelského rozhraní, což bylo umožněno hlavně použitím návrhového vzoru MVVM 5.1. Díky tomuto postupu byly snáze opravovány chyby, kdy se při vývoji serveru řešily pouze specifické chyby vznikající pouze na serveru.

V této podkapitole bych se rád zmínil o implementaci každého ze čtyř modulů serveru a řešeních, která jsem využil. Protože celá aplikace byla navrhnuta jako stavová s ukládáním veškerých stavů do databáze, byl jako první implementován databázový modul, který je nejrozsáhlejší. Následovalo propojení s uživatelským rozhraním. Rozhraní bylo totiž z celého systému vytvořeno nejdříve. Poté byla přidána podpora čtečky kódů a tisk kódů. A nakonec došlo k propojení se systémem PROMIS.

5.2.1 Databáze

Návrh databáze spolu s popisem všech důležitých tabulek a obrázkem celého databázového schématu jsem popisoval v kapitole 4.2. Zde se už budu věnovat pouze implementaci.

Pro přístup k Oracle databázi je využito OLEDB API vyvinuté Microsoftem. To umožňuje přístup k datům z vícero zdrojů jednotným způsobem. OLEDB využívá COM technologii a bylo zvoleno hlavně pro jednoduchost případných pozdějších úprav, kdyby došlo ke změně typu databáze nebo poskytovatele databází.

Pro obsluhu databáze jsem vytvořil třídy, které obalují veškerá data a operace nad databází. Jsou to tři druhy tříd:

- Třídy nazývané Repository. V těchto třídách se nachází metody, které vykonávají přímo SQL příkazy nad jednotlivými tabulkami. Metody buď upraví samotnou tabulku nebo vrátí obsah tabulky v závislosti na zvoleném SQL příkazu.
- Třídy nazývané Gateway. Metody v těchto třídách volají metody ze tříd Repository. Pokud je třeba získat některý údaj z více různých tabulek, Gateway zavolá metody

ze dvou různých `Repository` a dostupná data spojí do jednoho objektu, který vrací jako odpověď z databáze.

- Třídy reprezentující samotné databázové entity. Je třeba mít zvlášť entity pro třídy typu `Repository` a zvlášť pro třídy typu `Gateway`.

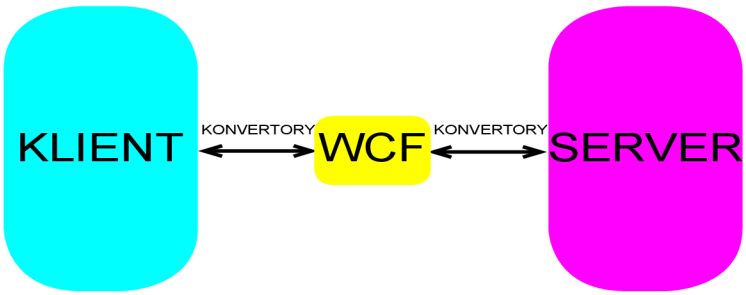
Ke každé tabulce je vytvořena vlastní `Repository` třída, která obvykle obsahuje potřebné `SELECT`, `UPDATE`, `DELETE` nebo `INSERT SQL` příkazy. Třída typu `Gateway` je pouze jediná a obsahuje všechny důležité metody pro získávání a ukládání potřebných dat do databází. `Gateway` objekt získá modul, který slouží k propojení uživatelského rozhraní se serverovými částmi, jako odpověď při volání metod z databázového modulu.

5.2.2 Propojení uživatelského rozhraní se serverem

Nejdůležitější modul celého serveru, který propojuje data z databází, `PROMISu` a hardwarových zařízení s uživatelským rozhraním. Modul je rozdělen na dvě části, kdy jedna komunikuje s klientem pro vstupní část systému a druhá s klientem pro výstupní část. Příliš odlišné však nejsou, pouze každý operuje s jinými datovými třídami, a proto využívá jiné databázové metody z `Gateway` třídy nebo volá jiné operace `PROMISu`.

Pro komunikaci klienta se serverem je využívána `WCF`[1] technologie od Microsoftu. Jedná se o jednotný framework, dostupný v technologii `.NET`, určený pro vytváření `SOA`[2] aplikací. Příkladem `SOA` aplikace může být klient-server aplikace a `WCF` slouží jako `API` sjednocující všechny dřívější technologie Microsoftu pro komunikaci mezi klientem a serverem. Základním prvkem `WCF` aplikace je Služba. Služba poskytuje jeden nebo více koncových bodů. Tyto koncové body slouží pro odesílání nebo příjem `SOAP` zpráv. Služba také publikuje metadata, která ji popisují. Kromě implementace samotné služby a koncových bodů je důležité také hostovací prostředí. To je místo, kde `WCF` služba poběží [9].

Klient má vytvořeny své vlastní entity, ve kterých udržuje zobrazovaná data, která pomocí konvertorů mapuje na entity `WCF` služby. Z `WCF` služby se opět pomocí konvertorů mapuje na entity serverové části a naopak.



Obrázek 5.1: Ilustrační obrázek komunikace klient-wcf-server.

Server využívá dva typy konvertorů. První typ konvertuje objekty `WCF` na objekty serveru a obráceně. Druhý typ provádí konverzi mezi `Gateway` objekty z databáze a objekty serveru.

[1] Windows Communication Foundation
[2] Service-Oriented Architecture

Objekty, kde server uchovává databázová nebo jiná data, jsou aktualizovány ve veřejné metodě `GetInfo`. V této metodě nedochází k žádným změnám dat, pouze se získávají dostupná data. `GetInfo` využívá k získání dat z databáze metody z `Gateway` tříd a navrácené `Gateway` objekty pomocí konvertorů převádí do serverových objektů. Kromě databázových tříd využívá například i metodu pro ověření autentizace uživatele z knihovny `PROMIS`.

Po `GetInfo` následuje dalších 5 metod při kterých dochází i ke změnám v databázích, `PROMISu` či stavu celé aplikace:

- `InsertedCassetteBarcode` metoda provádí veškeré akce, které vznikají v reakci na zadání čárového kódu. Tato metoda je hlavní řídicí metodou celého serveru. V závislosti na stavu aplikace ukládá, upravuje nebo maže data v databázi. Také provádí příkaz `SilentTrackIn` a u výstupní části aplikace komunikuje s tiskárnou.
- `RemoveCassetteBarcode` metoda, která při vytváření várky provede nezbytné změny atributů v tabulkách databáze tak, aby se kazeta v uživatelském rozhraní odebrala pryč z várky a vložila zpět do informačního seznamu kazet čekajících na zpracování. Je vyvolána při stisku tlačítka s křížkem u jedné ze tří kazet ve várce.
- `NewBatch` metoda přepne `DewaxInputSpec` aplikaci do stavu vytváření várky. Je vyvolána po stisku tlačítka `Nová`.
- `MoveBatch` metoda provede přihlášení všech kazet pomocí vyvolání příkazu na přihlášení v `PROMISu` ve várce a přepne aplikaci do stavu přesouvání kazet. Vyvolá ji tlačítko `Přihlaš`.
- `TrackIn` metoda nastaví v databázi u kazet ve várce čas začátku zpracovávání přístrojem a přepne aplikaci do výchozího stavu. Vyvolaná tlačítkem `Vytvoř`.

