Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Informační systém pro podporu podnikových procesů oddělení nákupu

Bc. Petr Slavotínek

Vedoucí práce: Ing. David Buchtela, Ph.D.

7. května 2013

Poděkování

Rád bych poděkoval své přítelkyni, rodině a nejbližším přátelům za jejich vynikající podporu po celou dobu studia. Poděkování patří také zaměstnancům společnosti Magna Exteriors & Interiors za jejich ochotu ke spolupráci a rovněž vedoucímu práce Ing. Davidu Buchtelovi, Ph.D., který se mé práce ujal a poskytl mi cenné rady.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Jablonci nad Nisou dne 7. května 2013 .....

Odkaz na tuto práci

Slavotínek, Petr. *Informační systém pro podporu podnikových procesů oddělení nákupu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

Abstract

The goal of this thesis is to develop an information system to support business processes of the purchasing department of Magna Exteriors & Interiors Bohemia s.r.o. The thesis introduces the company and its processes as well as principles of web applications development and used technologies. It describes the entire development process from introductory studies and analysis, through design and implementation using ASP.NET MVC framework, to testing and evaluation. The result is a functional application used by employees.

Keywords Magna Exteriors & Interiors, .NET framework, ASP.NET MVC, Transact SQL, jQuery

Abstrakt

Cílem práce je vytvořit informační systém pro podporu podnikových procesů oddělení nákupu společnosti Magna Exteriors & Interiors Bohemia s.r.o. Práce seznamuje čtenáře se společností a jejími procesy, s principy

vývoje webových aplikací a s použitými technologiemi. Popisuje celý vývoj od provedení úvodní studie a analýzy, přes návrh a implementaci ve frameworku ASP.NET MVC, až po testování a zhodnocení. Výsledkem je funkční aplikace používaná zaměstnanci společnosti.

Klíčová slova Magna Exteriors & Interiors, .NET framework, ASP.NET MVC, Transact SQL, jQuery

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Informační technologie a systémy jsou dnes již běžnou a mnohdy nepostradatelnou součástí podniků všech velikostí a zaměření. Díky vyspělým softwarovým aplikacím je možné ukládat velká množství dat, pohodlně na ně nahlížet, rychle v nich vyhledávat a efektivně je zpracovávat. Papír dnes již není jediným prostředkem k uchování důležitých informací. Společnosti zavádějí informační systémy umožňující jejich zaměstnancům odvést více práce v kratším čase a vyšší kvalitě. Špatný či zanedbaný návrh architektury podnikového informačního systému však může mít opačný efekt. Zaměstnanci jsou zahlceni množstvím složitých aplikací, data jsou špatně organizována, jejich dohledání je problematické a časově náročné a může dojít k jejich ztrátě či úniku. Jednou ze společností, která podobným problémům musí čelit je i Magna Exteriors & Interiors Bohemia s.r.o.

Společnost Magna Exteriors & Interiors Bohemia s.r.o. je významnou mezinárodní pobočkou společnosti Magna International Inc. Její hlavní sídlo a několik závodů se nachází v České republice. Další závody se nalézají v Rusku, Maďarsku a Německu. Společnost se zabývá návrhem a výrobou především plastových dílů a kompletů pro automobilový průmysl.

Oddělení nákupu společnosti Magna Exteriors & Interiors Bohemia s.r.o. zajišťující nákup dílů od externích dodavatelů se potýká s mnoha problémy, které pramení ze špatného využívání informačních technologií a prostředků. Společnost je certifikována podle několika standardů a má tak pevně definované všechny své procesy včetně formátu vstupních a výstupních dokumentů. V případě oddělení nákupu však tyto procesy nejsou dostatečně podpořeny informačními technologiemi a jejich možnostmi. Data a dokumenty v elektronické podobě, se kterými zaměstnanci denně pracují, nejsou nijak pevně organizovány. Neexistuje žádný informační systém, který by data shromažďoval a přehledně prezentoval. Vedení oddělení si je těchto ne-

dostatků vědomo, a proto žádá vytvoření a zavedení softwarové aplikace, která bude co nejlépe přizpůsobena běžné práci zaměstnanců oddělení a jejich požadavkům.

Cíle práce

Cílem této práce je navrhnout, implementovat a uvést do provozu zcela nový informační systém určený pro oddělení nákupu společnosti Magna Exteriors & Interiors Bohemia s.r.o. Návrhu systému bude předcházet důkladné seznámení se s procesy probíhajícími v oddělení, s pracovní náplní zaměstnanců a se stávajícími používanými informačními prostředky. Výsledný systém by měl být využíván jako hlavní pracovní nástroj zaměstnanců oddělení a měl by jim jejich práci usnadnit a zefektivnit. Systém by měl být nasazen co nejdříve, aby bylo možné pozorovat jeho přijetí uživateli a zhodnotit jeho přínos.

Struktura práce

Práce se skládá z následujících kapitol:

Úvodní studie: Představuje společnost Magna Exteriors & Interiors Bohemia s.r.o., popisuje oddělení nákupu, podnikové procesy, které v něm probíhají a informační technologie a prostředky, které jeho zaměstnanci využívají.

Analýza: Jedná se o popis první fáze vývoje aplikace, jejíž součástí je sestavení požadavků a tvorba modelu případů užití a konceptuálního modelu. V poslední části kapitoly jsou představeny technologie použité k implementaci aplikace.

Návrh: Součástí je návrh architektury systému, jednotlivých vrstev aplikace a uživatelského rozhraní.

Realizace: Obsahuje popis implementace důležitých částí a vrstev aplikace. Je zde zařazeno také nasazení aplikace na interní server společnosti.

Testování: Popisuje, jak probíhalo funkční a akceptační testování a testování uživatelského rozhraní.

Zhodnocení aplikace: Shrnuje přínos aplikace, dosavadní názory a připomínky uživatelů a uvádí, jak bude probíhat její další vývoj.

Kapitola 1

Úvodní studie

V Kapitole představuji společnost Magna Exteriors & Interiors Bohemia s.r.o. a především její oddělení nákupu. Popisuji zde hlavní podnikový proces a další procesy probíhající ve zmíněném oddělení. V poslední části kapitoly uvádím přehled využívaných ICT prostředků a jejich zhodnocení.

1.1 Společnost Magna International Inc.

Magna International Inc. [36] je celosvětový, a v Severní Americe zároveň největší, dodavatel komponentů a systémů pro automobilový průmysl. Hlavní sídlo společnosti se nachází ve městě Aurora v Kanadě. Další provozní skupiny společnosti Magna sídlí v USA, Mexiku, Číně, Japonsku a v několika evropských zemích včetně České republiky. Ve všech pobočkách pracuje přibližně 117 000 zaměstnanců.

Společnost Magna se zabývá návrhem, vývojem, výrobou a testováním dílů využívaných pro výrobu osobních i nákladních automobilů. Jako příklad lze uvést různé součásti karosérií i interiérů vozů, hnací, elektronické a střešní systémy, ale třeba i systémy pro hybridní vozy či elektromobily a podobně. Mezi její největší zákazníky patří v USA společnosti General Motors, Ford Motor Company a Chrysler LCC, v Evropě Volkswagen a BMW a v Asii Toyota.

1.2 Společnost Magna Exteriors & Interiors Bohemia s.r.o.

Společnost Magna Exteriors & Interiors Bohemia s.r.o. (Magna E&I) [34] je od května 2009 významnou pobočkou společnosti Magna International Inc. sídlící v České republice. Ředitelství společnosti, a zároveň jeden z výrobních závodů a nástrojárna, jsou umístěny v Severních Čechách v Liberci. Další výrobní závody lze nalézt v Libáni u Jičína a v Nymburku. Pod společnost Magna E&I spadají také společnosti Plastimat Hungary z Maďarska, ruský Technoplast a Meerane sídlící v Německu.

Společnost svůj výrobní program soustředí především na produkci plastových dílů a kompletů pro automobilový průmysl. Největší podíl výrobní kapacity zabírá výroba nárazníků, přístrojových desek (oboje přes 30 %) a dveřních výplní (20 %). Zákazníky společnosti tvoří evropští a asijští výrobci osobních a nákladních automobilů. Konkrétně se jedná například o Škodu Auto, TPCA v Kolíně, Volkswagen, Audi, Opel a Suzuki.

Součástí společnosti je vlastní vývojový tým. Díky němu je firma schopna na základě více či méně přesných požadavků zákazníka navrhovat, inovovat a testovat díly, které posléze také sama produkuje. Tým se nesoustředí pouze na návrh a úpravu samotných výrobků, ale i výrobních procesů, což by souhrnně mělo vést například k úspoře hmotností dílů a úspoře nákladů na straně zákazníka i společnosti Magna.

Všechny závody společnosti Magna E&I jsou certifikovány podle následujících standardů:

- **ISO 9001: Systém managementu kvality** [23] – Norma ISO 9001 má svému certifikovanému držiteli zajistit splnění požadavků jeho zákazníků a dalších zainteresovaných stran spolu s požadavky, které se vztahují k výrobkům. Toho je dosaženo nastavením, realizováním, měřením a monitorováním procesů, které pomáhají dosáhnout stanovených cílů a plánů společnosti v oblasti kvality její produkce. Zpětná vazba získaná měřením slouží k určení účinných opatření, které eliminují nedostatky v procesech. Obsah normy se týká principů řízení dokumentace, lidských zdrojů, infrastruktury, způsobu komunikace se zákazníky, hodnocení dodavatelů a interních auditů.

- **ISO/TS 16949: Management kvality v automobilovém průmyslu** [7] – Norma ISO/TS 16949 specifikuje požadavky na systém řízení kvality výrobců dílů pro automobilový průmysl. Vznikla doplněním normy ISO 9001 o požadavky týkající se automobilového průmyslu. Na jejím vývoji se podílela především mezinárodní pracovní

skupina pro sektor automobilového průmyslu (IATF). Klíčovým požadavkem verze normy z roku 2009 je naplnění požadavků zákazníka ze strany výrobce i jeho dodavatelů.

- **ISO 14001: Systém environmentálního managementu** [22, 8] – Norma ISO 14001 vznikla jako reakce na neustále se zvyšující nároky společností i zákazníků na systém ochrany životního prostředí a jeho efektivní realizaci v organizacích. Společnost implementující tuto normu si klade za cíl vytvořit a udržovat procesy napomáhající k naplnění plánů v oblasti emisí ze své produkce.
- **OHSAS 18001: Systém managementu bezpečnosti a ochrany zdraví při práci** [9] – Norma OHSAS 18001 svojí strukturou navazuje na normu ISO 9001 a ISO 14000. Vychází z analýzy rizik environmentálních aspektů a jejich minimalizace. Organizace certifikované podle této normy by měly mít zavedena opatření, která odstraní nebo alespoň minimalizují veškerá hrozící nebezpečí, nebo od nich alespoň své zaměstnance izolují. Pracovní činnosti by měly být naplánovány tak, aby jejich vykonávání bylo bezpečné a neohrožovalo zdraví.

Zavedení mezinárodně uznávaných standardů v oblastech systémů managementu by dle [23, 7, 22] mělo společnostem obecně přinést:

- Vysokou a stálou úroveň výrobního procesu a kvalitní poskytované služby a výrobky zákazníkům.
- Možnost snížení nákladů na provoz, nekvalitní výrobky, suroviny a energie a zvýšení zisku a spokojenosti zákazníků.
- Možnost oslovit, a větší pravděpodobnost získat, náročné a velké zákazníky.
- Zdokonalení systému řízení a organizační struktury společnosti.
- Zvýšení důvěryhodnosti u zákazníků, veřejnosti i státních orgánů.
- Možnost flexibilně reagovat na změny požadavků trhu a zákazníků, legislativy a strategie organizace.

Jednotlivá oddělení společnosti jsou organizována v liniové organizační struktuře. Struktura je zachycena na diagramu v příloze B. Součástí společnosti jsou standardní oddělení přítomná ve většině velkých firem. Jedná se o ekonomický, personální a IT odbor. Mezi další oddělení specifické pro výrobní a automobilové společnosti patří odbor prodeje, nákupu, centrální

logistiky, výrobního engineeringu a nových projektů, kvality a vývoje a oddělení systémů štíhlé výroby.

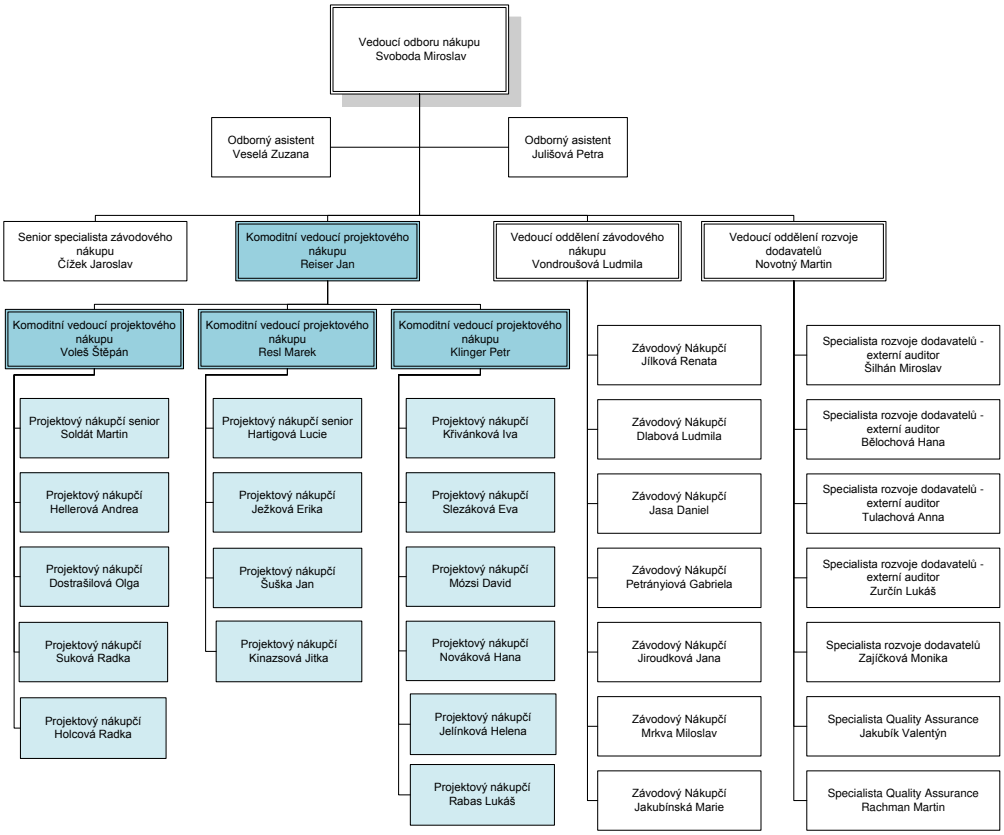
Společnost Magna E&I, a především její nákupní tým, klade důraz na rozvíjení dobrých vztahů s klíčovými dodavateli, aby si zaručila růst a stabilitu [35]. Dodavatelé jsou pro ni strategickou součástí všech zakázek a zároveň důležitým partnerem při vývoji nových výrobků.

1.3 Odbor nákupu

Odbor nebo také oddělení nákupu a jeho členové se velkou mírou podílejí na hlavním podnikovém procesu společnosti Magna Exteriors & Interiors Bohemia nazvaném Řízení projektu. Celý proces je popsán v [31] a pro další práci není nutné jej podrobně popisovat. Níže tedy pouze představím jeho hrubou strukturu a uvedu, jakou roli v něm sehrává oddělení nákupu.

Organizační struktura oddělení nákupu je zachycena na obrázku 1.1. V čele oddělení stojí vedoucí, který je v liniové organizační struktuře společnosti přímo podřízen finančnímu řediteli. Oddělení je rozdělené na tři samostatně fungující pododdělení, jejichž pracovní náplň se různí. Jedná se o projektový nákup, závodový nákup a rozvoj dodavatelů. Zadavatelem této práce je komoditní vedoucí projektového nákupu, a její výsledek by proto měl sloužit především členům tohoto pododdělení. Vedoucímu projektového nákupu jsou podřízeni tři vedoucí komoditních týmů, které se každý skládají ze čtyř až šesti projektových nákupčí. Jak název napovídá, každý komoditní tým zajišťuje nákup určitých, jemu přidělených komodit. Komoditou může být například plast, lak, spojovací materiál, textilie, těsnění, kovy, kabeláž, elektronika, světla a podobně. Vedoucímu pododdělení závodového nákupu je podřízeno osm závodových nákupčí. Pododdělení rozvoje dodavatelů se skládá z vedoucího, pěti specialistů rozvoje dodavatelů a dvou specialistů zajištění kvality. V prvním čtvrtletí 2013 bylo v odboru nákupu zaměstnáno celkem 39 osob, z toho 19 v pododdělení projektového nákupu.

Oddělení projektového nákupu (OPN) je pověřeno obstaráním dodavatelů pro díly, které se společnost rozhodne z ekonomických či jiných důvodů nevyrábět, ale nakupovat. Práce zaměstnanců oddělení se skládá především z komunikace s potenciálními i stávajícími dodavateli a shromažďováním potřebných dokumentů a smluv uzavíraných mezi nimi a společností Magna. Nejdůležitějším cílem je, aby byl vybraný dodavatel seznámen s požadavky zákazníka a byl schopen je plnit. Tento cíl je obecně pro oddělení nákupu stanoven i v normě ISO/TS 16949.



Obrázek 1.1: Organizační struktura oddělení nákupu společnosti Magna E&I

1.3.1 Hlavní proces Řízení projektu

Řízení projektu je hlavní podnikový proces společnosti Magna E&I. Cílem projektu je připravit zákazníkem poptávaný díl nebo komplet pro sériovou výrobu. Díl je považován za připravený, a projekt tedy končí, až v okamžiku, kdy jsou odladěny veškeré technologické a konstrukční detaily dílu. Musí být uzavřeny smlouvy s dodavateli dílčích částí dílu a dojednány veškeré náležitosti jako obaly a logistika mezi podnikem a dodavateli a mezi podnikem a zákazníkem. Díl je před uvedením do sériové výroby také samozřejmě řádně prověřen, otestován a odsouhlasen zákazníkem.

Za organizaci projektu a plnění kvalitativních cílů a termínů zodpovídá vedoucí projektu. Řídící strukturu projektového týmu doplňují vedoucí dílčích projektů. Jedná se o zástupce z oddělení, jejichž účast je na projektu vyžadována, a kteří stojí v čele dílčích projektových týmů z daných oddělení. Tyto dílčí týmy se řídí vlastními stanovenými procesy. Různé projekty

vyžadují podle svého rozsahu a náplně zapojení různých oddělení, a proto konkrétní složení projektových týmů může být s každým projektem odlišné. Jeden z dílčích projektových týmů může být složen ze zástupců OPN. Stálým členem projektového týmu je funkce customer assurance, která zajistí, že požadavky zákazníka budou zohledněny. Mezi další povinné členy patří pracovníci z oblasti životního prostředí, požární ochrany a bezpečnosti a ochrany zdraví při práci. Členové projektového týmu se pravidelně scházejí, aby kontrolovali průběh práce na projektu a plnění stanovených termínů.

Proces Řízení projektu začíná získáním nové zakázky, která je výstupem procesu Poptávkové řízení. V průběhu tohoto předcházejícího procesu je sestavena a zákazníkovi předložena závazná nabídka na dodávku výrobku. Je zde oceněn nabízený výrobní proces, vypracována studie vyrobitelnosti a vytvořen a potvrzen nabídkový výrobní koncept. V prvním kroku procesu Řízení projektu je zpracován záměr projektu a jmenován vedoucí projektu. Následuje sestavení projektového týmu, jmenování vedoucích dílčích projektů a vypracování termínového plánu. Termínový plán musí zohledňovat všechny milníky stanovené a vyžadované zákazníkem. Vedoucí dílčích projektů zodpovídají za to, že termíny jejich projektů budou zkoordinovány s hlavním termínovým plánem. V dalších krocích procesu jsou vedoucí dílčích projektů i členové jejich týmů proškoleni ve specifických požadavcích zákazníka a je vypracován rozpočet projektu. Následuje dlouhodobá práce na projektu, která je proložena šesti přezkoumáními projektu. Každé přezkoumání má svou přesně vymezenou dobu trvání, která je závislá na jiných milnících z termínového plánu. Přezkoumání slouží k tomu, aby byla zajištěna požadovaná úroveň rozpracování projektu před jeho uvolněním do další fáze. Po posledním šestém přezkoumání je projekt ukončen. Celý proces je podrobně popsán v [31]. Vývojový diagram procesu je uveden v příloze C.

Po celou dobu trvání projektu je práce na něm monitorována a měřena, aby bylo zaručeno dosažení požadované kvalitativní úrovně a dodržování termínů. Průběžně se shromažďuje a uchovává předepsaná projektová dokumentace v papírové i elektronické podobě. Během práce na projektu, a především po jeho ukončení, jsou vytvářeny reporty zachycující silné i slabé stránky projektu a obsahující doporučení pro budoucí projekty a návrhy na zlepšení procesů.

Do procesu Řízení projektu vstupuje interní objednávka vývoje výrobku nebo technické přípravy výroby s předepsanými přílohami, nominační dopis zákazníka, technická specifikace výrobku, základní projektové milníky zákazníka a platná nabídka zákazníkovi. Technická specifikace zákazníkem požadovaného výrobku může být dodána v různém stupni rozpracovanosti a přesnosti. Může se například jednat o velice přesný výkres dílu včetně všech předepsaných rozměrů a materiálů. V jiném extrémním případě však může

být zákazníkem dodán pouze hrubý náčrt poptávaného výrobku. Společnost Magna následně zapojením svého vývojového oddělení požadavek rozpracovává a upřesňuje, dokud zákazník není zcela spokojen.

V prvním čtvrtletí 2013 společnost Magna E&I pracovala na sedmnácti projektech spadajících pod pobočky v České republice, přičemž nejstarší z těchto projektů byl zahájen v roce 2009 a nejnovější v roce 2012. Dalších devět projektů náleželo pobočkám v Rusku. Průměrná doba trvání jednoho projektu jsou dle zaměstnanců OPN dva roky. Na jeden projekt připadá průměrně dvacet dílčích částí připravovaného výrobku.

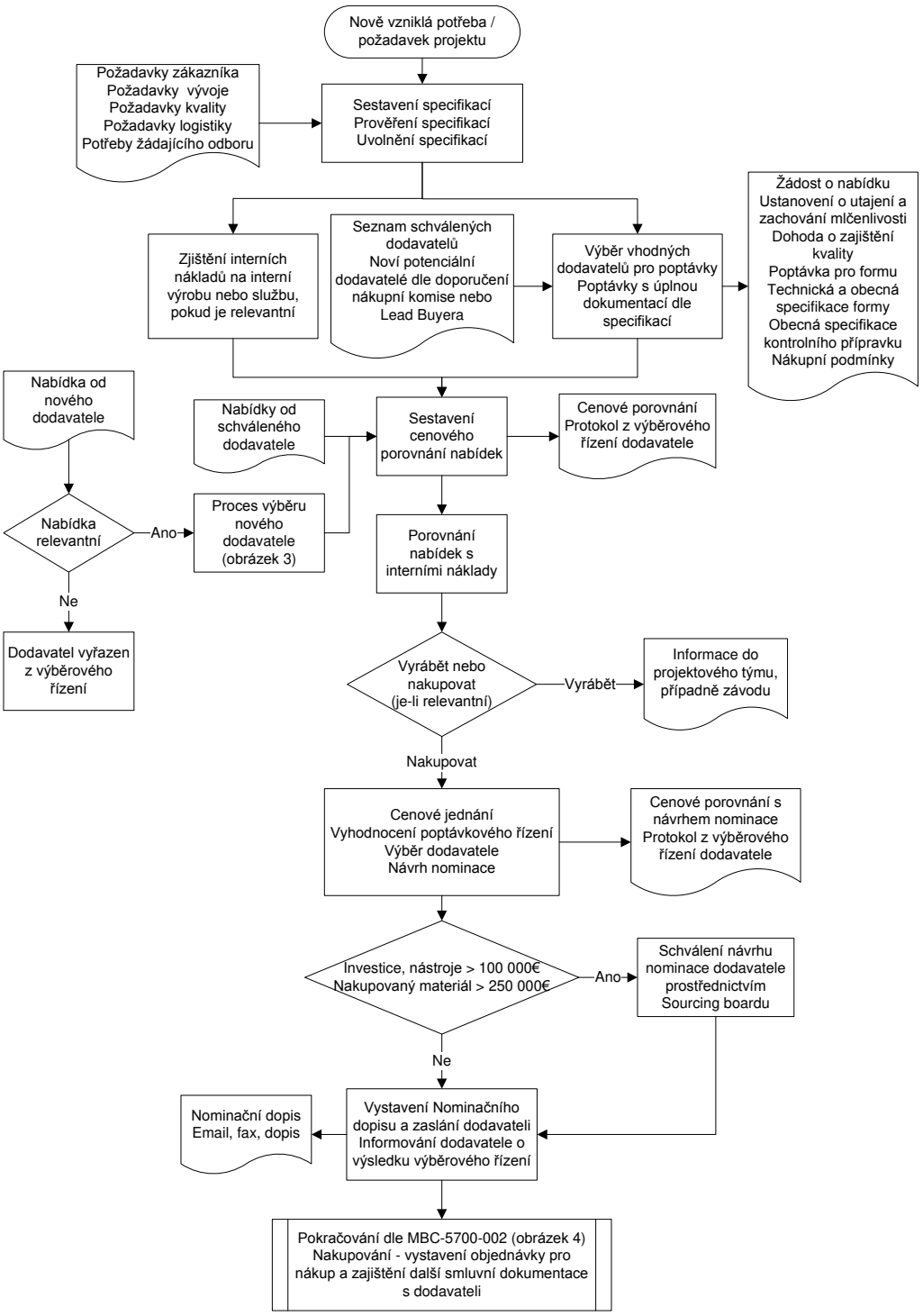
1.3.2 Procesy oddělení projektového nákupu

Pracovní činnost OPN se řídí především dvěma předepsanými procesy nazvanými Nakupování – Specifikace a poptávky, výběr dodavatele a Nakupování – vystavení objednávky pro nákup a zajištění další smluvní dokumentace s dodavateli. Druhý jmenovaný proces přímo navazuje na ten první. Procesy jsou vykonávány v rámci hlavního procesu Řízení projektu. Vykonávacím subjektem je dílčí projektový tým složený z vedoucího dílčího projektu a dalších vybraných osob z řad zaměstnanců OPN. Společným cílem obou procesů je vybrat vhodné dodavatele nakupovaných dílů, zajistit předání požadovaných dokumentů mezi nimi a společností Magna a přenést na ně požadavky zákazníka. Pro sepsání následujícího popisu procesů jsem použil informace z [32] a [33] doplněné o komentáře a vlastní zkušenosti zaměstnanců OPN.

Nakupování – Specifikace a poptávky, výběr dodavatele

Do prvního procesu vstupují požadavky zákazníka, požadavky interních oddělení (vývoj, kvalita, logistika), požadavky vyplývající z potřeb projektu a nabídky dodavatelů. Jeho náplní je výběrové řízení na nejvhodnějšího dodavatele nakupovaného dílů. Proces končí nominováním zvoleného dodavatele. Vývojový diagram procesu je zachycen na obrázku 1.2. Diagram byl převzat z dokumentu interního nařízení společnosti Magna E&I.

Samotnému procesu musí vždy předcházet několik kroků, které jsou součástí hlavního procesu Řízení projektu. Poté co podnik od zákazníka získá novou zakázku, je založen nový projekt a sestaven projektový tým. Pokud poptávaný výrobek vyžaduje nakoupení některých dílů, je členem projektového týmu i zástupce OPN. Stává se takzvaným vedoucím dílčího projektu (VDP) a jeho prvním úkolem je důkladné seznámení se s projektem a především poptávaným výrobkem (často se označuje jako sestava), jednotlivými díly, ze kterých se skládá, upřesňujícími požadavky zákazníka a termíny.



Obrázek 1.2: Vývojový diagram procesu Nakupování – Specifikace a poptávky, výběr dodavatele

VDP sestaví svůj projektový tým složený z nákupčích, což jsou výhradně zaměstnanci OPN a přidělí jim zodpovědnosti za jednotlivé díly.

Společnost Magna E&I používá pro plánování výroby a přehled zdrojů ERP[1] systém SAP. Každému dílu je proto třeba přidělit unikátní identifikační číslo, pod kterým bude díl do systému zanesen. Přidělování čísel probíhá následovně. Každý nákupčí si ze šanonu sloužícího k rezervaci čísel přečte poslední využité číslo a zapíše nové číslo, které vznikne navýšením původního o počet jemu přidělených dílů. Čísla dílů mají vždy tvar 8 XXX XXX, kde X je libovolná číslice. Následně pro každý díl nákupčí vyplní a vytiskne formulář Kmenová věta materiálu (viz příloha D). Formulář je k dispozici v elektronické podobě jako soubor pro kancelářskou aplikaci Excel. Musí být vyplněn v souladu s interním podnikovým nařízením. Vytištěné formuláře nákupčí předá pověřenému referentovi v OPN, který podle nich zadá díly do systému SAP.

V první fázi výběru dodavatele je nejprve specifikována poptávka na nakupované díly. Jsou shromážděny veškeré údaje o dílech. Výkresy a podrobná data většinou dodává oddělení konstrukce společnosti Magna E&I, v některých případech poskytne přesné specifikace zákazník. Upřesněno je také množství dílů vyrobených za rok, barevné varianty, materiál a podobně.

Pověření nákupčí následně sestavují seznam vhodných dodavatelů. Každý nákupčí má obecně v OPN na starosti jednu nebo více komodit. Zodpovědnosti za díly v dílčích projektech jsou rozřazovány právě na základě příslušnosti nákupčích k daným komoditám. Nákupčí tak nemusejí znát všechny dodavatele, se kterými Magna E&I spolupracuje. Jejich znalosti a zkušenosti s konkrétními dodavateli proto mohou být na vyšší úrovni. Nákupčí tedy ze seznamu schválených dodavatelů podle svého nejlepšího uvážení vyberou nejvhodnější kandidáty, zašlou jim předepsané dokumenty a požádají je o vypracování cenové nabídky.

Podnik také může obdržet nabídky od nových dodavatelů, případně mohou být dosud neschválení dodavatelé nominováni zákazníkem. Dodavatelé, jejichž nabídky nejsou relevantní, jsou automaticky vyřazeni z výběrového řízení. Ostatní musejí podstoupit a projít následujícím vedlejším schvalovacím procesem (obrázek 1.3). Nejprve je ověřeno, zda potenciální dodavatel vlastní certifikát na některou z norem ISO/TS 16949, ISO 14001, ISO 9001, nebo zda má zavedený jiný systém řízení. Dodavatelé nesplňující tuto podmínku jsou vyřazeni. Postupující dodavatele následně ověří auditor společnosti Magna E&I v místě jejich působení. Kontroluje se způsobilost dodavatele k sérovým dodávkám zjišťováním, zda dodavatel má například dostatečné výrobní kapacity, vlastní měrové středisko, či dostatek perso-

[1]ERP – Enterprise Resource Planning

nálu. Dodavatelům, kteří výsledky měření neuspokojují, může být udělena výjimka nebo nabídnuta možnost provedení nápravných opatření. Dodavatelé s uspokojivými výsledky jsou zařazeni do seznamu schválených dodavatelů. V konečném kroku vedlejšího procesu výběru nového dodavatele je mezi společností Magna E&I a dodavateli uzavřeno několik smluv. Jedná se o Rámcovou smlouvu, Logistický protokol a Nákupní podmínky.

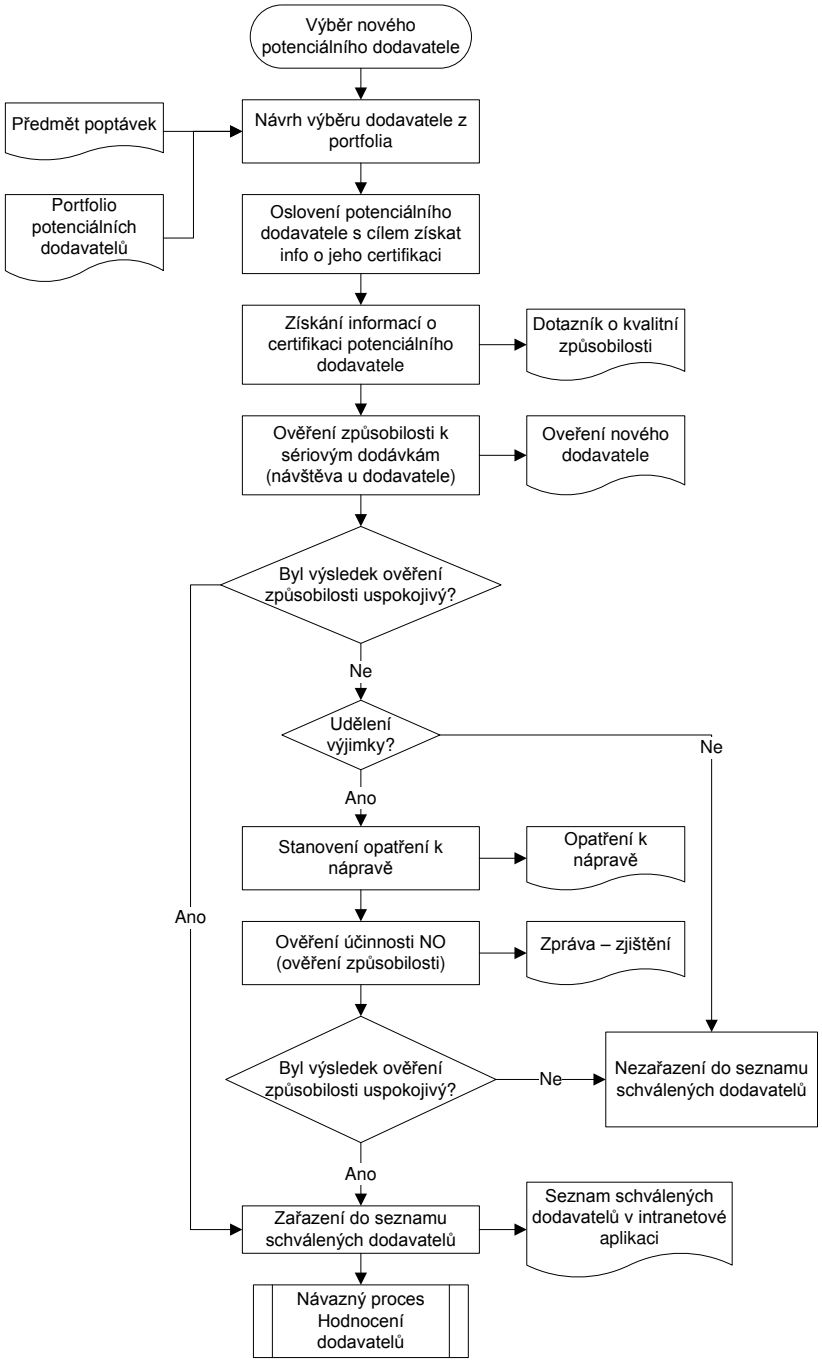
Dodavatelé na základě obdržené poptávky vypracují cenovou nabídku a odešlou ji zpět. Společnost Magna E&I mezitím provede zjištění interních nákladů na interní výrobu, pokud tato připadá v úvahu. Cenové nabídky jsou nejprve porovnány s výší interních nákladů na výrobu a je zvolen vhodnější přístup. Rozhodne-li se společnost díl nakupovat, jsou nabídky dodavatelů porovnány mezi sebou. Vhodný dodavatel je obvykle vybrán na základě nejnižší ceny, platebních podmínek a schopnostech plnit požadované termíny. Vybranému dodavateli je zaslán nominační dopis, který mu oznamuje, že byl vybrán na dodávku daného dílu, v daném období a počtu kusů a za danou cenu. Proces končí poté, co dodavatel potvrdí svou nominaci.