Kromě těchto metod server ještě obsahuje další pomocné metody, které využívají již implementované funkce v knihovně pro práci s `PROMISem`. Kromě volání pouze přizpůsobují data těmto metodám.

5.2.3 Čtení a tisk čárových kódů

Identifikace kazet s křemíkovými deskami je na celé výrobní lince založena na unikátních kódech, které obsahuje každá kazeta, deska nebo várka s deskami. Ovládání obou uživatelských rozhraní je proto kompletně přizpůsobeno k reakci na zadávané kódy. Všechny operace v aplikaci, kromě přihlašovacího dialogu a přepínaní stavu u `DewaxInputSpec` části, bývají vyvolávány po zadání kódu.

Přestože operátor může kódy zadávat i pomocí klávesnice, tak tento způsob je považován spíše za nouzový. Dvanáct hodin číst a následně zadávat více než desetimístné kódy desek by vedlo k překlepům, následným opravám a ve výsledku ke zpomalení operátorovy práce. Proto je u každého počítače, kde je třeba ukládat informace nebo řídit přístroj, přítomna čtečka čárových kódů. Samotná čtečka je připojena prostřednictvím sériového portu počítače a z hlediska této aplikace nebylo třeba žádné zvláštní implementace kódu, jelikož `.NET` framework má potřebné obslužné rutiny zabudovány přímo v sobě.

U výstupní části aplikace ještě vznikl požadavek na tisk nových čárových kódů pro další část výroby. Pro tuto funkci bylo třeba vytvořit třídu `PrintBarcode.cs`, která sestaví

potřebné příkazy pro ZEBRA Technologies tiskárnu a tyto příkazy tiskárně odešle prostřednictvím firemního intranetu. Příkazy bylo nutno sestavit pomocí dokumentace [1] v takzvaném ZPL jazyce, kterým je možno nakonfigurovat, jaký čárový kód tiskárna vytvoří. Tyto příkazy jsou uvedeny v souboru labelAfterDewaxSPEC.zpl. Cesta k tomuto souboru je uvedena v konfiguračním souboru aplikace. Třída PrintBarcode.cs načte potřebné příkazy ze souboru do datové struktury typu string a následně takto uložený text upraví tak, že zde vloží příslušný kód z databáze mezi správné příkazy. Nakonec takto vygenerovaný text je odeslán přes intranet pomocí socketu přímo do tiskárny, která vytiskne nalepovací kód na kazetu. IP adresa a port tiskárny je také uložen v konfiguračním souboru aplikace.

5.2.4 Promis

Informační a řídící systém PROMIS slouží v celé organizaci ON Semiconductor k uchovávání nejen přístupových údajů o operátorech, ale také obsahuje informace a identifikační údaje jednotlivých výrobních zařízení, popisy výrobních programů, informace o všech výrobních kazetách a deskách a také slouží k některým operacím. Pokud například není kazeta správně přihlášena v PROMISu, nelze zařízení spustit. Slouží tedy i jako poslední pojistka před případnou chybou. Veškeré výrobní aplikace vznikající v ON Semiconductor v poslední době tvoří grafické nadstavby nad tímto letitým systémem umožnující operátorům jednodušší ovládání. Aplikace DewaxInputSpec a DewaxOutputSpec nejsou výjimkou.

Z popisu PROMISu vyplývají pro aplikaci DewaxInputSpec tyto požadavky:

- Autentizace uživatele do PROMISu, bez přihlášení není možné s aplikací pracovat
- TrackIn - příkaz pro přihlášení sady/kazety pro zpracování na daném zařízení. Tento krok je nevratný a předpokládá se zpracování sady daným přístrojem. Pokud operátor provede TrackIn sady a do přístroje například naloží jinou, obvykle dochází ke zmetkování všech desek. U DewaxInputSpec k této operaci dochází po vytvoření várky, při stisku tlačítka Přihlásit před přepnutím stavu aplikace do režimu přesouvání.
- SilentTrackIn - zjednodušený příkaz TrackIn, kdy je u kazety/sady pouze kontrolována možnost přihlášení. Obvykle slouží k odhalení chyb operátora z předchozí části výroby. Sady nejsou přihlášeny, pouze jsou kontrolovány některé parametry, které by mohly úspěšnému přihlášení zabránit. Operace je vratná a opakovatelná. U DewaxInputSpec je tato operace volána při každém přidávání kazety do várky.

Pro DewaxOutputSpec je potřeba:

- Autentizace uživatele do PROMISu, bez přihlášení není možné s aplikací pracovat
- TrackOut - odhlášení sady/kazety z výrobního zařízení. Operace je také nevratná. Je u ní možné zadat tzv. scrapy, které označují počet a číslo zničených desek z celé sady/kazety. U aplikace je tato operace prováděna po přečtení nově vygenerovaného kódu pro kazety, kdy se objeví odhlašovací dialog s možností zadání scrapu nebo přidáni komentáře.

Pro provedení výše zmíněných operací používá aplikace ve vybraných místech metody z již implementované firemní knihovny pro práci s PROMISem. Jsou to metody PromisTrackIn, PromisSilentTrackIn, PromisTrackOut a PromisLogin, kterým je pouze potřeba ve vyžadovaném formátu předložit informace.

5.3 Implementace uživatelského rozhraní

V této podkapitole naváži na návrh z kapitoly 4.4 a popíši způsob, jakým byl tento návrh převeden do praxe. Protože implementace obou uživatelských rozhraní byla podobná, vyjmenuji hlavní část práce v podkapitole o `DewaxInputSpec` a v podkapitole o `DewaxOutputSpec` pouze zmíním odlišnosti.

Vytváření uživatelského rozhraní probíhalo v několika etapách:

1. Nejprve proběhl návrh a prvotní implementace základních prvků `DewaxInputSpec` aplikace v jazyce `XAML`.
2. Následně po konzultaci s operátory o způsobu jejich práce a po prostudování požadavků na aplikaci proběhla finální úprava a rozmístění prvků ovšem bez `ModernUI` stylu.
3. Poté došlo k implementaci testovacího simulátoru pro maximální odladění uživatelského rozhraní aplikace `DewaxInputSpec`. Simulátor tvořilo několik testovacích tříd, které simulovaly činnost serverové části.
4. Po odladění rozhraní `DewaxInputSpec` byl vytvořen návrh uživatelského rozhraní `DewaxOutputSpec` a opět proběhla prvotní implementace základních prvků v jazyce `XAML`, kdy bylo využito některých částí `DewaxInputSpec`.
5. Opět došlo k upravení dosavadní aplikace podle zpětné vazby operátorů a zakomponování všech požadavků ze specifikace.
6. Implementace testovacího simulátoru pro `DewaxOutputSpec` a odladění uživatelského rozhraní.
7. Následně probíhala implementace serverové části obou aplikací a její propojení s uživatelskými rozhraními.
8. Nakonec proběhlo stylizování dosavadních strohých uživatelských rozhraní, které pouze obsahují význačné ovládací prvky, do `ModernUI` stylu.