Nakupování – vystavení objednávky pro nákup a zajištění další smluvní dokumentace s dodavateli

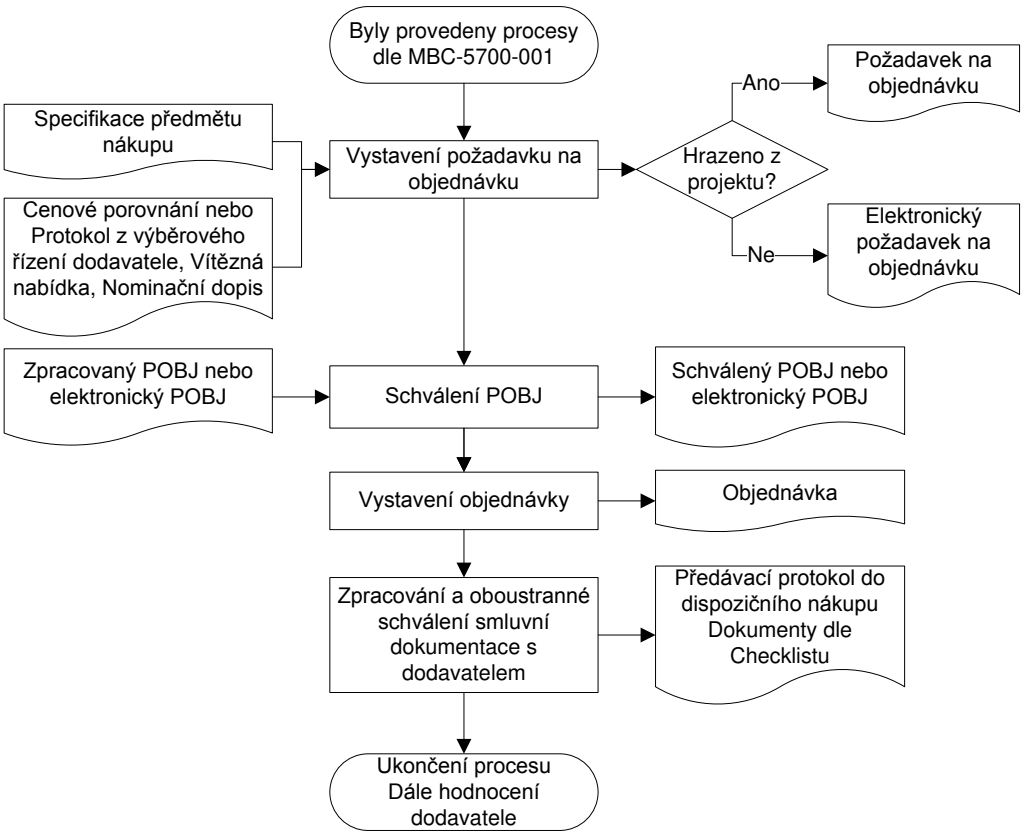
Mezi vstupy do druhého procesu patří cenové porovnání z předchozího procesu, protokol z výběrového řízení dodavatele, vítězná nabídka, specifikace předmětu nákupu a nominační dopis, vše v předepsaných formátech. Účelem procesu je vystavení objednávky pro nákup a uzavření smluv s dodavatelem, které jsou nezbytné pro zajištění dodávky specifikovaných výrobků v požadovaném množství, čase, kvalitě a ceně. Vývojový diagram procesu ukazuje obrázek 1.4.

Proces vychází z obecného procesu nakupování a rozšiřuje jej o povinnou dokumentaci a další kroky, které musí pověření nákupčí provést. V prvním kroku je vystaven interní požadavek na objednávku, který se v tomto případě vyplňuje do předepsaného formuláře. Objednávka může být vystavena až poté, co je požadavek schválen. Dodavateli se zasílá objednávka na nakupované díly. V případě, že je dodavatel pověřen výrobou nástroje, tak také objednávka na tento nástroj. Proces nakupování končí, když je zpracována a oboustranně schválena smluvní dokumentace s dodavatelem. Práce OPN však pokračuje až téměř do ukončení hlavního podnikového projektu.

Společně s objednávkou je dodavateli zaslán termínový plán a dodavatel má povinnost potvrdit, že termíny zanesl do svého výrobního procesu. Termíny jsou rozloženy do celé délky trvání projektu a reflektují stav rozpracovanosti poptávaného dílu. Jedná se například o termíny První výpadové



Obrázek 1.3: Vývojový diagram procesu Nakupování – Specifikace a poptávky, výběr dodavatele – část Výběr dodavatele



Obrázek 1.4: Vývojový diagram procesu Nakupování – vystavení objednávky pro nákup a zajištění další smluvní dokumentace s dodavateli

kusy, První vzorkování NOTE 1 (uvolněno) či První vzorkován NOTE 3 (uvolněno s podmínkou). V momentě, kdy je společnost Magna E&I spokojená s podobou a kvalitou dílu, dojde k jeho uvolnění. Návštěvou auditora je ověřena připravenost výrobních procesů dodavatele a plnění kapacit, což odpovídá termínu Audit 2DP (dvoudenní produkce).

Dalším důležitým milníkem je takzvané Zahájení předsériových dodávek, zkráceně PVS. Termín vyžaduje připravenost a odsouhlasení balících předpisů a plánů dodávek. Do systému SAP jsou vloženy doplňující informace o díle, jeho obalech, počtu dodávaných kusů a dodavateli. Je vytvořen předávací protokol. Povinností nákupčích je zajistit dodání první předsériové dodávky dílů do skladu společnosti Magna E&I. I když se jedná o předsériovou dodávku, vše probíhá tak, jakoby se jednalo o klasickou sériovou. Tento krok slouží k poslednímu ověření připravenosti obou stran na ostrý výrobní provoz. Po naskladnění dodávky a vložení příslušných dat do sys-

tému SAP podepíše referent dispozičního nákupu předávací protokol, čímž oficiálně převezme od nákupčích zodpovědnost za objednávky a dodávky dílů.

Posledním úkolem nákupčích je sepsání Přílohy rámcové smlouvy s dodavatelem, kde jsou zrekapitulovány ceny, doba platnosti smlouvy, množství dodávaných dílů a další podmínky. Pokud je dodáván i nástroj, musí být podepsána Smlouva o výpůjčce nástrojů, protože nástroje jsou ve skutečnosti majetkem společnosti Magna E&I nebo jejího zákazníka.

Projektová dokumentace

Jak vyplývá z popisu procesů, dochází v jejich průběhu k výměně několika dokumentů mezi společností Magna E&I a potenciálním či nominovaným dodavatelem. Jejich přehled je uveden v tabulce 1.1. Zdrojový formát dokumentů je v případě hotových smluv nejčastěji PDF. Dokumenty vyžadující vyplnění jednou nebo oběma stranami jsou vytvářeny v aplikacích MS Word nebo MS Excel. Vyplněné dokumenty jsou vytištěny, podepsány dodavatelem i společností Magna E&I, naskenovány zpět do elektronické podoby a uchovány ve formátu PDF.

Tabulka 1.1: Přehled projektové dokumentace

Název dokumentu	**Účel**
Balící předpis	Předepisuje například typ, váhu a rozměry balení.
Cenové porovnání	Dokumentuje rozhodnutí o výběru dodavatele z hlediska porovnání nabídnutých cen. Obsahuje cenovou nabídku od každého poptávaného dodavatele.
Dohoda o zajištění kvality	Definuje požadavky na zkoušky a kontrolu výrobků, které provádí dodavatel. Uvádí, které parametry a funkce výrobků v sérii mají být testovány.
Dotazník o kvalitativní způsobilosti dodavatele	Při výběru nového dodavatele slouží k ověření, zda je dodavatel certifikovaný nebo zda má zaveden jiný systém řízení.
Katalog požadavků na nakupovaný díl	Obsahuje všeobecné požadavky na konkrétní nakupovaný díl, dokumentaci, termíny, konstrukci dílu, jeho kvalitu a požadavky na logistiku.
Logistický protokol	Jedná se o smlouvu upřesňující způsob dodávání, odpovědnosti za obaly a převoz, odběrová místa a podobně. Vztahuje se vždy k určitému podnikovému závodu společnosti Magna E&I.
Nákupní podmínky	Musejí být akceptovány dodavatelem, aby vůbec existovala možnost spolupráce. Mohou být k dispozici již z předchozích projektů nebo odsouhlaseny až v době poptávky.

Nominační dopis	Informuje dodavatele, že byl nominován na výrobu a dodání určitých dílů v dohodnutém počtu kusů, v daných termínech a za sjednané ceny.
Ověření nového dodavatele	Dokument obsahuje sérii bodů, podle kterých auditor ověří, že dodavatel splňuje požadavky společnosti Magna E&I.
Plán zkoušek	Definuje požadavky na zkoušky a kontrolu výrobků, které provádí dodavatel. Předepisuje kritéria, která se mají testovat u náhodně vybrané skupiny dílů. Zkoušky se provádí pravidelně ve stanovených časových úsecích.
První vzorkování	Soubor dokumentace, kterou dodavatel předkládá spolu s několika referenčními vzorky zákazníkovi ke schválení. Dokumentace i vzorky jsou ověřeny, zda odpovídají požadavkům nadefinovaným společností Magna E&I.
Předávací protokol na logistiku	Dokument, na základě kterého je předána odpovědnost z OPN na logistiku za objednávání dílů.
Příloha rámcové smlouvy	Upřesňuje Rámcovou smlouvu. Představuje její konečnou platnou podobu.
Rámcová smlouva	Představuje všeobecnou smlouvu definující způsob, místo a dobu trvání dodávek a další platební či dodací podmínky.
Rozpad vítězné nabídky	Obsahuje podrobný cenový rozpad dílu na jednotlivé položky.
Smlouva o výpůjčce nástrojů	Je podepisována, pokud je součástí objednávky mimo dílů i nástroj. Nástroj je majetkem společnosti Magna E&I nebo jejího zákazníka.
Ustanovení o utajení a zachování mlčenlivosti	Podepsáním dokumentu, který je nedílnou součástí poptávky, se dodavatel zavazuje ke střežení tajemství a k nesdílení důvěrných informací se třetí stranou.
Životopis nakupovaného dílu	Je dodavatelem zasílán s každou novou ukázkou rozpracovaného dílu v průběhu jeho optimalizace. Popisuje vždy aktuální stav a podobu dílu.

Checklist

Po celou dobu trvání projektu nákupčí zaznamenávají stav rozpracovanosti dokumentace do Checklistu, který je průběžně kontrolován vedoucím dílčího projektu. Jedná se o soubor pro kancelářskou aplikaci Excel. Checklist má podobu tabulky, kde jsou v řádcích uvedeny jednotlivé díly a jejich dodavatelé a ve sloupcích vyžadované dokumenty, smlouvy další náležitosti vztahující se k dílu rozdělené do kategorií. Do buněk tabulky nákupčí vyplňují procentuální postup rozpracovanosti konkrétní položky nebo jinou hodnotu – například v případě sloupce „Způsob výběru dodavatele“. Hodnota 50 %

znamená, že dokument byl dodavateli odeslán. Na hodnotu 100 % jsou vyplněny položky, kde byla dokumentace odsouhlasena oběma stranami. Na konci projektu by každá buňka, která to umožňuje, měla obsahovat hodnotu 100 % nebo „Nerelevantní“ s vysvětlením, proč položka v tomto případě není relevantní. Vyplněný Checklist ukazue, že veškerá dokumentace je shromážděna a projekt je připraven na předání do sériového nákupu. Ukázka Checklistu je uvedena v příloze E.

Přehled vývoje cen
Každá úprava dílu v průběhu optimalizační fáze projektu často přináší také změnu ceny dílu. Vývoj cen dílů se zaznamenává do předepsaného souboru pro aplikaci Excel. Dokument má opět podobu tabulky – v řádcích jsou uvedeny jednotlivé díly a do sloupců se zaznamenávají změny cen dílů. Ke každé změně se vyplňuje její číslo, důvod, původce, datum platnosti a nová cena.

Kusovník
Kusovník obsahuje seznam jednotlivých dílů, ze kterých se skládá nějaký zákazníkem objednaný výrobek. Na OPN přichází z oddělení konstrukce. Nevýhodu je, že dokument nemá žádný předepsaný formát a každý konstruktér tak zasílá Kusovník v jiné podobě. Nákupčí proto jeho obsah přepisují do vlastního dokumentu, který někdy také nazývají Přehled nakupovaných dílů. Oba dokumenty jsou nejčastěji ve formátu pro aplikaci Excel.

1.3.3 ICT prostředky využívané v oddělení projektového nákupu

Každý zaměstnanec OPN má k dispozici stolní počítač nebo notebook, který využívá pro svou každodenní práci. Všechny počítače jsou připojeny k internetu i vnitřní podnikové síti (intranetu). Interní nařízení a předpisy procesů společnosti Magna E&I vyžadují archivaci mnoha projektových dokumentů v tištěné podobě. Veškerá dokumentace včetně Checklistu je po ukončení práce OPN na dílčím projektu předána oddělení sériového nákupu rovněž v tištěné podobě. Po celou dobu trvání projektu jsou však dokumenty a další soubory uloženy a upravovány v elektronické podobě. Dále uvádím seznam aplikací a dalších ICT prostředků, které zaměstnanci OPN využívají.

Intranet
Společnost Magna E&I má vytvořenou rozsáhlou intranetovou síť, do které jsou zapojeny stolní počítače, notebooky a několik firemních serverů. Ser-

very se využívají k provozování některých aplikací a především k ukládání předepsaných dokumentů a dalších souborů. Stolní počítače jsou v případě OPN vybaveny operačním systémem Microsoft Windows verze XP nebo 7, programy z kancelářské sady Microsoft Office 2010 (výjimečně starší verze) a internetovým prohlížečem Microsoft Internet Explorer 8 na operačních systémech Windows XP a prohlížečem Microsoft Internet Explorer 9 na operačních systémech Windows 7.

Na takzvaném projektovém serveru je pro každý nový projekt vytvořen samostatný adresář. Slouží ke shromažďování veškeré projektové dokumentace po celou dobu trvání projektu. Přístup sem mají všechna oddělení, která se na projektu nějakým způsobem podílejí. OPN do vyhrazeného adresáře povinně ukládá vždy aktuální verzi Checklistu. Na obsah souboru mohou nahlížet členové všech oddělení, ale právo modifikace mají pouze zaměstnanci OPN. Na serveru je uložen také soubor aplikace Microsoft Project, který zaznamenává strukturu termínového plánu. Aplikace MS Project obecně slouží k plánování projektů, sledování termínů, přiřazování zdrojů a sledování jejich využití [38]. Termínový plán slouží celému závodu a není přizpůsoben potřebám OPN.

Na jednom ze serverů je pro OPN vyhrazen prostor pro ukládání dokumentů. Seznam těch nejdůležitějších je uveden v tabulce 1.1. Adresář však nemá pevně danou strukturu podadresářů a vyhledávání v něm je dle zaměstnanců OPN obtížné. Dokumenty jsou prvotně rozřazeny podle svého typu (například Rámcová smlouva či Životopis). Další řazení je náhodné a dokumenty se někdy ukládají podle projektů, jindy podle dodavatelů, či jsou umístěny zcela samostatně. Názvy adresářů a souborů nemají ani ustálenou formu a jsou voleny podle uvážení každého zaměstnance. Před několika lety se oddělení pokusilo do organizování dokumentace vnést řád. Navržené řešení se ale neujalo a po krátké době se zaměstnanci uchýlili zpět k neuspořádanému způsobu ukládání souborů. Byl vytvořen soubor pro aplikaci Excel, který sloužil jako rozcestník do jednotlivých adresářů. Umožňoval základní filtrování a vyhledávání. Nákupčí měli povinnost dokument správně pojmenovat a uložit do adresáře Nezařazená dokumentace. Jeho obsah pravidelně kontroloval pověřený administrátor a nové dokumenty rozřazoval do příslušných adresářů.

Některé pracovní dokumenty, například Kmenovou větu materiálu, si zaměstnanci ukládají na svých osobních počítačích. Pro sdělování informací kromě ústní komunikace využívají pouze email. Pracují výhradně s emailovým klientem Microsoft Outlook.

Quality Platform

Quality Platform (QPF) je webová komunikační platforma, která umožňuje

sledovat a řídit kompletní realizační řetězec [26]. Společnost Magna E&I ji využívá pro komunikaci, výměnu informací a sdílení termínového plánu s dodavatelem především v průběhu optimalizační fáze projektu. Mimo systém QPF probíhá komunikace s dodavateli prostřednictvím zasílání elektronické pošty. Aplikace byla vytvořena na zakázku pro pobočku Magna Steyr společností JAWA Management Server GmbH. Je postavena na platformě inforum (Internet Information Platform) [25].

ASTRAS e-RFx

Systém ASTRAS e-RFx slouží výhradně k vedení výběrového řízení dodavatelů. Dodavatelé mají možnost se do webové aplikace zaregistrovat a jejím prostřednictvím následně komunikovat se společností Magna E&I. Na začátku výběrového řízení nákupčí zvolí vhodné dodavatele, v systému založí nový projekt a dodavatele do něj přidá. Ti automaticky obdrží email s upozorněním a další komunikace již probíhá především přes tento systém. Nákupčí zde specifikují požadavky závodu a dodavatelé vyplňují své nabídky. Systém také umožňuje sdílení důležitých dokumentů. Zkratka e-RFx znamená Electronic Request For [x] (elektronický požadavek na [x]), kde x je možné nahradit za nabídku, ocenění, informaci, či soutěžní nabídku. Všechny se vztahují k situacím, kdy zákazník oslovuje potenciální dodavatele za účelem ohodnocení a porovnání. Jedná se o SRM (Supplier Relationship Management – řízení vztahů s dodavateli) modul kompletního řešení ASTRAS [1] od německé společnosti Allocation Network GmbH.

SAP

SAP [54] je oblíbený a celosvětově využívaný ERP systém. Nabízí rozsáhlou škálu funkčností, které pokrývají prakticky veškeré potřeby dnešních podniků. Je dodáván ve formě samostatných, ale navzájem integrovatelných modulů, mezi které patří například finanční účetnictví, controling, plánování výroby, podpora prodeje, řízení lidských zdrojů, management kvality a podobně. K dispozici jsou také řešení přizpůsobená určitým odvětvím průmyslu, včetně automobilového. Společnost Magna E&I jej využívá především k řízení výroby a sledování skladových zásob.

V OPN se systémem SAP pracuje pouze vybraný referent, který do něj vkládá nakupované díly a s nimi související data o dodavatelích a dodávkách. Do systému mají přístup i někteří další zaměstnanci oddělení, avšak mají oprávnění pouze nahlížet. Systém jim může sloužit ke kontrole stavu dodávek, dohledávání zboží a podobně. Nemají ale žádnou povinnost jej využívat.

Microsoft Office
Balík kancelářských aplikací Microsoft Office v dnešní době využívá většina podniků z celého světa. Společnost Magna E&I není v tomto ohledu výjimkou. Zaměstnanci OPN nejčastěji pracují s aplikacemi Word a Excel. Vytvářené soubory ukládají na svých osobních počítačích nebo na intranetové síti. V aplikaci Excel byly vytvořeny šablony pro Kmenovou větu materiálu, Checklist, Vývoj cen a Kusovník.

1.3.4 Zhodnocení ICT prostředků

Společnost Magna E&I má podle zaměstnanců OPN vyhovující programové vybavení pro komunikaci s potenciálními i nominovanými dodavateli. Systémy ASTRAS e-RFx a Quality Platform nejsou využívané pouze pobočkou Magna E&I, ale i dalšími, především evropskými, provozními skupinami společnosti Magna International Inc. Nelze tak předpokládat, že by aplikace mohly být v blízké budoucnosti nahrazeny jinými.

Za nedostačující se dají pokládat prostředky pro organizaci dokumentů. K jejich ukládání slouží pouze vyhrazené místo na podnikové síti. Adresářová hierarchie není vedením OPN pevně zadána, a proto se s přibývajícím množstvím dokumentů stává čím dál tím více nepřehlednou a vyhledávání se zpomaluje. Není jisté, že by zavedení pravidel pro vytváření a pojmenovávání adresářů bylo zaměstnanci dodržováno. Vhodným řešením by bylo zajistit rozřazování dokumentů programově. Lze například uvažovat o nějaké podobě systému pro správu dokumentů[2].

Zaměstnanci OPN některé dokumenty uchovávají na svých osobních počítačích. Soubory tak nejsou dostupné ve sdíleném prostoru na podnikové síti. Nákupčím, kteří potřebují nahlédnout na práci svých kolegů, to stěžuje práci či zcela zabraňuje v jejím dalším vykonávání. Soubor Checklistu může upravovat libovolný nákupčí. Před každou úpravou by měl soubor uzamknout, aby nedocházelo ke kolizním změnám v případě editace více uživateli najednou. Na uzamčení souboru nákupčí občas zapomínají a dochází tak k chybám či ztrátám dat.

Vedení OPN ani VDP nemají vhodné nástroje pro kontrolu práce nákupčích mimo Checklist. Neexistuje žádný sdílený kalendář upravený pro potřeby OPN, ve kterém by byly zaznamenány projektové termíny. Každý nákupčí nese vlastní zodpovědnost za jejich poznamenání a plnění a vedení tak nemá možnost jednoduše kontrolovat dodržování harmonogramu.

Kancelářská aplikace Excel slouží dobře pro potřeby vytváření a vyplňování šablon Checklistu, Přehledu vývoje cen a Kmenové věty materiálu.

[2]DMS – Document Management System

Nákupčí jsou však nespokojeni s nutností opakovaného vypisování parametrů dílů nebo dodavatelů. Tato vlastnost také zvyšuje pravděpodobnost výskytu chybných údajů či pouhých překlepů.

Kapitola 2

Analýza

V kapitole analýza se zaměřuji na požadavky kladené na systém, na sestavení uživatelských rolí, na rozbor případů užití a na výběr implementačních technologií.

2.1 Požadavky

Z předchozího zhodnocení programového vybavení a dalších IT technologií, které OPN využívá, vyplývá, že jejich úroveň je nedostatečná a mohla by být zlepšena. Vedení oddělení si uvědomilo příležitost ke zlepšení a možnému zefektivnění práce svých zaměstnanců a nedostatky se rozhodlo odstranit nasazením nové aplikace. Měla by sloužit jako jednotný výchozí nástroj pro řízení projektů, jejichž řešením je OPN pověřeno, a především k organizaci projektových dat a dokumentace.

Nová aplikace by měla obstarat automatické pojmenování projektových dokumentů a adresářů a zařazování dokumentů do příslušných adresářů. Musí také existovat způsob dokumenty snadno vkládat, vyhledávat a odstraňovat. Aplikace uživatelům poskytne možnost ukládat, prohlížet a měnit důležité údaje o projektech, termínech, dílech a dodavatelích. Bude možné vytvářet vzájemné vazby mezi projekty, termíny, díly, dodavateli a dokumenty tak, aby se uživatelé mohli snadno pohybovat v aplikačním prostředí a ihned pochopili souvislosti. Aplikace bude sloužit k vyplňování, uchovávání a kontrole Checklistu. Kmenovou větu materiálu, Přehled vývoje cen a Checklist bude možné exportovat do formátu vhodného pro tisk, který bude odpovídat předepsaným šablonám. Aplikaci bude možné využít pro připomínání blížících se termínů a ze strany vedení ke kontrole průběhu práce na projektech.

Hotová aplikace bude nasazena na jeden z interních firemních serverů. Pro uživatele bude dostupná přes webové rozhraní – bude se tedy jednat o webovou aplikaci. Aplikace bude zpočátku určená především pro zaměstnance a vedení OPN. Při vývoji však musí být brán ohled na její možné budoucí rozšíření nejen pro další pododdělení odboru nákupu, ale i pro jiná oddělení společnosti Magna E&I. Je velice pravděpodobné, že uživatelé z jiných oddělení budou k aplikaci přistupovat s rozdílným stupněm oprávnění. Budou mít například možnost pouze nahlížet na její vybrané části. Aplikace by měla být připravena na převod do jiné jazykové podoby než české. Po dokončení a odladění aplikace ji OPN plánuje nabídnout i zahraničním pobočkám společnosti Magna E&I.

OPN se rozhodlo pro vývoj nové aplikace zcela od základů. Důvodem pro toto rozhodnutí bylo především velké množství specifických požadavků, kterým nevyhovoval žádný komerční produkt. Předcházelo mu zvážení následujících alternativních řešení:

Nákup komerčního software a jeho přizpůsobení na míru
V dnešní době existuje velké množství typových aplikačních software (TASW), které jsou vyvíjeny s úmyslem naplnit společné požadavky co největšího množství zákazníků v daném odvětví s minimální potřebou customizace[3]. Naproti tomu individuální aplikační software (IASW) se využívá v podnicích s nestandardními procesy a potřebami. Vývoj IASW je obecně považován za časově i finančně náročnější. Rozsah aplikace, kterou vyžaduje OPN, však není zdaleka tak veliký, jako například v případě systémů pro správu účetnictví nebo plánování výroby. Lze proto oprávněně předpokládat, že náklady na vývoj vlastního řešení by nedosáhly výše nákladů vynaložených na zakoupení, přizpůsobení a nasazení nějakého TASW. Hlavním důvodem pro zamítnutí nákupu typového software je množství specifických požadavků OPN a plánů na budoucí rozvoj aplikace. Zakoupením komerčního řešení se zákazník stává závislým na poskytovateli software a zpracování požadavků na změnu může trvat velice dlouhou dobu. Detailněji o rozdílech mezi TASW a IASW pojednává Voříšek v [60].

Přizpůsobení open-source software
Podobnou, ale levnější alternativou k zakoupení TASW je výběr a přizpůsobení open-source software vlastními prostředky. Nevýhodou tohoto řešení je, že neexistuje záruka podpory ze strany vývojářů a dalšího rozvoje systému. Nepodařilo se mi také nalézt žádné řešení, které by splňovalo většinu

[3]Customizace – Dle [60] se jedná o proces přizpůsobení TASW podnikovým procesům a specifickým požadavkům zákazníka.

požadavků na výsledný produkt. Obecné požadavky naznačují, že se nebude jednat o plnohodnotný systém na řízení projektů, který se vyznačuje především definováním termínů, úkolů a přiřazením zodpovědností. Stejně tak se nebude jednat o plnohodnotný systém pro správu dokumentů (důvody viz dále). S budoucím rozšiřováním aplikace by také mohlo nastat, že by vybraný základní open-source systém nedokázal uspokojit nově vzniklé požadavky, se kterými se na začátku nepočítalo. To by mohlo mít kritický dopad na další vývoj systému. Možnost výběru nějakého produktu jako základu a jeho doplnění o další funkce jsem proto zamítl.

Rozšíření a přizpůsobení systému SAP o vlastní modul
Systém SAP umožňuje vývoj customizovaných modulů, které lze integrovat se stávající implementací systému v podniku [53]. IT oddělení společnosti Magna E&I v současné době pracuje na vývoji vlastního modulu pro jedno z oddělení podniku. Projekt měl však již na začátku roku 2013 několikaměsíční zpoždění a přitom byl stále daleko od dokončení. Vedení OPN tuto možnost zamítlo, protože nasazení nové aplikace bylo vyžadováno co nejdříve.

Nasazení systému pro správu dokumentů nebo systému pro správu podnikového obsahu
Jedná se o podobně zaměřené systémy pro správu obsahu a dokumentů, přičemž ECM[4] systémy obecně nabízejí širší funkčnost než DMS. Hlavní podstatou systémů je organizovat dokumenty a řídit jejich životní cyklus v kontextu procesů podniku. Dokumenty je možné třídit, přiřazovat jim atributy, uzamykat, verzovat, prohlížet, upravovat a zařazovat je do pracovních a schvalovacích oběhů – takzvaných workflow. ECM systémy navíc nabízejí funkce pro správu webového obsahu (tvorbu internetových a intranetových stránek), řízení podnikových procesů, správu kontaktů, vedení sdíleného kalendáře a podobně. Některé produkty zahrnují i vlastní API pro vývojáře. Díky němu je možné systémy doplnit o vlastní vyžadovanou funkčnost. Cena licencí těchto produktů pro komerční využití je však obecně velice vysoká. Pohybuje s v řádech desítek až stovek tisíců korun. Jak jsem již zmínil v analýze ICT prostředků OPN, dokumenty, se kterými aplikace bude pracovat, mají podobu podepsaných naskenovaných smluv s dodavatelem. Není proto nutné je v aplikaci dále upravovat. Převážná většina úprav bude provedena dodavatelem. Předávání dokumentů v elektronické podobě probíhá pouze mezi OPN a dodavateli, nikoliv mezi samotnými zaměstnanci oddělení, a slouží k tomu výše popsané aplikace Quality Platform a AST-

[4]ECM – Enterprise Content Management

RAS e-RFx. Nová aplikace tedy nemusí obsahovat podporu pro definici a řízení workflow. Jak je vidět, hlavní funkce ECM systémů by v plánované aplikaci zůstaly nevyužité. Investici do systému ECM nebo DMS jsem tedy v tomto případě posoudil jako zbytečnou a zamítl.

Plánovaná aplikace nemá být navržena tak, aby podporovala každý jednotlivý krok procesů oddělení. Svou funkčností by měla procesy podporovat jako celek. Tím bude dosaženo značného zjednodušení pro uživatele, protože jim odpadne nutnost podstupovat schvalovací procesy. Práce na každém projektu může vyžadovat odlišný přístup nákupčích. Údaje k dílům i projektová dokumentace mohou být přidávány v různém pořadí, a proto definování přesného postupu by ani nebylo možné.

2.1.1 Funkční požadavky

Funkční požadavky popisují požadované funkce systému [2]. Při jejich sestavování jsem vycházel z analýzy současného stavu ICT prostředků OPN a z několika konzultací se zaměstnanci oddělení. Doplnil jsem je dle vlastního uvážení. Funkční požadavky jsem se rozhodl rozdělit do tematických kategorií.

Uživatelské role a zabezpečení

F.01.01: *Systém bude zabezpečen proti přístupu neoprávněných osob:* Přístup do aplikace bude uživateli umožněn po zadání správné kombinace jména a hesla. Aplikace bude využívat systému integrované autentizace Windows[5]. Uživatelé se tedy budou přihlašovat stejnými údaji, kterými se přihlašují do intranetové podnikové sítě.

F.01.02: *Systém bude provádět autorizaci přihlášených uživatelů na základě přidělených rolí:* Uživatelům budou přiděleny role v systému. Na jejich základě budou oprávněni vykonávat konkrétní akce, nahlížet na vybrané části aplikace a zastávat určité pozice v projektovém týmu. Tyto role jsou specifické pro aplikaci a nemají na rozdíl od přihlašovacích údajů nic společného s oprávněními v intranetové síti.

F.01.03: *Systém umožní zakládání uživatelů vybraných rolí mimo administrační část:* Struktura projektového týmu vyžaduje, aby uživatelské účty s některými rolemi mohli být zakládány jinými uživateli s dostatečným oprávněním. Kompletní správa uživatelských účtů bude v kompetenci administrátora.

[5]IWA – Integrated Windows Authentication

Projekty

F.02.01: *Systém bude zahrnovat správu projektů:* Uživatelé v roli „Nákupčí“ budou moci vytvářet nové projekty a spravovat je. Projekt má své povinné a nepovinné vlastnosti a projektový tým, který se skládá z aktivních i neaktivních uživatelů systému v různých rolích. Uživatel, který projekt zakládá, se stává jeho vedoucím – přesněji vedoucím dílčího projektu (VDP).

F.02.02: *Systém bude rozlišovat oprávnění uživatelů na základě jejich příslušnosti k projektovému týmu:* Systém bude uživatelům udělovat oprávnění nejen na základě jejich rolí, ale i v závislosti na jejich pozicích v dílčích projektech.

Termíny

F.03.01: *Systém bude zahrnovat správu termínů:* VDP a zástupce VDP mohou vytvářet, upravovat a mazat termíny vztahující se k projektu. Termíny mohou být předdefinovaných i vlastních typů. Termín je možné zadat i bez konkrétního data a doplnit jej později.

F.03.02: *Systém zobrazí termíny v kalendáři:* Termíny bude možné zobrazit a upravovat pomocí měsíčního kalendáře.

F.03.03: *Systém bude upozorňovat na blížící se termíny:* VDP a zástupce VDP budou mít možnost nastavit upozornění na blížící se termín. Upozornění budou zasílána na email vybraným členům týmu zadaný počet dnů před proběhnutím termínu. Ke každému termínu může být přiřazen libovolný počet upozornění.

Dodavatelé

F.04.01: *Systém bude zahrnovat správu dodavatelů:* Bude možné vytvářet, upravovat a mazat dodavatele. Bude se jednat o stejné dodavatele, kteří jsou již uloženi v systému SAP. Sdílení dat mezi aplikacemi však nebylo povoleno v ani jednom směru, a proto musejí být duplicitní data uložena i v plánované aplikaci.