5.3.1 Implementace Hlavní stránky DewaxInputSpec

Výsledné uživatelské rozhraní je tvořeno jednou hlavní obrazovkou, která umožňuje ovládání aplikace, ale přitom plní i vizualizační funkci. Její rozdělení s ovládacími prvky umístěnými uprostřed mezi dva informační seznamy jsem už popisoval v kapitole návrhu.

Akce spojené s prací se systémem `PROMIS` jsem implementoval formou dialogů. Jednak je hlavní stránka zjednodušena o tyto ovládací prvky, a za druhé může takový dialog operátora upozornit, že provádí kritickou část své práce.

Každá stránka v `Silverlightu` je nazývána jako `UserControl`. Je tvořena dvěma soubory.

První má za názvem přítomnu příponu `XAML`. Tento `UserControl.xaml` definuje vzhled samotné obrazovky a je vytvořen pomocí značkovacího jazyka `XAML` (z angl. Extensible Application Markup Language), který kromě všech prvků celé obrazovky nabízí i možnost deklarovat vektorovou grafiku a animace. Další výhodou je i to, že jednotlivé `UserControls` je možné vkládat do sebe, jako klasické ovládací prvky. Takto jsem například lépe dekomponoval kód a vytvořil dialogy v oddělených souborech a poté je vložil podobně, jako třeba tlačítko.

Druhý je normální zdrojový soubor jazyka C# `UserControl.cs`. Tento soubor je nazýván jako `Code-Behind` a obvykle obsahuje obslužné metody, pro reakci uživatelského rozhraní na uživatelovy pokyny. Bez tohoto souboru by vznikl pouze vzhled rozhraní bez funkčnosti. U větších projektů se složitějším rozhraním se však obsluha rozhraní v `Code-Behind` stává složitou a nepřehlednou. Já jsem se tomuto problému vyhl díky návrhovému vzoru `MVVM`, kde `Code-Behind` nahrazují třídy `ViewModels`.

Vzhled hlavní stránky je implementován v souboru `MainPage.xaml`. `ModernUI` dlaždice jsem vytvořil taky jako `UserControls` a jsou umístěny ve složce `Views`.

Chování uživatelského rozhraní ve ViewModelech

Hlavní obsluha celého uživatelského rozhraní se nachází ve třídě `MainViewModel`. V této třídě se nachází, jak zobrazované vlastnosti, tak pomocné metody, které v závislosti na akci připraví potřebná data a poté je předají ke zpracování serverové části aplikace.

`MainViewModel` obsahuje listy `Batches, InputSpecCassettes, FromCassettes, ToCassettes`, ze kterých jsou zobrazována data. Tyto listy jsou typu `ObservableCollection`, což je speciální list používaný v `MVVM` a který vždy informuje o změně svého obsahu. Prvky v těchto listech jsou v uživatelském rozhraní viditelné díky `DataBindingu`. Konkrétně je například v `MainPage.xaml` definován prvek `Seznam` a tomu je předán list `InputSpecCassettes`:

```
...
<ListBox
        Grid.Row="1"
        Margin="0"
        Width="Auto"

        ItemsSource="{Binding InputSpecCassettes}"
        ItemTemplate="{StaticResource LotListStyle}"
        ItemContainerStyle="{StaticResource LotHorizontNonSelect}"
        ItemsPanel="{StaticResource FluidMove}"/>
...
```

`Batches` zobrazuje várky v pravém listu obrazovky, `InputSpecCassettes` zobrazuje kazety v levém listu a `FromCassettes, ToCassettes` zobrazují data v `ModernUI` dlaždicích.

Posledním viditelným prvkem, kromě čtyř listů je `ReturnCode` typu `string`. Tento prvek je předán `Popisku` a zobrazuje informační a chybové kódy aplikace.

Po prvcích, které zobrazují data, obsahuje `MainViewModel` tři `Commandy`. Tento mechanismus se v `MVVM` využívá pro vyvolání akcí po stisku tlačítka. Po stisku jednoho ze tří tlačítek se tedy zavolá serverová metoda, která změní stav celé aplikace.

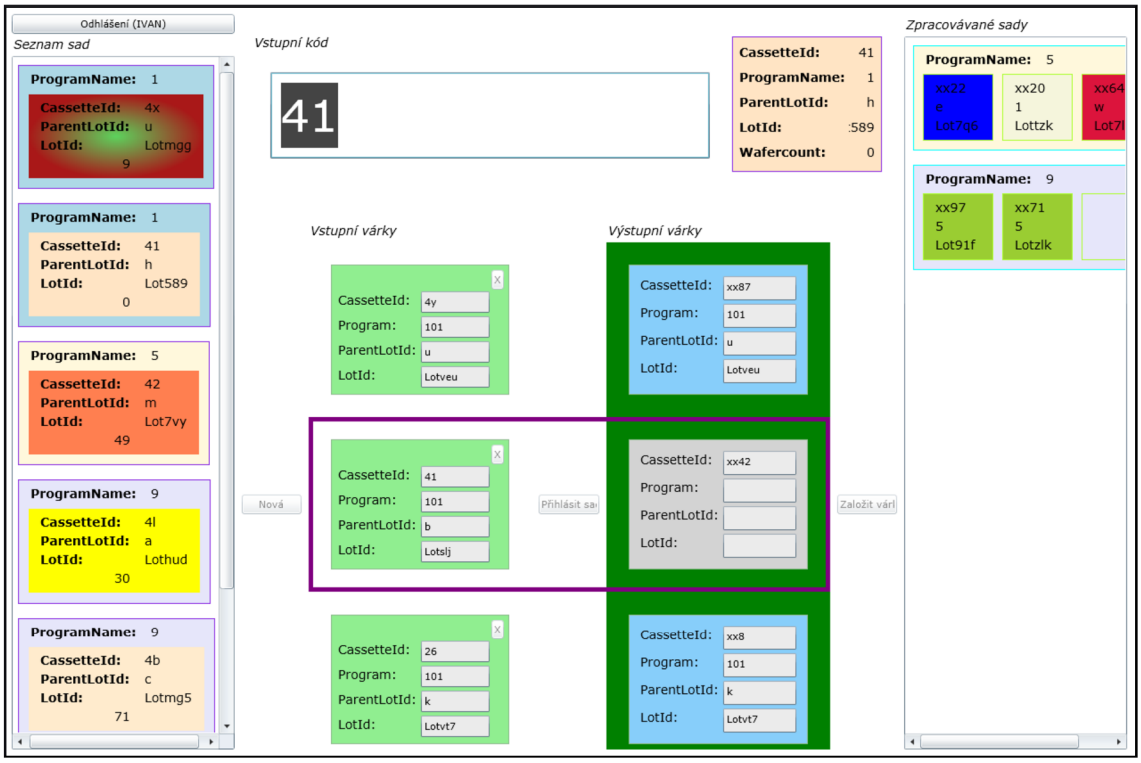
Na volání akcí po zadání kódu do `Textového pole` se zde nachází `string EnterBarcode` a metoda `BCAction`. `EnterBarcode` je pomocí obousměrného `DataBindingu` propojen s `Textovým polem`.