F.04.02: *Systém bude zahrnovat správu kontaktních osob dodavatele:* K dodavatelům bude možné přiřadit libovolný počet kontaktních osob.

F.04.03: *Systém bude zahrnovat správu zařízení umístěných u dodavatelů v majetku zákazníka:* K dodavatelům bude možné přiřazovat libovolný

počet zařízení pro výrobu či testování dílů jím dodávaných. U zařízení se bude evidovat jejich cena složená z neomezeného počtu položek.

F.04.04: *Systém umožní vyhledávání dodavatelů:* Systém nabídne rozhraní pro vyhledávání dodavatelů. Budou rozlišováni dodavatelé dodávající alespoň jeden díl v některém projektu, a ti co žádný nedodávají. Vyhledáváni budou podle názvu, SAP čísla a projektu.

F.04.05: *Systém umožní importování dodavatelů do databáze ze souboru s předdefinovanou strukturou:* Bude možné importovat seznam libovolného počtu dodavatelů zadaných v souboru formátu .CSV do databáze aplikace, což uspoří čas, který by musel být vynaložen na jejich manuální vložení. Soubor se seznamem dodavatelů může být například získán exportem ze systému SAP.

Díly

F.05.01: *Systém bude zahrnovat správu dílů:* Bude možné vytvářet, upravovat a mazat díly. Jedná se o dílčí části nějakého výrobku, který má být výsledkem projektu.

F.05.02: *Systém umožní přiřazení dílů k projektům:* VDP a zástupci VDP bude umožněno přiřazovat díly k projektům. Mohou přidávat nové díly nebo vybírat z existujících. Jeden díl může být přiřazen k libovolnému počtu projektů. V rámci projektu bude možné díly zařazovat do kategorií. Každému dílu musí mít být v rámci projektu přiřazen alespoň jeden zodpovědný nákupčí, který bude VDP a zástupcem VDP vybrán z členů projektového týmu. Pouze zodpovědný nákupčí, VDP a zástupce VDP mohou upravovat vlastnosti dílu.

F.05.03: *Systém umožní evidovat historii cen dílů:* U dílu bude možné měnit jeho cenu, přičemž bude zachována historie všech cen. Aktuální cena bude viditelně odlišena od ostatních neplatných.

F.05.04: *Systém umožní přiřazovat k dílu dodavatele a zařízení:* Ke každému dílu bude možné přiřadit jeho dodavatele a toto přiřazení měnit. Bude evidována historie změn dodavatelů. Zároveň může být k dílu přiřazeno libovolné množství zařízení. Lze vždy vybírat pouze ze zařízení, která se vztahují k aktuálně přiřazenému dodavateli.

F.05.05: *Systém umožní editaci základních vlastností více dílů najednou v rámci jednoho projektu:* Nákupčí bude mít možnost v rámci projektu vybrat více dílů, za které je zodpovědný, a souhrnně jim upravit požadované vlastnosti.

F.05.06: *Systém umožní vyhledávání dílů:* Systém nabídne rozhraní pro vyhledávání dílů. Budou rozlišovány díly zařazené do alespoň jednoho projektu a díly bez zařazení. Vyhledávat bude možné podle názvu, SAP čísla, dodavatele a projektu.

F.05.07: *Systém umožní importování nových dílů ze souboru s předdefinovanou strukturou:* VDP a zástupce VDP budou mít možnost importovat seznam libovolného počtu dosud neexistujících dílů zadaných v souboru formátu .CSV do databáze aplikace. Soubor bude vytvořen například úpravou kusovníku obdrženého z oddělení konstrukce.

F.05.08: *Systém bude automaticky generovat formulář Kmenová věta materiálu pro tisk:* Bude možné vygenerovat formulář Kmenová věta dílu pro účely tisku. Struktura formuláře bude odpovídat struktuře předepsané vedením oddělení uvedené v příloze D. Vybraná pole budou automaticky vyplněna údaji z dílu.

Dokumentace

F.06.01: *Systém umožní nahrávání dokumentů na podnikový server a zajistí jejich správnou organizaci:* Systém poskytne rozhraní pro nahrávání projektových dokumentů uvedených v tabulce 1.1. Bude rozlišovat v této tabulce uvedené typy dokumentů. Zajistí jejich vhodné pojmenování a uložení do příslušného adresáře ve vyhrazeném prostoru na podnikovém serveru. Systém také zajistí tvorbu adresářové struktury a mazání nepřiřazených souborů a prázdných adresářů. U některých typů dokumentů bude systém evidovat jejich historii.

F.06.02: *Systém umožní stažení nahraných dokumentů:* Nahrané dokumenty bude možné ze systému stáhnout do počítače či jiného zařízení a prohlédnout si je.

F.06.03: *Systém umožní přiřazovat dokumenty k dodavatelům, projektům, dílům a závodům:* Bude možné vytvářet závislosti mezi dokumenty, dodavateli, projekty, díly, závody a jejich kombinacemi. Přiřazení dokumentů bude možné zrušit. Při vytváření vztahu mezi dokumentem a dílem bude možné vybrat více dílů – dokument pak bude přiřazen ke všem vybraným dílům. K dílům bude možné přiřazovat již nahrané dokumenty, které mají vazbu na jiné díly.

F.06.04: *Systém umožní vyhledávání dokumentů:* Systém nabídne rozhraní pro vyhledávání dokumentů. Vyhledávat bude možné podle typu dokumentu, dodavatele projektu a dílu.

Checklist

F.07.01: *Systém bude zahrnovat správu Checklistu:* Systém bude dynamicky sestavovat Checklist pro každý projekt. Nákupčí budou moci měnit hodnoty položek Checklistu u dílů, za které zodpovídají. Hodnoty budou vybírat ze seznamu hodnot relevantních k dané položce. U některých hodnot může být vyžadováno zadání komentáře.

F.07.02: *Systém zajistí, aby některé hodnoty položek Checklistu mohly být vybrány až po nahrání příslušných dokumentů:* Většina položek souvisí s některým z projektových dokumentů. Aby u těchto položek mohla být vybrána hodnota 100 %, musí být příslušný podepsaný dokument nahrán v systému.

F.07.03: *Systém umožní stažení nahraných dokumentů přes rozhraní Checklistu:* Dokumenty související s položkami Checklistu bude možné odsud stáhnout.

F.07.04: *Systém bude automaticky generovat formulář Checklist pro tisk:* Bude možné vygenerovat formulář Checklist pro účely tisku. Struktura formuláře bude odpovídat struktuře předepsané vedením oddělení uvedené v příloze E.

Přehled vývoje cen

F.08.01: *Systém nabídne rozhraní pro přehled vývoje cen dílů v rámci projektu:* Uživatelům bude zpřístupněno rozhraní přebírající podobu formuláře Přehled vývoje cen. Rozhraní umožní provádění změn cen.

F.08.02: *Systém bude automaticky generovat formulář Přehled vývoje cen pro tisk:* Formulář Přehled vývoje cen bude možné nechat automaticky převést do podoby vhodné pro tisk.

Administrace

F.09.01: *Systém bude obsahovat administrační část aplikace:* Součástí systému bude administrační rozhraní, do kterého budou mít přístup uživatelé v roli „Administrátor“. Bude zde prováděna správa uživatelů, jazykových verzí, a všech entit, které spravování vyžadují.

Internacionalizace

F.10.01: *Systém bude připraven na rozšíření do další jazykové mutace:* Systém musí být naprogramován tak, aby jej bylo možné snadno rozšířit

o další jazykové mutace. Požadavek se netýká pouze uživatelského rozhraní, ale souvisí i s některými entitami, které jsou uloženy v databázi.

F.10.02: *Systém umožní přepínání jazyků uživatelského rozhraní za běhu:* Uživatelské rozhraní systému bude obsahovat prvky, které umožní přepnutí systému do jiného libovolného jazyka, jenž bude k dispozici. Jazyk bude možné změnit za běhu systému bez nutnosti jej restartovat.

F.10.03: *Systém umožní správu překladů z administračního rozhraní:* Jazykové verze i slova a slovní spojení pro překlad bude možné přidávat, upravovat a mazat z administračního rozhraní systému.

2.1.2 Nefunkční požadavky

Nefunkční požadavky formulují omezení kladená na systém nebo proces vývoje [2]. Vycházejí především z požadavků a podmínek kladených IT oddělením společnosti Magna E&I.

Typ aplikace: Systém bude vytvořen a používán v podobě webové aplikace.

Umístění: Systém bude nasazen a provozován na interních serverech poboček společnosti Magna E&I. Jako běhové prostředí bude sloužit webový server Microsoft Internet Information Services 7.5.

Omezení na klientská rozhraní: Systém může vyžadovat spouštění aplikace na internetových prohlížečích s podporou jazyka JavaScript a standardů HTML 5 a CSS 3.

Implementace: Systém bude implementován za použití .NET frameworku v jazyce C#.

Databáze: Systém bude veškerá data mimo dokumentů ukládat v databázi umístěné na podnikovém databázovém serveru Microsoft SQL Server 2005.

Přístup k datům: Systém bude k databázovým datům přistupovat, číst je, ukládat, měnit a mazat výhradně prostřednictvím uložených procedur.

Dostupnost a spolehlivost: Systém bude zaměstnancům OPN dostupný v jejich pracovní době s minimálními výpadky.

Údržba: Po nasazení a odladění systému bude jeho správa předána IT oddělení společnosti Magna E&I. IT oddělení zajistí také zapracování budoucích požadovaných změn systému.

2.2 Uživatelské role a oprávnění

Systém bude kontrolovat oprávnění uživatelů pohybovat se v něm a provádět specifické akce na základě dvou různých faktorů.

2.2.1 Kontrola oprávnění na základě rolí

V prvním případě budou práva uživatelů ověřována podle jim přidělených rolí, jejichž výčet bude pevně daný. Na základě role bude uživatel moci vstupovat do jednotlivých sekcí aplikace a také zastávat určitou pozici v projektovém týmu.

V první fázi vývoje a provozování aplikace požaduje OPN vytvoření pouze dvou sekcí. Jedna sekce bude přidělena zaměstnancům oddělení a bude uzpůsobena ke kompletní správě projektů. Měla by naplňovat všechny požadavky na systém uvedené v předchozí kapitole, kromě požadavků na administraci. Tyto zbývající požadavky budou zohledněny při implementaci druhé administrační sekce, která bude sloužit ke správě uživatelských účtů a vedlejších atributů.

Výčet uživatelských rolí, které mohou být uživateli přiděleny během zakládání jeho účtu je následující:

- **Administrátor:** má přístup do administrační sekce aplikace.
- **Člen OPN:** má přístup do sekce se správou projektů.
- **Privilegovaný člen OPN:** jedná se o člena OPN, který má v kontextu všech projektů stejná oprávnění jako jejich vedoucí (viz dále).
- **Vedoucí projektu:** v první fázi vývoje nemá do aplikace přístup.
- **Administrátor projektu:** v první fázi vývoje nemá do aplikace přístup.
- **SQA:** v první fázi vývoje nemá do aplikace přístup.
- **Logistik:** v první fázi vývoje nemá do aplikace přístup.
- **Kvalitář:** v první fázi vývoje nemá do aplikace přístup.
- **Konstruktér:** v první fázi vývoje nemá do aplikace přístup.

2.2.2 Kontrola oprávnění na základě pozice v projektovém týmu

V druhém případě slouží k ověření práv pozice konkrétního uživatele v projektovém týmu daného projektu. Oprávnění tohoto typu nejsou pro každého uživatele pevně stanovena, ale mění se s projektem, v jehož kontextu se uživatel pohybuje a provádí akce.

Projektový tým dílčího projektu, který zpracovává OPN, se skládá z vedoucího dílčího projektu (VDP), jeho zástupce a projektových nákupčí. OPN však vyžaduje, aby systém ukládal informace o dalších členech projektů, jejichž organizační příslušnost spadá pod jiná oddělení podniku.

Celkový seznam pozic v projektovém týmu s jejich oprávněními je uveden níže. Zmíněny jsou také uživatelské role, kterým může být pozice přidělena. Každou pozici musí zastávat alespoň jeden uživatel. Některé mohou být reprezentovány více uživateli.

- **VDP nákup:** vedoucí dílčího projektu v OPN. Stává se jím automaticky uživatel, který v systému zakládá projekt a nemůže být změněn. Má oprávnění měnit veškeré vlastnosti projektu a dílů přiřazených k projektu. Vybírá zbylé členy projektového týmu. Pozici mohou zastávat uživatelé v roli Člen OPN.

- **Zástupce VDP:** zástupce vedoucího dílčího projektu v OPN. Má stejná oprávnění jako VDP. Pozici mohou zastávat uživatelé v roli Člen OPN.

- **Projektový nákupčí:** v týmu se jich může vyskytovat několik. Jsou jim přiděleny zodpovědnosti za díly. Nákupčí mohou upravovat vlastnosti jen těch dílů, za které nesou zodpovědnost. Zároveň pouze k těmto dílům mohou nahrávat dokumentaci, přiřazovat dodavatele a měnit související položky v Checklistu. Pozici mohou zastávat uživatelé v roli Člen OPN.

- **Vedoucí projektu:** hlavní vedoucí celého projektu, který probíhá napříč většinou oddělení podniku. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli Vedoucí projektu.

- **Administrátor projektu:** vykonává pomocné činnosti v projektovém týmu jako vytváření a shromažďování dokumentace, organizování schůzek, připravování prezentačních materiálů a podobně. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli Administrátor projektu.

- **SQA**[6]**:** přenáší na dodavatele požadavky zákazníka a zajišťuje, že budou splněny. V týmu se jich může vyskytovat několik. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli SQA.

- **Logistika nakupovaných dílů:** zajišťuje průběh přepravy dílů od dodavatele do pobočky společnosti Magna E&I. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli Logistik.

- **VDP kvalita:** vedoucí dílčího projektu v oddělení kvality. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli Kvalitář.

- **VDP Konstrukce:** vedoucí dílčích projektů v oddělení konstrukce. V týmu se jich může vyskytovat několik. V první fázi vývoje aplikace nemá přidělena žádná oprávnění. Pozici mohou zastávat uživatelé v roli Konstruktér.

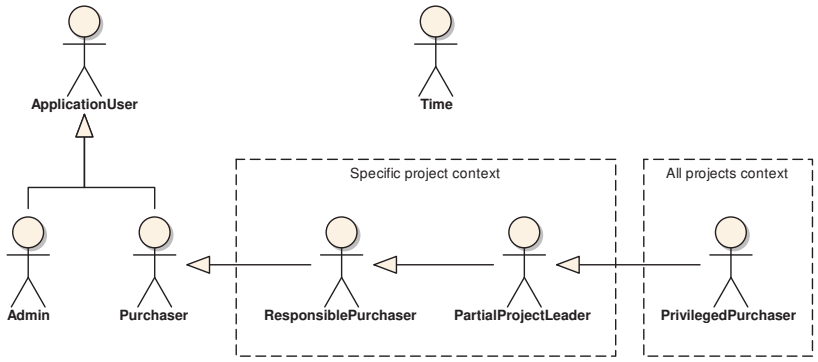
2.3 Model případů užití

Případy užití jsou způsobem zachycení požadavků na systém [2]. Rumbaugh [52] je definuje jako „specifikaci posloupnosti akcí, včetně proměnných a chybových posloupností, které může systém, podsystém nebo třída vykonávat prostřednictvím interakce s externími objekty (aktéry) za účelem poskytnutí služby přinášející nějakou hodnotu“. Představují ucelené jednotky funkčnosti, které systém poskytuje [52], a které jsou vždy vyvolány aktérem [2] zasláním zprávy systému.

Neexistuje žádná standardní formální specifikace struktury a obsahu modelu případů užití. Mnoho odborných prací se však touto problematikou zabývá a definuje vlastní přístup k vypracování modelů, které jsou si většinou hodně podobné. Při sestavování modelu jsem se řídil částí metodiky Unified Process popsané v [2]. Lehce odlišný způsob popisuje například [28].

Modelování případů užití podle metodiky Unified Process se skládá z iterativního opakování vyhledání hranic systému, nalezení aktérů a nalezení případů užití. Výsledný model pak obsahuje hranice systému, vnější aktéry, případy užití a vazby mezi aktéry a případy užití. Všechny části modelu bývají zachyceny na diagramech. Podrobněji model specifikují detaily případů užití, což jsou textové popisy obsahující nejčastěji název případu užití,

[6]SQA – Supplier Quality Assurance



Obrázek 2.1: Model případů užití – aktéři

jedinečný identifikátor, stručný popis, zapojené aktéry, vstupní podmínky, hlavní scénář, výstupní podmínky a alternativní scénáře [2].

V kapitole nejprve popisuji aktéry systému a následně případy užití opět rozdělené do tematických kategorií. Hranice systému jsem pro větší názornost v diagramech rozdělil na dvě oblasti. Jedná se o administrační sekci (Administration section) a sekci pro správu projektů (Project management section). Neuvádím scénáře všech případů užití, ale pouze některé vybrané, jež se odlišují od tradičních, jejichž postup je přímočarý.

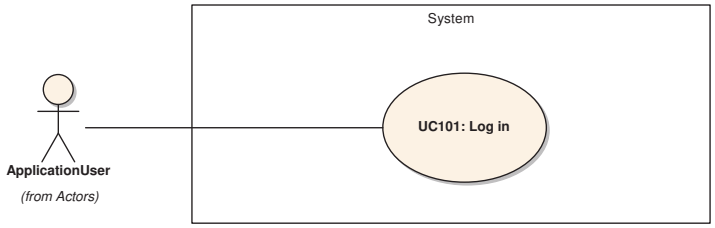
2.3.1 Aktéři

Aktéři vyjadřují role přidělené externím entitám, které přímo používají daný systém [2]. Může se jednat o obecné nebo konkrétní osoby, cizí podniky, jiné systémy a čas. Při hledání aktérů se zjišťuje, kdo nebo co systém používá či s ním komunikuje.

Diagram na obrázku 2.1 zachycuje aktéry, které jsem pro systém definoval. Jsou v něm obsaženi aktéři odvození od systémových rolí a aktéři představující pozice v projektovém týmu. Na několika místech je v diagramu použita vazba generalizace aktéra (znázorněna šipkou). Konkretizovaný aktér přejímá od obecného všechno jeho chování a dále jej může rozšiřovat o vlastní specifické interakce se systémem.

ApplicationUser (Uživatel aplikace): Jedná se o obecného abstraktního uživatele, který se může přihlásit do systému, ale nemá přístup do žádné ze sekcí. Všichni ostatní aktéři dědí jeho práva.

Admin (Administrátor): Reprezentuje stejně pojmenovanou uživatelskou roli. Má přístup do administrační sekce.



Obrázek 2.2: Model případů užití – obecné

Purchaser (Člen OPN): Představuje uživatelskou roli Člen OPN. Může přistupovat do sekce pro správu projektů. Je nadřazeným aktérem pro zbylé tři, jejichž schopnost interakce se různí v závislosti na projektu, v jehož kontextu akci vykonávají.

ResponsiblePurchaser (Zodpovědný projektový nákupčí): Aktéra Purchaser rozšiřuje o možnosti provádět aktivity související s konkrétními díly, za které nese zodpovědnost.

PartialProjectLeader (Vedoucí dílčího projektu OPN): Schopnosti aktéra ResponsiblePurchaser vztahuje na všechny díly v projektu a dále přidává aktivity související se správou projektu.

PrivilegedPurchaser (Privilegovaný člen OPN): Přebírá schopnosti aktéra PartialProjectLeader a aplikuje je na všechny projekty v systému.

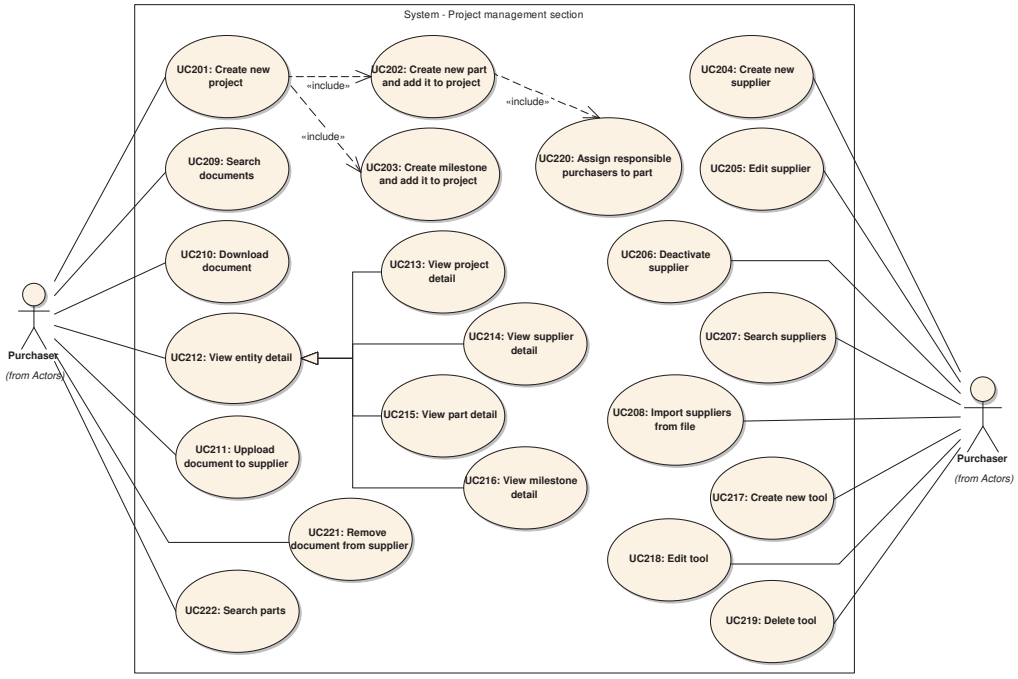
Time (Čas): Nepředstavuje žádnou konkrétní uživatelskou roli, pouze vyvolává aktivity, které vyžadují spuštění v určitém časovém okamžiku.

2.3.2 Případy užití – obecné

Do této kategorie patří případy užití, které může vykonávat obecný uživatel aplikace. Patří sem jediná aktivita – přihlášení uživatele do systému, protože systém neumožňuje žádný přístup nepřihlášeným uživatelům. Diagram této části je zachycen na obrázku 2.2.

2.3.3 Případy užití – sekce správy projektů – obecné

Tato kategorie zahrnuje případy užití, které spouští aktér Purchaser po přihlášení do systému a vstupu do sekce správy projektů. Jedná se například o zakládání projektů, nahlížení na detail projektu, dodavatele, dílu a termínu a úplnou správu dodavatelů. Její diagram je na obrázku 2.3.



Obrázek 2.3: Model případů užití – sekce správy projektů – obecné

2.3.4 Případy užití – sekce správy projektů – projekty

Do této části jsem zařadil případy užití, které vykonává aktér PartialProjectLeader a souvisí se správou existujících projektů, jejich vlastností a souvisejících atributů. Patří mezi ně úprava vlastností projektu, správa projektového týmu a seznamu dílů a kompletní správa termínů a upomínek na ně. Všechny zde zmíněné aktivity může provádět také aktér PrivilegedPurchaser. Jak jsem již uvedl, jeho oprávnění se vztahuje na všechny existující projekty. Aktér PartialProjectLeader může tyto aktivity naopak vykonávat pouze v kontextu projektů, kterých je vedoucím nebo zástupcem vedoucího. Větší práva aktéra PrivilegedPurchaser jsem na diagramu 2.4 graficky znázornil pomocí případu užití „UC323: Manage all projects and parts“. Diagram 2.4 zachycuje i všechny ostatní případy užití z této skupiny.

Případ užití: Přidat existující díl do projektu (Add existing part to project)
ID: UC302
Stručný popis: Výběr existujícího dílu, jeho zařazení do projektu a přiřazení zodpovědných nákupčí.

Obrázek 2.4: Model případů užití – sekce správy projektů – projekty

Aktéři: VDP nebo zástupce VDP (PartialProjectLeader)
Vstupní podmínky:

1. Je vytvořen nějaký projekt.
2. Uživatel je přihlášen a je v roli PartialProjectLeader ve vztahu k danému projektu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Přidat existující díl“.
2. Systém zobrazí formulář pro přidání existujícího dílu.
3. Uživatel zadá „Vybrat díl“.
4. Systém zobrazí seznam všech existujících dílů, které dosud nejsou zařazeny do projektu.
5. Uživatel vybere požadovaný díl a výběr potvrdí.
6. Zahrnout (UC220: Přiřadit nákupčí zodpovědné za díl).
7. Uživatel zadá „Uložit“.
8. Systém přiřadí vybraný díl se zadanými zodpovědnými nákupčími k projektu.
9. Systém zobrazí přidaný díl na seznamu dílů v projektu.

Výstupní podmínky:

1. Díl byl přiřazen k projektu.
2. Zodpovědnost za díl byla přiřazena vybraným nákupčím.

Alternativní scénáře:

UC302.1: Žádné existující díly k dispozici

Alternativní scénář: Přidat existující díl do projektu: Žádné existující díly k dispozici
ID: UC302.1
Stručný popis: Systém informuje uživatele, že nejsou k dispozici žádné existující díly, které by mohly být přidány do projektu.
Aktéři: VDP nebo zástupce VDP (PartialProjectLeader)
Vstupní podmínky:
1. Nejsou k dispozici žádné díly, které by mohly být přidány do projektu.

Hlavní scénář:
1. Scénář začíná krokem 3. hlavního scénáře.
2. Systém zobrazí hlášení, že nejsou k dispozici žádné existující díly na výběr.
3. Uživatel ukončí přidávání existujícího dílu do projektu.

Výstupní podmínky:
1. Žádný díl nebyl přidán do projektu.

Případ užití: Přiřadit nákupčí zodpovědné za díl (Assign responsible purchasers to part)
ID: UC220
Stručný popis: Uživatel vybere z projektového týmu nákupčí zodpovědné za díl.
Aktéři: VDP nebo zástupce VDP (PartialProjectLeader)
Vstupní podmínky:
1. Je vytvořen nějaký projekt.
2. Uživatel je přihlášen a je v roli PartialProjectLeader ve vztahu k danému projektu.

Hlavní scénář:
1. Případ užití začíná, když uživatel zadá příkaz „Vybrat zodpovědné nákupčí“.
2. Systém zobrazí nákupčí, kteří jsou členy projektového týmu daného projektu.
3. Uživatel vybere požadované nákupčí a výběr potvrdí.
4. Systém přiřadí nákupčí k dílu.
5. Systém zobrazí seznam vybraných zodpovědných nákupčí.

Výstupní podmínky:
1. Vybraní zodpovědní nákupčí byli přiřazeni k dílu.

Případ užití: Přidat upozornění na blížící se termín (Add milestone reminder)
ID: UC318
Stručný popis: Vytvoření upozornění na blížící se datum termínu, které bude zasláno požadovaným osobám zadaný počet dní před proběhnutím termínu.
Aktéři: VDP nebo zástupce VDP (PartialProjectLeader)
Vstupní podmínky:

1. Je vytvořen nějaký projekt.
2. Uživatel je přihlášen a je v roli PartialProjectLeader ve vztahu k danému projektu.
3. Je vytvořen nějaký termín v daném projektu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Přidat upozornění“.
2. Systém zobrazí formulář pro zadání nového upozornění.
3. Uživatel vyplní všechny povinné údaje (počet dní, kdy má být upozornění zasláno před proběhnutím termínu a kdo obdrží upozornění – na výběr je: všichni, vedoucí a zástupce a výběr členů projektového týmu) a volitelně vyplní nepovinné údaje (poznámku k upozornění).
4. KDYŽ uživatel zadal, že upozornění obdrží výběr členů týmu:
   4.1. Místo rozšíření: UC322: Vybrat členy projektového týmu, kteří budou upozorněni na blížící se termín.
5. Uživatel zadá „Uložit“.
6. Systém vytvoří nové upozornění na blížící se termín.
7. Systém zobrazí nově vytvořené upozornění.

Výstupní podmínky:

1. Upozornění na blížící se termín bylo vytvořeno.

Případ užití: Vybrat členy projektového týmu, kteří budou upozorněni na blížící se termín (Select project team members to remind)
ID: UC322
Stručný popis: Uživatel vybere členy projektového týmu, kteří budou upozorněni na daný blížící se termín.
Aktéři: VDP nebo zástupce VDP (PartialProjectLeader)

Vstupní podmínky:

1. Uživatel zadal, že upozornění obdrží výběr členů projektového týmu.

Hlavní scénář:

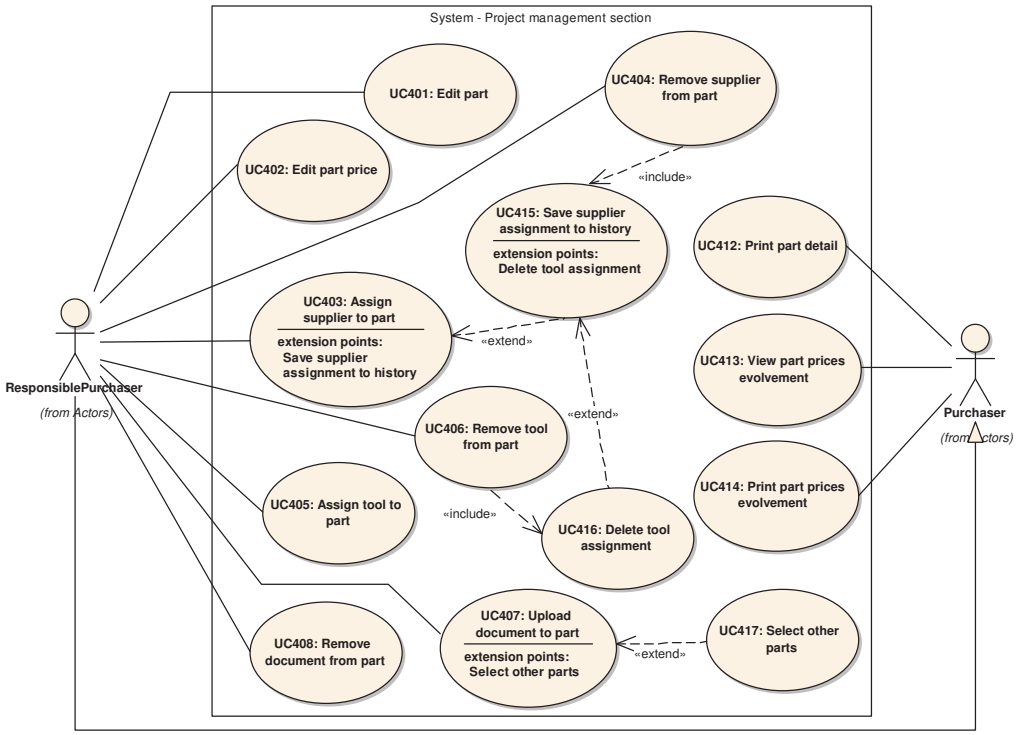
1. Uživatel zadá „Vybrat členy projektového týmu“.
2. Systém zobrazí seznam uživatelů, kteří vystupují v daném projektu v roli nákupčí.
3. Uživatel vybere požadované členy a výběr potvrdí.
4. Systém k danému upozornění přiřadí vybrané členy.
5. Systém zobrazí vybrané členy.

Výstupní podmínky:

1. Členi, kteří mají být upozorněni na blížící se termín byli vybráni.

2.3.5 Případy užití – sekce správy projektů – díly

Tato kategorie obsahuje případy užití související se správou dílu. Většinu z nich může vykonávat aktér ResponsiblePurchaser, tedy uživatel zodpovědný za díl, v jehož kontextu je aktivita prováděna. Z toho vyplývá, že stejné aktivity má možnost vykonávat také aktér PartialProjectLeader, za



Obrázek 2.5: Model případů užití – sekce správy projektů – díly

podmínky, že interaguje s díly v kontextu projektu, jehož je vedoucím nebo zástupcem vedoucího. Stejná oprávnění sdílí také aktér PrivilegedPurchaser bez ohledu na kontext. Zbylé aktivity může vykonávat aktér Purchaser. Jsou zde zahrnuty případy užití jako úprava vlastností dílu, úprava jeho ceny, přiřazení dodavatele a nástroje a nahrání dokumentu. Přehled těchto případů užití je zobrazen na diagramu 2.5.

Případ užití: Přiřadit dodavatele k dílu (Assign supplier to part)
ID: UC403
Stručný popis: Uživatel vybere a přiřadí dodavatele k dílu v nějakém projektu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:

1. Je vytvořen nějaký projekt a k němu je přiřazen alespoň jeden díl.
2. Uživatel je přihlášen a je v roli ResponsiblePurchaser ve vztahu k danému dílu v daném projektu.
3. Uživatel má zobrazen detail daného dílu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Vybrat dodavatele“.
2. Systém zobrazí seznam všech aktivních dodavatelů.

3. Uživatel vybere požadovaného dodavatele.
4. Uživatel zadá „Uložit“.
5. KDYŽ k dílu je již přiřazen nějaký dodavatel:
   5.1. Místo rozšíření: UC415: Uložit přiřazení dodavatele do historie.
6. Systém uloží nové přiřazení dodavatele k dílu.
7. Systém zobrazí informace o vybraném dodavateli v detailu dílu.

Výstupní podmínky:

1. Dodavatel byl přiřazen k dílu.

Případ užití: Odebrat přiřazení dodavatele od dílu (Remove supplier from part)
ID: UC404
Stručný popis: Odebere dodavatele přiřazeného k dílu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:

1. Je vytvořen nějaký projekt a k němu je přiřazen alespoň jeden díl, který má přiřazeného dodavatele.
2. Uživatel je přihlášen a je v roli ResponsiblePurchaser ve vztahu k danému dílu v daném projektu.
3. Uživatel má zobrazen detail daného dílu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Odebrat dodavatele“.
2. Systém vyžádá od uživatele potvrzení prováděné akce.
3. Zahrnout (UC415: Uložit přiřazení dodavatele do historie).
4. Systém v detailu dílu zobrazí, že aktuálně není přiřazen žádný dodavatel.

Výstupní podmínky:

1. Přiřazení dodavatele k dílu bylo odebráno.

Případ užití: Odebrat přiřazení zařízení od dílu (Remove tool from part)
ID: UC406
Stručný popis: Odebere zařízení přiřazené k dílu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:

1. Je vytvořen nějaký projekt a k němu je přiřazen alespoň jeden díl, který má přiřazený nástroj.
2. Uživatel je přihlášen a je v roli ResponsiblePurchaser ve vztahu k danému dílu v daném projektu.
3. Uživatel má zobrazen detail daného dílu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Odebrat zařízení“.
2. Systém vyžádá od uživatele potvrzení prováděné akce.
3. Zahrnout (UC416: Smazat přiřazení zařízení k dílu).
4. Systém v detailu dílu zobrazí jen zbývající přiřazená zařízení. Pokud není přiřazeno žádné, pak systém zobrazí, že aktuálně není přiřazeno žádné zařízení.