Posledním členem `MainViewModelu` je metoda `UiUpdate` s ostatními pomocnými metodami. Tato je v závislosti na časovači volána každou vteřinu, aby obnovila stav uživatelského rozhraní. `UiUpdate` totiž volá serverovou metodu `GetInfo` a plní `Batches, InputSpecCassettes, FromCassettes, ToCassettes` a `ReturnCode` dostupnými daty v databázi na serverové straně. Data jsou konvertována mezi objekty uživatelského rozhraní a

WCF serverem podobně, jako u serveru. UiUpdate kromě získávání dat, vyvolává i některé animace a zajišťuje i ostatní zvýrazňování a vizualizaci kroků v uživatelském rozhraní v závislosti na stavu aplikace nebo dostupným datům.

Kromě MainViewModel, která přísluší k MainPage.xaml, mají i ostatní UserControl třídy své vlastní ViewModels. Dvě třídy ViewModel přísluší i dvěma krajním listům. Protože vzhled prvků v listu je možné upravit pomocí DataTemplate, tak všechny akce k těmto DataTemplate jsou obsluhovány zde.

Propojením všech výše zmíněných prvků vzniká základní podoba uživatelského rozhraní. K finální fázi už chybí pouze doplnění ModernUI stylu:



Obrázek 5.2: Podoba UI bez ModernUI Stylu.

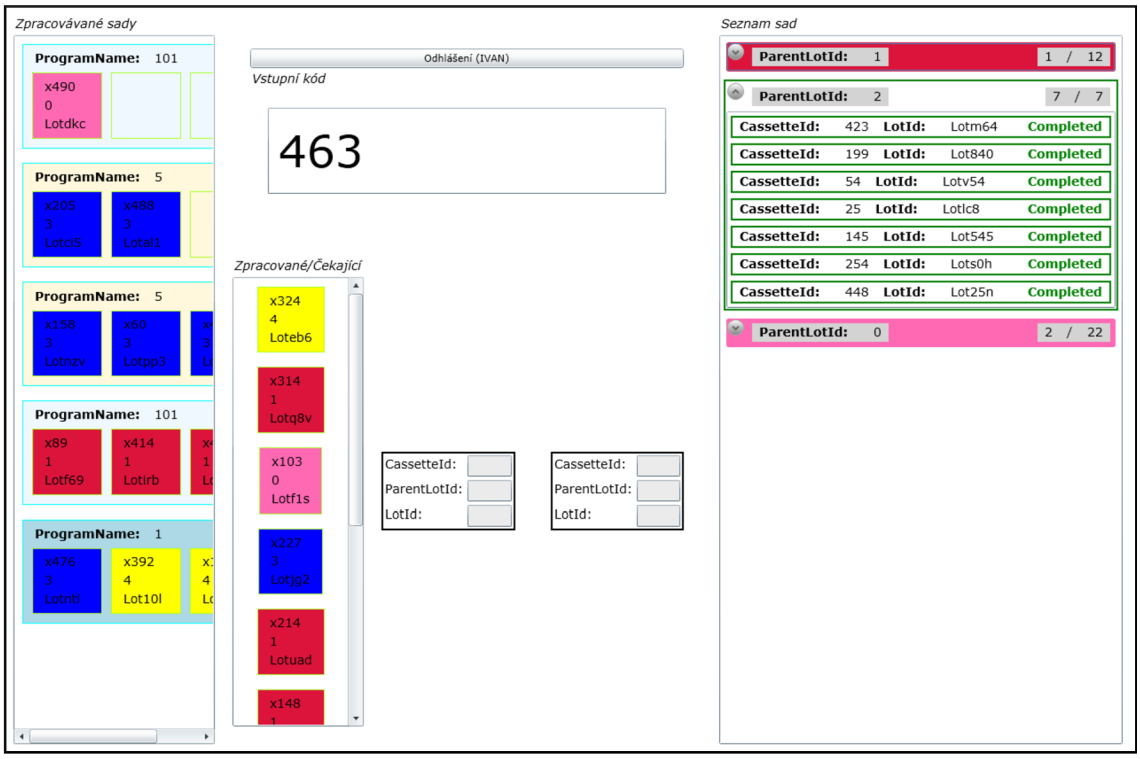
5.3.2 Implementace Hlavní stránky DewaxOutputSpec

Vzhled hlavní stránky je implementován stejně jako u DewaxInputSpec v MainPage.xaml a ostatní definované UserControls a jsou umístěny ve složce Views.

Chování uživatelského rozhraní ve ViewModels

MainViewModel tentokrát obsahuje listy SpecBatches, DoneCassettes a WaitCassettes, ze kterých jsou zobrazována data. Opět jsou typu ObservableCollection. SpecBatches zobrazuje várky v levém listu obrazovky, DoneCassettes zobrazuje kompletně hotové kazety v pravém listu a WaitCassettes zobrazují v listu kazety ze zpracovaných várek. Dále jsou přítomny prvky FocusCassetten a PrintCassetten obsahující data pro dvě ModernUI dlaždice. Posledním viditelným prvkem je opět ReturnCode typu string vracející kódy aplikace.

Následující řešení opět používá `UiUpdate` metodu, která nyní plní jinak pojmenované listy. Propojením všech výše zmíněných prvků vzniká základní podoba uživatelského rozhraní. K finální fázi už chybí pouze doplnění `ModernUI` stylu:



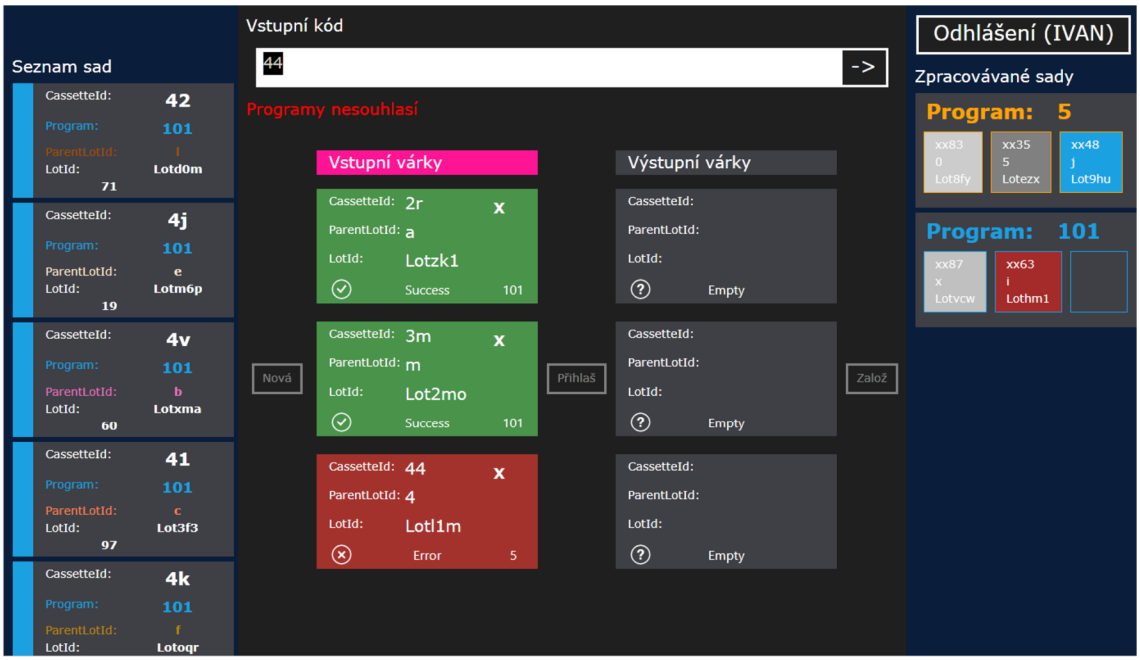
Obrázek 5.3: Podoba UI bez ModernUI Stylu.

5.3.3 Stylizování aplikace do ModernUI stylu

Poslední prací bylo stylizování obou uživatelských rozhraní. Což znamenalo nejen změnu barev a vzhledu, ale pro intuitivnější interakci i přidání animací. Právě přidávání animací tvořilo část práce na uživatelském rozhraní, kde jsem se potýkal s nejvíce problémy. Animace bylo nutné přizpůsobit použitému návrhovému vzoru `MVVM`. I poté se vyskytovaly některé překážky. Například pro animování odebírání prvku ze seznamu, bylo nutné přizpůsobit i serverovou část, kdy server musí klientu i vracet index mazaného prvku.

Při stylizování prvků, jsem využil předpřipravených stylů [10], které jsem si dále přizpůsobil. Hlavně u `DewaxInputSpec`, kde je v rozhraní přítomno více prvků, se stalo komplikovanější i přizpůsobení rozhraní pro nižší rozlišení obrazovek. Použil jsem totiž rozměry všech dlaždic podle knihy [6] a při optimalizaci pro nejnižší mnou vyžadované rozlišení 1366x768 jsem doslova hledal každý pixel ve vertikálním i horizontálním směru. Kompromisem například došlo výraznému zmenšení tří ovládacích tlačítek aplikace.

Kromě předpřipravených stylů pro generické prvky však bylo třeba vytvořit `ModernUI` dlaždice, vytvořit vlastní animace a hlavně systém zvýrazňování právě vybraných prvků. Rozhraní obsahuje tři druhy dlaždic. Dlaždice v levém listu kazet obsahují boční barevný pás pro zvýraznění programu. Dále ještě barevně zvýrazňují `ParentLot`. Dlaždice várek

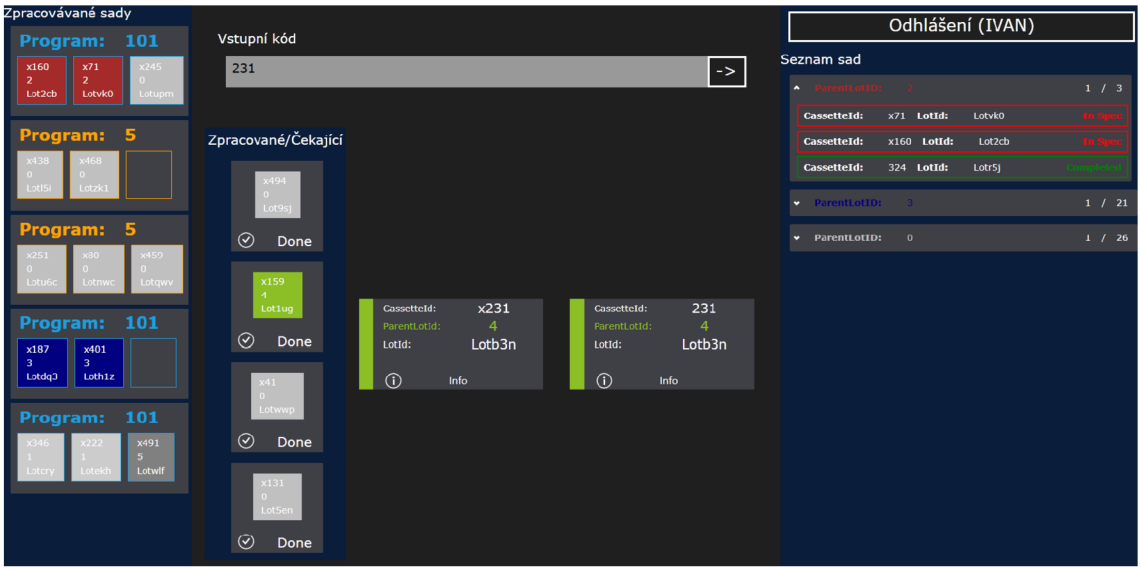


Obrázek 5.4: Hotový DewaxInputSpec.

v pravém pásu obsahují barevný nápis programu a tři barevné čtverečky pro naznačení kazet ve várce. Čtverec je průhledný, pokud je prázdný, jinak má barvu `ParentLotu`. Poslední typ dlaždic využívá šest prvků, které zvýrazňují přesouvání kazet. Ve sloupci, kde je zobrazováno sestavování várky dlaždice, po vložení kazety změní barvu na zelenou pokud programy souhlasí nebo na červenou pokud nesouhlasí. V pravém sloupci po přesunutí se barva pouze změní v modrou. Nakonec bylo potřeba vyřešit zvýraznění rozdílného stavu aplikace a zvýraznění focusu právě přesouvané kazety. U zvýraznění focusu byl použit typický prvek z `ModernUI`, kdy vybraná kazeta pomocí animace kolem sebe zobrazí výrazný růžový rámeček s fajfkou v pravém horním rohu. Pro zvýraznění stavu aplikace používám podbarvení nadpisu sloupce opět stejnou růžovou barvou. K tomuto řešení jsem se inspiroval u aplikace `Metro Commander` [13], který takto taky zvýrazňuje používané sloupce. Pro zvýraznění, že boční sloupce jsou pouze informační, mají tyto sloupce tmavě modrý barevný podklad namísto světle černého u ovládacích prvků.

U uživatelského rozhraní `DewaxOutputSpec` zabralo nejvíce práce přizpůsobení vzhledu prvku `Expander` do `ModernUI` stylu.