Výstupní podmínky:
1. Přiřazení zařízení k dílu bylo odebráno.

Případ užití: Uložit přiřazení dodavatele do historie (Save supplier assignment to history)
ID: UC415
Stručný popis: Uloží aktuální přiřazení dodavatele k dílu do historie a toto přiřazení smaže.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:
1. K danému dílu je přiřazen dodavatel.

Hlavní scénář:
1. Případ užití začíná, když uživatel odebírá přiřazení dodavatele k dílu nebo když přiřazuje nového dodavatele.
2. KDYŽ je k danému dílu přiřazeno alespoň jedno zařízení:
   2.1. PRO každé přiřazené zařízení:
      2.1.1. Místo rozšíření: UC416: Smazat přiřazení zařízení k dílu.
3. Systém uloží do historie přiřazení dodavatelů odebírané přiřazení a nastaví aktuální datum, jako datum zániku tohoto přiřazení.
4. Systém smaže toto přiřazení dodavatele k dílu.

Výstupní podmínky:
1. Přiřazení dodavatele k dílu bylo uloženo do historie a smazáno.

Případ užití: Smazat přiřazení zařízení k dílu (Delete tool assignment)
ID: UC416
Stručný popis: Smaže přiřazení daného zařízení k danému dílu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:
1. K danému dílu je přiřazeno dané zařízení.

Hlavní scénář:
1. Případ užití začíná, když uživatel odebírá přiřazení zařízení k dílu nebo když odebírá přiřazení dodavatele k dílu a k dílu je přitom přiřazeno alespoň jedno zařízení.
2. Systém smaže přiřazení zařízení k dílu.

Výstupní podmínky:
1. Přiřazení zařízení k dílu bylo smazáno.

Případ užití: Nahrát dokument k dílu (Upload document to part)
ID: UC407
Stručný popis: Nahraje dokument závislý na dodavateli, projektu a dílu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:

1. Je vytvořen nějaký projekt a k němu je přiřazen alespoň jeden díl, který má přiřazeného dodavatele.
2. Uživatel je přihlášen a je v roli ResponsiblePurchaser ve vztahu k danému dílu v daném projektu.
3. Uživatel má zobrazen detail daného dílu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Nahrát dokument“.
2. Systém zobrazí formulář pro nahrání dokumentu.
3. KDYŽ uživatel chce nahrát dokument k více dílům:
   3.1. Místo rozšíření: UC417: Vybrat další díly, ke kterým bude dokument přiřazen.
4. KDYŽ uživatel chce přiřadit existující dokument:
   4.1. Uživatel zadá „Vybrat existující dokument“.
   4.2. Systém zobrazí seznam dokumentů stejného typu, které jsou nahrány k dílům ze stejného projektu a dodávaných stejným dodavatelem.
   4.3. Uživatel vybere požadovaný dokument.
5. JINAK:
   5.1. Uživatel zadá příkaz „Vybrat nový dokument“.
   5.2. Systém zobrazí formulář pro výběr nového dokumentu.
   5.3. Uživatel vybere požadovaný dokument.
6. Uživatel zadá „Uložit“.
7. Systém nahraje a přiřadí dokument k danému dodavateli, projektu a vybraným dílům.
8. Systém zobrazí detail nahraného dokumentu v detailu dílu.

Výstupní podmínky:

1. Dokument byl nahrán a přiřazen k dílu.

Případ užití: Vybrat další díly, ke kterým bude dokument přiřazen (Select other parts)
ID: UC417
Stručný popis: Vybere více dílů, ke kterým má být přiřazen nahrávaný dokument.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)

Vstupní podmínky:

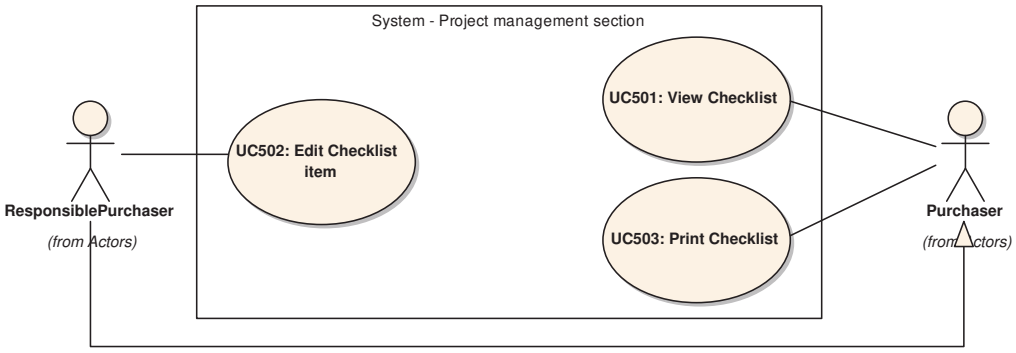
1. Existuje více dílů přiřazených v daném projektu a dodávaných stejným dodavatelem.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Vybrat další díly“.
2. Systém zobrazí seznam dílů, které jsou přiřazeny k danému projektu, a dodává je stejný dodavatel, který dodává díl, k němuž je dokument primárně přiřazován.
3. Uživatel vybere požadované díly a výběr potvrdí.
4. Systém zobrazí všechny vybrané díly, ke kterým bude nahrávaný dokument přiřazen.

Výstupní podmínky:

1. Byly vybrány díly, ke kterým má být dokument přiřazen.



Obrázek 2.6: Model případů užití – sekce správy projektů – checklist

2.3.6 Případy užití – sekce správy projektů – Checklist

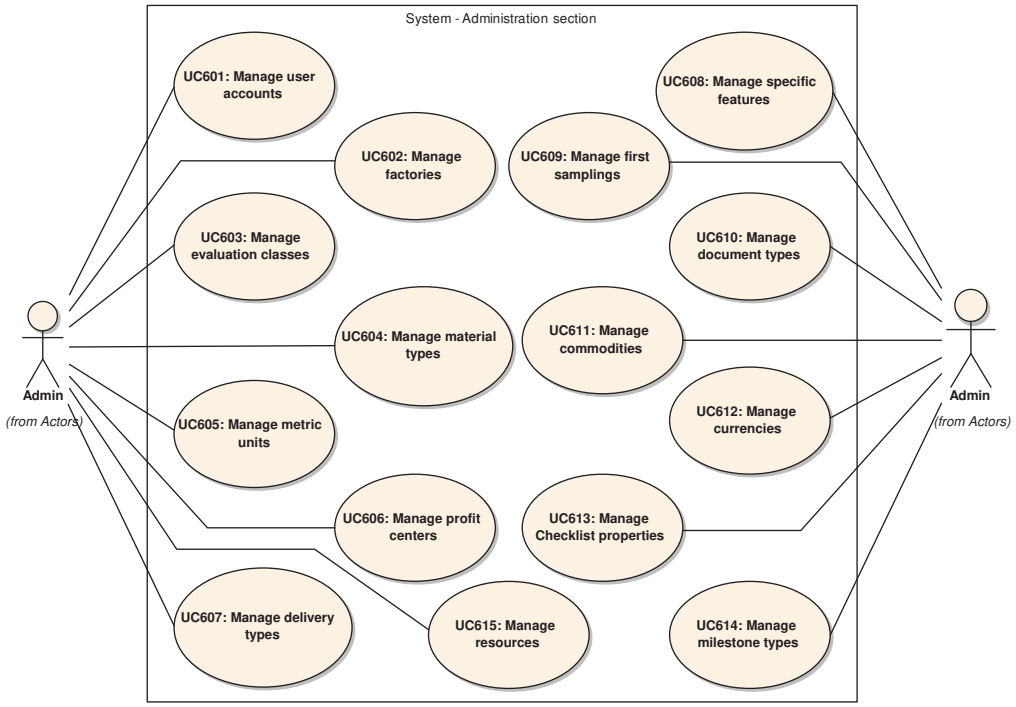
Jedná se o poslední kategorii případů užití, které spadají do sekce pro správu projektů. Vykonává je opět aktér ResponsiblePurchaser nebo Purchaser a na oprávnění se vztahují stejná pravidla, která jsem uvedl v předchozí části – sekce správy projektů – díly. Tyto případy užití souvisejí se správou a prohlížením Checklistu. Jsou zachyceny na obrázku 2.6.

Případ užití: Upravit položku Checklistu (Edit Checklist item)
ID: UC502
Stručný popis: Změní hodnotu vybrané položky Checklistu.
Aktéři: Zodpovědný nákupčí (ResponsiblePurchaser)
Vstupní podmínky:

1. Je vytvořen nějaký projekt a k němu je přiřazen alespoň jeden díl, který má přiřazeného dodavatele.
2. Uživatel je přihlášen a je v roli ResponsiblePurchaser ve vztahu k danému dílu v daném projektu.
3. Uživatel má zobrazen Checklist daného projektu.

Hlavní scénář:

1. Případ užití začíná, když uživatel zadá příkaz „Upravit položku“.
2. Systém zobrazí formulář pro úpravu položky Checklistu, kde je zobrazen související díl, typ položky a seznam možných hodnot a možnost zadat komentář.
3. Uživatel vybere požadovanou hodnotu a případně vyplní komentář. (Komentář vyplní povinně, pokud to vybraná hodnota vyžaduje.)
4. Uživatel zadá „Uložit“.
5. Systém uloží hodnotu položky.
6. KDYŽ upravovaná položka Checklistu má přiřazené požadované typy dokumentů:
   6.1. Systém vyhledá díly, které mají přiřazené totožné dokumenty shodného typu, jako jsou ty přiřazené k upravované položce.



Obrázek 2.7: Model případů užití – administrační sekce

6.2. Systém uloží hodnotu položky pro všechny nalezené díly.
7. Systém zobrazí změněné položky v Checklistu projektu.

Výstupní podmínky:

1. Položka Checklistu byla upravena.

2.3.7 Případy užití – administrační sekce

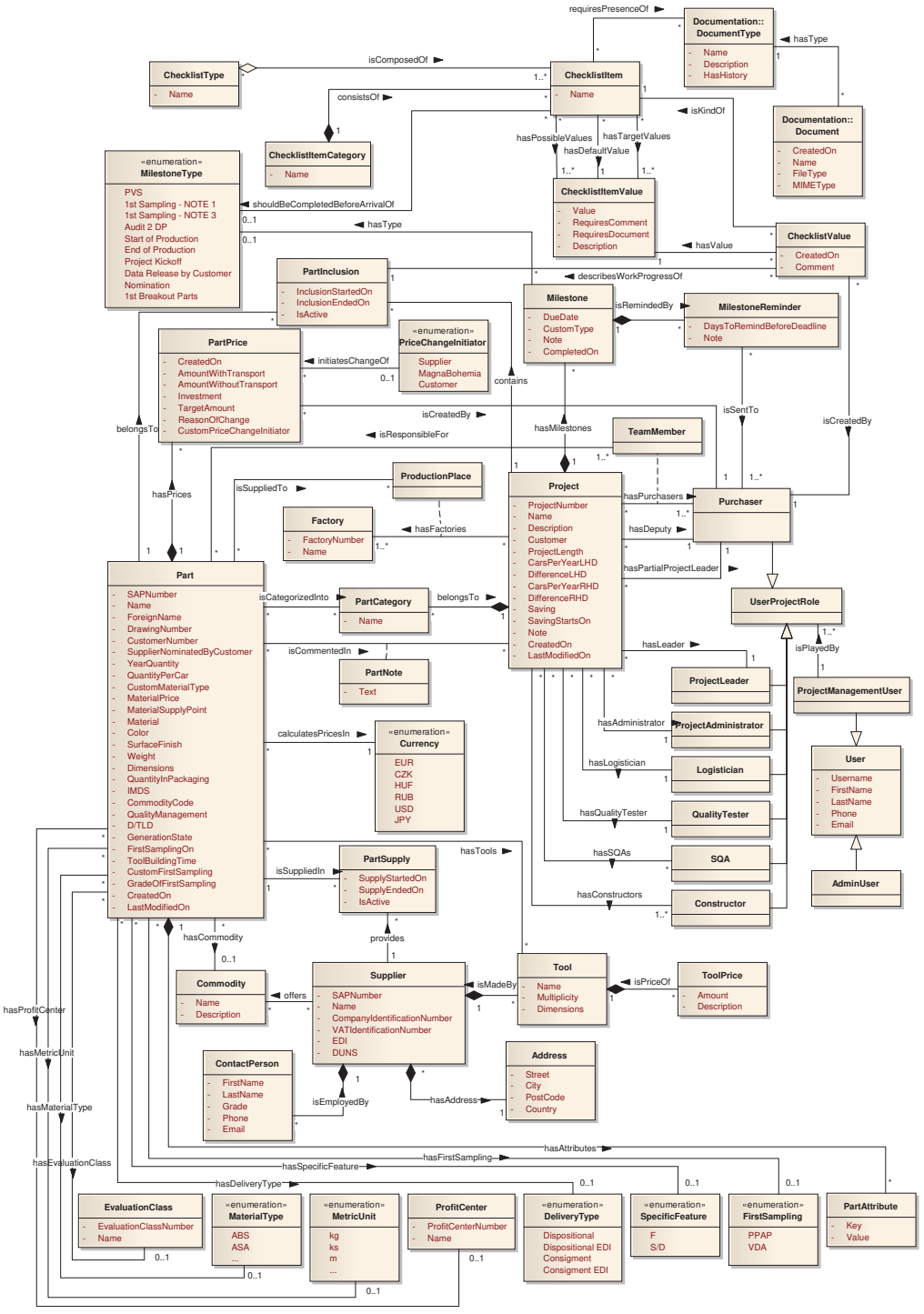
V této kategorii jsou obsaženy všechny případy užití, které vykonává aktér Admin v administrační sekci aplikace. Patří sem správa uživatelských účtů, typů dokumentů, typů termínů, druhů položek Checklistu a správa několika dalších atributů, z nichž uživatelé vybírají a přiřazují je k projektům a dílům v sekci pro správu projektů. Diagram těchto případů užití ukazuje obrázek 2.7.

2.4 Konceptuální model

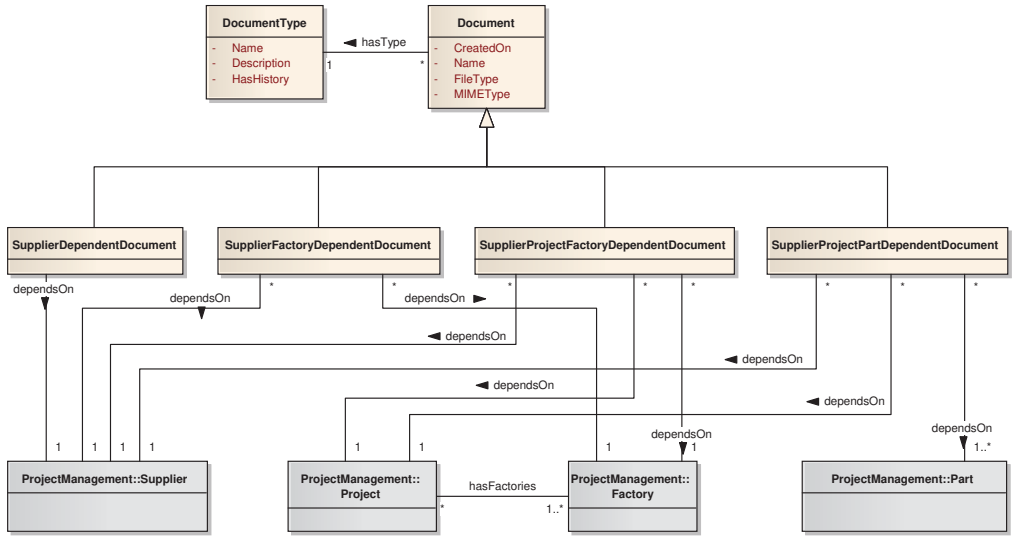
Konceptuální (nebo také doménový) model zachycuje konceptuální třídy reprezentující reálné objekty z modelované domény zájmu a vztahy mezi nimi. Nepopisuje softwarové třídy, komponenty ani jejich zodpovědnost nebo metody, viz [28]. Zachycuje se pomocí UML diagramu tříd a je zásadním výstupem procesu objektově orientované analýzy. Často slouží jako výchozí bod při návrhu platformově specifického implementačního modelu. Podrobný návod na tvorbu doménových modelů uvádí Larman [28].