Levý sloupec je převzatý pravý sloupec od `DewaxInputSpec`. Druhý levý sloupec jako jediný v obou aplikacích používá úzké čtvercové dlaždice. V této dlaždici je zobrazen nápis Done a čtverec s barvou `ParentLotu` a informacemi o kazetě. Poslední úplně pravý sloupec obsahuje `Expandery` s barvou `ParentLotu`. Po rozkliknutí se zobrazí známé kazety a jejich stav. Zelený rámeček mají už hotové, červený mají kazety před `DewaxSpec` přístrojem a žlutý kazety vytáhlé z `DewaxSpecu`. Prostřední dvě `ModernUI` dlaždice na boku pásem zvýrazňují `ParentLot`.



Obrázek 5.5: Hotový DewaxInputSpec.

5.4 Použité technologie a knihovny

Při vývoji aplikace bylo použito několik již vytvořených knihoven. Tyto knihovny zvýšily rychlost celého vývoje nebo vyřešily problémy, které jsem nebyl schopen vyřešit sám. Všechny knihovny jsou distribuovány pod volně dostupnými licencemi až na jedinou, která vznikla přímo ve firmě ON Semiconductor pro práci se systémem PROMIS. Pro ostatní firemní aplikace, kam spadá i `DewaxInputSpec` a `DewaxOutputSpec`, však není její použití nijak omezeno.

5.4.1 MVVM Light Toolkit

`MVVM Light Toolkit` je sada komponent, která usnadňuje vývojářům v `XAML/C#` frameworcích implementaci návrhového vzoru `Model - View - ViewModel` jehož výhody jsem popisoval v 5.1. Knihovna je podporována samotným Microsoftem na stránce `CodePlex`[1] a je možnost jejího jednoduchého zakomponování do projektu pomocí balíčkovacího systému `Visual Studia` s názvem `Nuget`. Autorem knihovny je Laurent Bugnion a její princip a použití je popsáno na jeho webu [4].

5.4.2 PROMIS knihovna pro C# projekty

Aby všichni vývojáři v ON Semiconductor organizaci nemuseli pro komunikaci s firemním systémem `PROMIS` vždy znovu vyvíjet kód, jehož funkčnost budou potřebovat i další aplikace, vyvinuli programátoři, kteří spravují `PROMIS`, univerzální knihovnu, která je používána prakticky všemi aplikacemi v celé organizaci. Protože v organizaci nevznikají pouze aplikace v jazyce `C#`, existují i varianty pro `Java` nebo `C++` aplikace.

Použití této knihovny jsem zmínil v podkapitole 5.2.

[1] Je internetový projekt společnosti Microsoft určený k hostování otevřeného softwaru. http://www.codeplex.com/ Veškerý software dostupný na těchto stránkách je dostupný pod otevřenou licencí CPOL.

5.4.3 OLEDB

Pro komunikaci s databázemi jsem využil v `.NET` frameworku přímo dostupné `OLEDB`. Je to aplikační rozhraní založené na `COM` technologii umožňující přístup k datům. `OLEDB` poskytuje přístup k datům v jakémkoliv formátu(databáze, tabulky, text), pokud pro ně existuje `OLEDB` poskytovatel. Práci s tímto `API` jsem nastudoval na stránce [7].

5.4.4 Silverlight Toolkit - Expander

`Silverlight` je aplikační platforma od společnosti Microsoft pro vývoj převážně webových aplikací nebo aplikací pro `Windows Phone` platformu. Oproti plnohodnotné platformě `.NET` je pro menší hardwarovou náročnost ochuzena o méně používané prvky a funkce. Protože však některé odstraněné prvky vývojáři stále požadovali, vznikl Microsoftem oficiálně vydaný `Silverlight Toolkit`, které samotný `Silverlight` rozšiřuje o množství nových prvků [5].

Mezi tyto prvky patří i `Expander`, který jsem využil u výstupního listu `DewaxOutputSpec` uživatelského rozhraní. `Expander` je prvek, který seskupuje list více prvků. Když jsou prvky seskupené, je viditelná pouze hlavička se názvem a rozevírací tlačítko, kterým je možné zobrazit celý seznam prvků.

5.4.5 Silverlight Metro Theme

Protože `Silverlight` Microsoft přestal vyvíjet už v roce 2011, neuvolnil pro něj žádné `ModernUI` styly. Abych se vyhnul zdlouhavému upravování všech ovládacích prvků aplikace použil jsem už takto vytvořené styly od Saschy Woltera dostupné pod licencí CPOL na stránce CodePlex [10].

Sascha Wolter vzal styly vytvořené pro `WPF` a převedl je i do `Silverlightu`. Výsledkem je zdrojový soubor `MetroDesignResourceDictionary.xaml`, který stačí přidat do svého projektu a základní prvky jako tlačítka, textová pole, seznamy budou stylizovány.

Já jsem si styly ještě upravil podle vlastní potřeby. `ModernUI` dlaždice, animace a další věci charakteristické pro `ModernUI` už jsem musel vytvořit vlastními silami.

Kapitola 6

Testování, nasazování a zhodnocení dosažených výsledků

Celkovou funkčnost systému není možné otestovat bez nainstalované Oracle databáze a vytvořených tabulek. K otestování chování uživatelského rozhraní, je však možné využít testovací simulátor, který jsem v průběhu vývoje vytvořil pro odladění maximálního množství chyb v uživatelských rozhraních, bez vlivu serverové části. Po zapnutí simulátoru namísto reálné serverové části, dojde ke generování náhodně vytvořených kazet, s kterými je možné otestovat všechny prvky rozhraní.

Při vývoji byla aplikace průběžně testována ve virtuálním počítači, kde byla vytvořena kompletní Oracle databáze, aby došlo k maximálnímu odhalení všech chyb. V určitém stádiu, kdy aplikace už byla ve virtuálním počítači odladěná, mohlo dojít k testování přímo v čistých prostorách přímo v produkční lince. I přesto, že návrh uživatelského rozhraní byl už od počátku upravován a vytvořen i podle připomínek operátorů, bylo třeba ještě provést závěrečné vyhodnocení hotového návrhu. Proto se v tento moment k vyzkoušení aplikace dostali i samotní operátoři.

6.1 Zpětná vazba

Protože pohyb v čistých prostorách je kvůli zachování čistoty a bezpečnosti komplikovaný, musel jsem metody sbírání zpětné vazby přizpůsobit i těmto omezením. Nakonec jsem se rozhodl využít metodu společného procházení doplněnou reportováním dojmů při testovacím běhu. Tento způsob jsem zvolil z důvodu ne vždy kompletní přítomnosti všech operátorů. Počet operátorů, kteří přijdou do styku s tímto softwarem bude osm, ale v době, kdy jsem měl povolen přístup do čistých prostor byla díky směnám v pracovní době přítomna vždy pouze polovina. S přítomnými operátory jsem provedl metodu společného procházení, kdy jsem operátorům pouze popsal způsob práce, ale neukázal. Následně jsem je vždy odděleně pozoroval při práci s rozhraním a zaznamenával případné připomínky. Nepřítomní operátoři mohli testovat při měsíčním testovacím běhu a jejich poznatky jsem dostal formou reportu společně i s odhalovanými chybami. Celkově proběhly dvě fáze testování. Při obou typech testování a v obou fázích jsem operátory nechal odpovědět na několik otázek:

1. Jste pokročilý uživatel PC?
2. Jaký byl Váš první dojem z aplikace?
3. Jsou informační pásy na stranách dostatečně přehledné?

4. Je způsob vizualizace přesouvání kazet dostatečně názorný?

5. Jaké další chyby a nedostatky shledáváte za nejzávažnější?

6. Jak hodnotíte použitelnost aplikace při zlepšení Vaší práce?

V první fázi operátoři testovali aplikaci pouze se základním vzhledem bez jakéhokoli stylizování a animací v aplikaci. Zde dokázali po krátkém komentáři bez předchozího předvedení stylu práce projít kompletní operací u obou rozhraní pouze dva operátoři. Tito operátoři o sobě prohlásili, že jsou pokročilými uživateli PC. Ohlasy ostatních dvou přítomných a operátorů, zkoušejících aplikaci bez mé přítomnosti, v tuto chvíli byly spíše rozpačité. Přesto toto úvodní nasazení pomohlo s odhalením několika chyb, hlavně v serverové části. Nejzávažnější chybou byl pád aplikace při zadávání poškozených desek u odhlašovacího dialogu.

Účastník	Jste pokročilý uživatel PC?	Jaký byl Váš první dojem z aplikace?	Jsou informační pásy na stranách dostatečně přehledné?	Je způsob vizualizace přesouvání kazet dostatečně názorný?	Jaké další chyby a nedostatky shledáváte za nejzávažnější?	Jak hodnotíte použitelnost aplikace při zlepšení Vaší práce?
1	ANO	Špatný	ANO	ANO	Přidat více popisků.	Použitelná
2	NE	Neutrální	ANO	ANO	Příliš mnoho barev.	Nevím
3	ANO	Neutrální	NE	ANO	Příliš barevné.	Použitelná
4	NE	Špatný	NE	ANO	Výstižnější popisky.	Nepoužitelná
5	NE	Neutrální	ANO	NE	Pád aplikace.	Spíše nepoužitelná
6	NE	Neutrální	ANO	ANO	Přidat více popisků.	Spíše nepoužitelná
7	NE	Neutrální	NE	ANO	Pád aplikace.	Nevím
8	NE	Špatný	ANO	NE	Ztráta focusu.	Nevím

Obrázek 6.1: Odpovědi z první fáze. Světle modrou barvu mají operátoři odpovídající bez mé přitomnosti.

Druhá návštěva proběhla už s kompletně hotovým a stylizovaným rozhraním. Samozřejmě došlo k opravě dříve odhalených chyb. Přestože všechny použité prvky a jejich rozmístění nebylo nijak významně pozměněno, byla reakce na stylizované rozhraní oproti prvnímu testu velice pozitivní. Ačkoli operátoři už z prvního testu a následného pilotního běhu ovládání aplikace znali, jejich subjektivní hodnocení bylo výrazně lepší. V prvních ohlasech shledali aplikaci dobře použitelnou pouze dva z osmi operátorů. Nyní to bylo sedm. Jako hlavní důvod nárůstu uživatelské přívětivosti vyplynulo z odpovědí operátorů hlavně doplnění animací. Přestylizování ovládacích prvků sice znamenalo vstřebávat změny, přesto převážil pozitivní dojem z animací.

Stejný výsledek jsem zaznamenal pokud jsem aplikaci předvedl nezávislé osobě, která s aplikací pracovat v reálu nikdy nebude.

Účastník	Jste pokročilý uživatel PC?	Jaký byl Váš první dojem z aplikace?	Jsou informační pásy na stranách dostatečně přehledné?	Je způsob vizualizace přesouvání kazet dostatečně názorný?	Jaké další chyby a nedostatky shledáváte za nejzávažnější?	Jak hodnotíte použitelnost aplikace při zlepšení Vaší práce?
1	ANO	Kladný	ANO	ANO	Světlejší tón.	Použitelná
2	NE	Neutrální	ANO	ANO	Mnoho písem.	Použitelná
3	ANO	Kladný	NE	ANO	Světlejší tón.	Použitelná
4	NE	Kladný	ANO	ANO	Výstižnější popis.	Použitelná
5	NE	Kladný	ANO	NE	Ztráta focusu.	Použitelná
6	NE	Kladný	ANO	ANO	Více popisků.	Spíše nepoužitelná
7	NE	Neutrální	NE	NE	Nevím.	Použitelná
8	NE	Kladný	ANO	ANO	Ztráta focusu.	Použitelná

Obrázek 6.2: Odpovědi z druhé fáze. Světle modrou barvu mají operátoři odpovídající bez mé přitomnosti.

Jako zpětnou vazbu považuji i komentáře vedoucího. Společně s jeho a připomínkami operátorů jsem pozměnil nebo upravil tyto věci:

- Za nejdůležitější připomínku, osobně, hodnotím požadavek od mého vedoucího na přidání animací. Protože významným způsobem pozitivně ovlivnilo subjektivní hodnocení aplikace operátory.
- Doplnění popisů dlaždic podle připomínek operátorů.
- Sjednocení použitých písem a zredukování jejich počtu na pokyn vedoucího.
- Oprava pádu aplikace při odhlašovacím dialogu kazet.
- Implementace logování v serverové části pro snazší odhalování chyb v aplikaci.
- Budoucí nahrazení informační tabulky o kazetách z `DewaxInputSpec`, pro celkově čistější prostředí. Tento prvek bude nahrazen nejspíše novou obrazovkou.
- Poslední připomínka souvisí s pohodlím při ovládání. Protože operátor bude používat hlavně čtečku čárových kódů, je potřeba, aby byl focus neustále po jakémkoli kroku v rozhraní vrácen zpět do Textového pole, aby operátor nemusel klikat myší a poté číst kódy. Tento nedostatek se mi dosud nepovedlo odstranit, protože jeho řešení není úplně triviální. Pro budoucí verzi aplikace má tento požadavek vysokou prioritu na opravu.

Po vyhodnocení a úpravách byla aplikace nasazena do 2 měsíčního pilotního běhu, kdy bude na celé lince docházet k běhu a odhalování dalších chyb. Po pilotním běhu už dojde k nasazení do výroby.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit dvě aplikace sloužící ke zefektivnění výroby křemíkových desek ve firmě ON Semiconductor. Při návrhu jsem se snažil splnit nejenom všechny požadavky ze strany firmy, ale při navrhování uživatelských rozhraní obou aplikací dodržovat vhodné zásady tvorby uživatelských rozhraní. V rámci možností se tyto dva požadavky povedlo splnit.

Výsledné aplikace se jmenují `DewaxInputSpec` a `DewaxOutputSpec`. `DewaxSpec` je název konkrétního výrobního zařízení na čištění výrobních desek a `DewaxInputSpec` slouží pro vstup a `DewaxOutputSpec` slouží pro výstup zařízení. Aplikace umožňují operátorovi vizualizovat pracovní postup a řídit výrobu prostřednictvím firemního informačního systému PROMIS. Obě uživatelská rozhraní jsou navrhována a vysloveně počítají s přítomností čtečky čárových kódů jako hlavním ovládacím prvkem aplikace.

V současnosti běží obě aplikace na výrobní lince, která je v testovacím provozu. Na přelomu července a srpna v roce 2013 by po odladění celé linky mělo dojít k zahájení výroby. Do té doby by mělo dojít k odhalení a opravě většiny chyb.

Jako nejbližší rozšíření stávající verze aplikace bude vyřešení problému s neměnícím se focusem u Textového pole případně doplnění nové informační obrazovky, kde budou zobrazeny maximálně podrobné informace o kazetách. Do budoucna by mohlo dojít sofistikovanějšímu řízení uživatelských práv. Nyní má aplikace podle přihlášené osoby pouze stav pro čtení, bez možnosti cokoliv dělat, nebo plnou kontrolu nad aplikací. Další možnou změnou by mohla být optimalizace `SQL` dotazů do databází, které jsou u této aplikace využívány v hojné míře. Zde ovšem bude třeba získat informace o vytíženosti při finálním běhu a až poté optimalizovat. Posledním a dlouhodobějším cílem je přizpůsobení klientských částí aplikace i pro běh na mobilních telefonech či tabletech. Idea je taková, že kdokoliv si například na služebním telefonu, tabletu či počítači, které budou připojeny do firemního intranetu, bude moct zobrazit průběh práce operátora. Nyní tento požadavek bude dostupný jen na omezeném množství počítačů.

Osobně mi byla práce přínosem především v poznání, že proces návrhu uživatelského rozhraní není jen doplňkovou disciplínou vývoje aplikace. Nýbrž je to rozsáhlá a sofistikovaná disciplína, obzvláště v dnešní době, kdy počítače obklopují úplně všechny, nejen pár odborných programátorů. Částečným přínosem byla i možnost nahlédnout na metodologie návrhu softwaru ve velké nadnárodní společnosti.

Literatura

[1] *ZPL II Programming Guide, Volume One: Command Reference for X.10.* Zebra Technologies Corporation, Listopad 2006.

[2] Halpin, T.; Morgan, T.: *Information Modeling and Relational Databases.* Morgan Kaufmann. druhé vydání, 2008, iSBN 978-0-12-373568-3.

[3] Šik, J.; Lorenc, M.; Válek, L.: Technologie růstu monokrystalů křemíku czochralskiho metodou. In Škola růstu krystalů 2004. Maxdorf, Praha, 2004, s. 48-57.

[4] Laurent Bugnion: MVVM Light Toolkit. [online], 2007 [cit. 2013-04-20].
URL http://www.galasoft.ch/mvvm/

[5] Microsoft: Silverlight Toolkit. [online], 2011 [cit. 2013-04-20].
URL http://silverlight.codeplex.com/

[6] Microsoft: Windows 8 User experience guidelines. [online], 2012-08-14 [cit. 2013-04-20].
URL http://go.microsoft.com/fwlink/p/?linkid=258743

[7] Microsoft: OLE DB Tutorial. [online], 2012 [cit. 2013-04-20].
URL http://msdn.microsoft.com/en-us/library/aa288452%28v=vs.71%29.aspx

[8] ON Semiconductor: Technologie výroby křemíku. [online], 2013 [cit. 2013-04-20].
URL https://www.onsemi.com/PowerSolutions/content.do?id=15031

[9] Papa, J.: *Silverlight datové služby.* Zoner Press, 2009, iSBN 978-80-7413-041-0.

[10] Sascha Wolter: Silverlight Metro Theme. [online], 2010-04-29 [cit. 2013-04-20].
URL http://metrotheme.codeplex.com/

[11] Shneiderman, B.; Plaisant, C.: *Designing the User Interface.* Addison-Wesley. čtvrté vydání, 2005, iSBN 0-321-19786-0.

[12] Smith, J.: *Advanced MVVM.* Josh Smith, 2010, iSBN ASIN: B003QF1RKI.

[13] Studio, B.: Metro Commander. [online], 2013 [cit. 2013-04-20].
URL http://apps.microsoft.com/windows/cs-cz/app/metro-commander/30456798-c35c-4a78-a72a-5e5c3271c36d

[14] Swick, R. R.; Ackerman, M. S.: The X Toolkit: More Bricks for Building User-Interfaces or Widgets for Hire. In *USENIX Winter'88*, 1988, s. 221–228.

[15] Zemčík, P.: *Tvorba uživatelských rozhraní.* Vysoké Učení Technické v Brně, Fakulta Informačních Technologií, Listopad 2006.

Příloha A

Obsah DVD

- Program - obsahuje projekt pro Visual Studio 2012 včetně zdrojových kódů aplikace a potřebných knihoven pro překlad.
- Dokumentace - obsahuje LaTeXzdrojové kódy bakalářské práce a kompletní práci v PDF formátu.
- Plakat - obsahuje PDF soubor s plakátem k aplikaci.
- Video - obsahuje video soubor s prezentací vlastní aplikace.

Příloha B

Manual

Tento manuál popisuje, jak přeložit a spustit výsledné aplikace. Ve složce `complete/ei/sln/spec` je dostupný solution file pro Visual Studio 2010. Pro správnou funkčnost je třeba nainstalovat i `Silverlight5_Tools` přibalené ve složce `complete`. Po překladu je možné spustit obě aplikace. Ke spuštění je třeba mít nainstalovaný `Silverlight 5`. Kvůli složité instalaci Oracle databází a nemožnosti se připojit k systému PROMIS, je aplikaci možné otestovat pomocí simulátoru, který nahrazuje serverovou část. Se simulátorem lze kompletně otestovat veškerou funkčnost uživatelského rozhraní.

- DewaxInputSpec lze bez Visual Studia spustit ve složce `complete` pomocí souboru `ei/src/cz2/spec/dewax/ui/input/client/ei.dewax.input.ui.client.sl5/Bin/Debug/ei.dewax.input.client.sl5TestPage.html`. Soubor byl testován v `Internet Exploreru`, odkud je možné stiskem pravého tlačítka myši a výběrem volby Instalovat spustit nainstalovat.
- DewaxOutputSpec lze bez Visual Studia spustit ve složce `complete` pomocí souboru `ei/src/cz2/spec/dewax/ui/output/client/ei.dewax.output.ui.client.sl5/Bin/Debug/ei.dewax.output.client.sl5TestPage.html`. Soubor byl testován v `Internet Exploreru`, odkud je možné stiskem pravého tlačítka myši a výběrem volby Instalovat spustit nainstalovat.