Konceptuální model aplikace pro společnost Magna E&I jsem rozdělil do dvou diagramů, abych dosáhl větší přehlednosti. Na diagramu 2.8 jsou zachyceny téměř veškeré entity, které jsem identifikoval. Scházejí pouze entity a vazby ukazující závislosti různých typů dokumentů. Mezi stěžejní entity patří *Project* (projekt), *Part* (díl), *Supplier* (dodavatel) a *Milestone* (termín, milník). Neméně důležité jsou také entity představující uživatele systému (*User*) a jeho dva konkrétní druhy *AdminUser* a *ProjectManagementUser*. Druhý z nich může vystupovat v projektovém týmu pod jednou či více projektovými uživatelskými rolemi (*UserProjectRole*). Na základě role, kterou má uživatel přidělenou (*Purchaser*, *ProjectLeader*, *ProjectAdministrator*, *Logistician*, *QualityTester*, *SQA* a *Constructor*), může zastávat konkrétní pozice, které jsou navíc určeny i druhem vazby mezi rolí a projektem. Existuje několik způsobů jak modelovat (nejen uživatelské) role v aplikaci. Řešení, které jsem zvolil, vychází z [46], kde je nazvané jako generalizace třídy role (Role Class Generalization). Umožňuje zachytit fakt, že uživateli může být přiděleno několik rolí a zároveň může být součástí několika projektových týmů. V jednom týmu může vystupovat opakovaně, ale vždy v jiné pozici. Jedna z vazeb mezi entitami *Project* a *Purchaser* má napojenu asociační třídu *TeamMember* (člen týmu – nákupčí), na kterou je dále navázána entita „Part“. Tento vztah má vyjádřit, že za díl v projektu může být zodpovědný pouze projektu přiřazený nákupčí. Podobnou funkci má také asociační třída *ProductionPlace* napojená na vazbu mezi projektem a závodem (*Factory*). Jeden z požadavků na aplikaci je, aby byla uchována historie dílů zařazených do projektu a historie dodavatelů, kteří dodávali nějaký díl. Pro tyto účely slouží třídy *PartInclusion* a *PartSupply*.

Model obsahuje několik tříd, které společně vytváří projektový Checklist. Může existovat několik typů Checklistu (*ChecklistType*), které se skládají z několika sledovaných položek Checklistu (*ChecklistItem*). Položky se mohou seskupovat do kategorií (*ChecklistCategory*) a mají definované hodnoty (*ChecklistItemValue*), kterých mohou nabývat včetně jedné výchozí a několika konečných. Položka Checklistu může dále specifikovat, že pro zadání některých hodnot musí být nejprve v systému vloženy dokumenty



Obrázek 2.8: Konceptuální model – hlavní část



Obrázek 2.9: Konceptuální model – část projektové dokumentace

určitých typů (*DocumentType*), a že konečná hodnota by měla být zadána před uplynutím určitého typu termínu (*MilestoneType*). Systém uchovává konkrétní vyplněné hodnoty (*ChecklistValue*), které představují jednu z položek Checklistu a některou její hodnotu a váží se ke konkrétnímu dílu spadajícímu pod nějaký projekt. Tento návrh umožňuje dynamické definování struktury Checklistu, a je proto připraven na situaci, kdy společnost upraví sledované vlastnosti projektu a dílů.

Druhý diagram 2.9 obsahuje entitu představující dokument nahraný do systému (*Document*), který je vždy nějakého typu (*DocumentType*). Každý dokument je konkretizován na jednu z pěti tříd, podle toho, jakým způsobem je závislý na jedné nebo více třídách ze *Supplier*, *Project*, *Factory* a *Part*. Každý dokument souvisí s dodavatelem. Navíc se může vázat na závod nebo projekt a závod nebo projekt a jeden nebo více dílů. V tabulce 2.1 je uveden přehled všech dokumentů a jejich závislostí na zmíněných entitách. Také je zde zachyceno, zda typ dokumentu vyžaduje možnost ukládat více verzí.

Konceptuální model neumožňuje zachytit některá omezení kladená na vazby mezi entitami. Jedná se například o skutečnost, že člen týmu – nákupčí může být v projektu zodpovědný pouze za díly, které jsou součástí projektu. Tento druh omezení proto uvádím v podobě textového komentáře u každé vazby, která to vyžaduje, v níže uvedeném popisu vazeb.

V další části kapitoly popisuji význam jednotlivých entit, jejich atributů a vzájemných vztahů mezi nimi.

Tabulka 2.1: Závislosti dokumentů na entitách

Název dokumentu	Závislosti				Více verzí
	Dodavatel	Projekt	Závod	Díl(y)	
Dotazník o kvalitativní způsobilosti dodavatele	×				Ne
Nákupní podmínky	×				Ne
Ověření nového dodavatele	×				Ne
Logistický protokol	×		×		Ne
Rámcová smlouva	×		×		Ano
Příloha rámcové smlouvy	×	×	×		Ne
Smlouva o výpůjčce nástrojů	×	×	×		Ne
Balící předpis	×	×		×	Ne
Cenové porovnání	×	×		×	Ne
Dohoda o zajištění kvality	×	×		×	Ano
Katalog požadavků na nakupovaný díl	×	×		×	Ano
Nominační dopis	×	×		×	Ne
Plán zkoušek	×	×		×	Ano
První vzorkování	×	×		×	Ne
Předávací protokol na logistiku	×	×		×	Ne
Rozpad vítězné nabídky	×	×		×	Ne
Ustanovení o utajení a zachování mlčenlivosti	×	×		×	Ne
Životopis nakupovaného dílu	×	×		×	Ne

2.4.1 Entity a jejich atributy

User: abstraktní uživatel systému, který má vytvořený uživatelský účet. Do systému se mohou přihlašovat pouze uživatelé s platným uživatelským jménem.
Username – uživatelské jméno,
FirstName – křestní jméno,
LastName – příjmení,
Phone – telefonní číslo,
Email – email.

AdminUser: konkrétní uživatel představující administrátora systému.

ProjectManagementUser: konkrétní uživatel, který může zastávat jemu přidělené pozice v projektovém týmu.

UserProjectRole: abstraktní projektová uživatelská role, která může být přidělena uživateli.

Purchaser: konkrétní projektová role, která opravňuje uživatele zastávat pozici VDP, zástupce VDP a nákupčího zodpovědného za díly.

ProjectLeader: konkrétní projektová role, které opravňuje uživatele zastávat pozici projektového vedoucího.

ProjectAdministrator: konkrétní projektová role, které opravňuje uživatele zastávat pozici projektového administrátora.

Logistician: konkrétní projektová role, které opravňuje uživatele zastávat pozici logistika.

QualityTester: konkrétní projektová role, které opravňuje uživatele zastávat pozici kvalitáře.

SQA: konkrétní projektová role, které opravňuje uživatele zastávat pozici SQA.

Constructor: konkrétní projektová role, které opravňuje uživatele zastávat pozici konstruktéra.

Project: představuje projekt, jehož vypracováním je OPN pověřeno. Představuje hlavní entitu modelu, na kterou jsou nějakým způsobem navázány ostatní třídy.
ProjectNumber – unikátní číslo projektu,
Name – název projektu,
Description – poznámka k projektu,
Customer – zákazník, pro něhož je v průběhu projektu připravován produkt,
ProjectLength – délka projektu v letech,
CarsPerYearLHD – počet vyrobených vozů s levostranným řízením za rok,
DifferenceLHD – procentuální odchylka od počtu vyrobených vozů s levostranným řízením za rok,
CarsPerYearLHD – počet vyrobených vozů s pravostranným řízením za rok,
DifferenceLHD – procentuální odchylka od počtu vyrobených vozů s pravostranným řízením za rok,
Saving – saving projektu,
SavingStartsOn – začátek savingu v roce,
Note – poznámka k projektu,
CreatedOn – datum založení projektu,
LastModifiedOn – datum poslední změny projektu.

TeamMember (asociační třída): představuje vztah vznikající přiřazením nákupčího k projektu.

Factory: výrobní závod společnosti Magna E&I.

FactoryNumber – unikátní číslo závodu,
Name – název závodu.

ProductionPlace (asociační třída): představuje vztah, který vzniká mezi projektem a závodem.

Milestone: projektový milník, který může mít buď přiřazen předdefinovaný typ, nebo se může jednat o projektově unikátní termín. Váže se k projektu.
DueDate – datum, do kterého mají být splněny povinnosti spojené s termínem,
CustomType – uživatelem zadaný vlastní typ termínu,
Note – poznámka k termínu,
CompletedOn – datum splnění povinností spojených s termínem.

MilestoneType (výčet): seznam možných typů termínů. Musí být možné je v systému upravovat, přidávat a mazat.

MilestoneReminder: upozornění na blížící se termín, které je uživateli zasláno emailem.
DayToRemindBeforeDeadline – počet dní před datem proběhnutí termínu, kdy má být upozornění zasláno,
Note – poznámka k upozornění.

Part: díl, který je součástí výsledného produktu projektu. Přiřazuje se k projektu.
SAPNumber – unikátní číslo přidělené dílu v podnikovém systému SAP,
Name – název dílu,
ForeignName – cizojazyčný název dílu,
DrawingNumber – číslo výkresu zákazníka,
CustomerNumber – zákaznické číslo dílu,
SupplierNominatedByCustomer – příznak určující zda dodavatele nominuje zákazník,
YearQuantity – počet vyrobených dílů za rok,
QuantityPerCar – počet dílů na vůz,
CustomMaterialType – specifický typ materiálu, který není v nabídce,
MaterialPrice – cena materiálu,
MaterialSupplyPoint – odběrné místo materiálu,
Material – obchodní název materiálu, ze kterého je díl vyroben,
Color – barva dílu,
SurfaceFinish – povrchová úprava díly,
Weight – váha dílu,
Dimensions – rozměry dílu,
QuantityInPackaging – počet dílu v jednom balení,

IMDS – standardizované IMDS[7] číslo materiálu,
CommodityCode – MEPI klíč - komoditní kód,
QualityManagement – příznak určující zda díl podléhá vstupní kontrole,
D/TLD – příznak určující, zda má díl přiřazený tento identifikační kód používaný koncernem Volkswagen.
GenerationState – generační stav dílu,
FirstSamplingOn – datum prvního vzorkování,
ToolBuildingTime – doba stavby nástroje v kalendářních týdnech,
CustomFirstSampling – specifický druh prvního vzorkování, který není v nabídce,
GradeOfFirstSampling – úroveň předložení prvního vzorkování,
CreatedOn – datum vytvoření dílu,
LastModifiedOn – datum poslední změny dílu.

PartInclusion: zachycuje přiřazení dílu do projektu.
InclusionStaredOn – datum, kdy byl díl přiřazen k projektu,
InclusionEndedOn – datum, kdy byl díl odebrán z projektu,
IsActive – příznak určující zda je přiřazení dílu momentálně aktivní.

PartPrice: cena dílu. K dílu je možné přidávat více cen, ale platná je vždy pouze ta poslední. Je tak možné zachytit postupný vývoj ceny dílu v čase.
CreatedOn – datum vytvoření ceny,
AmountWithTransport – cena dílu s dopravou,
AmountWithoutTransport – cena dílu bez dopravy,
Investment – jednorázová investice do přípravy a výroby dílu,
TargetAmount – targetová cena dílu,
ReasonOfChange – číslo a důvod změny ceny dílu,
CustomPriceChangeInitiator – specifický původce změny ceny dílu, který není v nabídce.

PriceChangeInitiator (výčet): původce změny ceny dílu. Musí být možné je v systému upravovat, přidávat a mazat.

PartCategory: kategorie, do kterých mohou být díly pro větší přehlednost zařazovány. Obecně může být díl umístěn do mnoha kategorií, v kontextu projektu však pouze do jedné.
Name – název kategorie.

PartNote (asociační třída): poznámka k dílu, která je unikátní pro aktivní přiřazení dílu k projektu.

[7]IMDS – International Material Data Systém – Mezinárodní systém pro správu dat o materiálech, který umožňuje automobilovým výrobcům a dodavatelům jednotlivých součástí standardizovat své procesy a umožnit efektivní výměnu dat. [21]

Text – text poznámky.

Currency (výčet): měna, ve které jsou uváděny ceny dílu. Musí být možné je v systému upravovat, přidávat a mazat.

EvaluationClass: třída ocenění přiřazovaná dílu.
EvaluationClassNumber – unikátní číslo třídy ocenění,
Name – název.

MaterialType (výčet): typ materiálu přiřazovaný dílu. Musí být možné je v systému upravovat, přidávat a mazat.

MetricUnit (výčet): metrická jednotka přiřazovaná dílu. Musí být možné je v systému upravovat, přidávat a mazat.

ProfitCenter: profitové centrum přiřazované dílu.
ProfitCenterNumber – unikátní číslo profitového centra,
Name – název.

DeliveryType (výčet): typ dodávek přiřazovaný dílu. Musí být možné je v systému upravovat, přidávat a mazat.

SpecificFeature (výčet): zvláštní znak přiřazovaný dílu. Musí být možné je v systému upravovat, přidávat a mazat.

FirstSampling (výčet): druh prvního vzorkování přiřazovaný dílu. Musí být možné je v systému upravovat, přidávat a mazat.

PartAttribute: vlastní atribut dílu.
Key – název atributu,
Value – hodnota atributu.

Supplier: dodavatel, se kterým společnost Magna E&I spolupracuje. Připravuje výrobu jednoho nebo více dílů a následně zajišťuje jejich dodávky.
SAPNumber – unikátní číslo přidělené dodavateli v podnikovém systému SAP,
Name – název dodavatele,
CompanyIdentificationNumber – identifikační číslo (IČ) dodavatele,
VATIdentificationNumber – daňové identifikační číslo (DIČ) dodavatele,
EDI - EDI[8] číslo dodavatele,
DUNS – DUNS[9] číslo dodavatele.

PartSupply: zachycuje přiřazení dodavatele k dílu, tedy skutečnost, že díl je dodáván nějakým dodavatelem. Dodavatel dílu může být změněn, a proto je takto zajištěno uchování historie dodavatelů.
SupplyStartedOn – datum, kdy byl dodavatel přiřazen dílu,
SupplyEndedOn – datum, kdy bylo přiřazení dodavatele zrušeno,

[8]EDI – Electronic Data Interchange
[9]DUNS – Data Universal Numbering System

IsActive – příznak určující zda je přiřazení dodavatele momentálně aktivní.

Commodity: komodita, kterou dodavatel nabízí, a která může být přiřazena dílu.
Name – název komodity,
Description – popis komodity.

Address: adresa dodavatele.
Street – ulice včetně čísla popisného,
City – město,
PostCode – poštovní směrovací číslo,
Country – země.

ContactPerson: kontaktní osoba zaměstnaná dodavatelem, se kterou nákupčí komunikují.
FirstName – křestní jméno,
LastName – příjmení,
Grade – pracovní funkce,
Phone – telefonní číslo,
Email – email.

Tool: zařízení vyráběné nějakým dodavatelem a použité na výrobu nebo otestování dílu. Přiřazuje se dílu.
Name – název zařízení,
Multiplicity – násobnost,
Dimensions – rozměry zařízení.

ToolPrice: dílčí cena zařízení. Celková cena zařízení se může skládat z mnoha dílčích.
Amount – částka,
Description – popis ceny.

ChecklistType: typ Checklistu, který definuje sledované položky. Jedná se například o projektový nebo dokumentační.
Name – název Checklistu.

ChecklistItem: položka Checklistu, která je sledována zvlášť pro každý díl v projektu. Položky tvoří sloupce tabulkové struktury Checklistu. Díly představují jednotlivé řádky.
Name – název položky Checklistu.

ChecklistItemCategory: kategorie, do kterých se zařazují položky Checklistu.
Name – název kategorie.

ChecklistItemValue: hodnoty, kterých mohou položky Checklistu nabývat. Položkám je přiřazena výchozí hodnota, jedna nebo více cílových hodnot a seznam přiřaditelných hodnot.
Value – název hodnoty,

RequiresComment – příznak určující zda vybrání hodnoty vyžaduje také vložení komentáře,
RequiresDocument – příznak určující zda vybrání hodnoty vyžaduje přítomnost dokumentu definovaného u položky Checklistu,
Description – popis hodnoty.

ChecklistValue: konkrétní hodnota zadaná uživatelem k dílu a položce Checklistu. Reprezentuje jedno pole z tabulkové struktury Checklistu.
CreatedOn – datum vytvoření hodnoty,
Comment – komentář zadaný při vkládání hodnoty.

Document: elektronický dokument nahraný do systému.
CreatedOn – datum vložení dokumentu,
Name – název dokumentu,
FileType – typ souboru,
MIMEType – MIME[10] typ souboru.

DocumentType: typ projektového dokumentu, které se shromažďují pro potřeby projektu.
Name – název typu dokumentu,
Description – popis typu dokumentu,
HasHistory – příznak určující zda typ dokumentu vyžaduje ukládání více verzí souborů.

SupplierDependentDocument: konkrétní dokument, který závisí na dodavateli.

SupplierFactoryDependentDocument: konkrétní dokument, který závisí na dodavateli a závodu.

SupplierProjectFactoryDependentDocument: konkrétní dokument, který závisí na dodavateli, projektu a závodu.

SupplierProjectPartDependentDocument: konkrétní dokument, který závisí na dodavateli, projektu a dílu.

2.4.2 Vztahy mezi entitami

isPlayedBy (ProjectManagementUser, UserProjectRole – 1:1..M) – uživatel (*ProjectManagementUser*) musí zastávat jednu nebo více projektových rolí (*UserProjectRole*).

hasPartialProjectLeader (Project, Purchaser – M:1) – každý projekt (*Project*) vede právě jeden VDP (*Purchaser*).

hasDeputy (Project, Purchaser – M:1) – každý projekt (*Project*) má právě jednoho zástupce VDP (*Purchaser*). VDP a zástupce VDP musejí být

[10]MIME – Multipurpose Internet Mail Extensions

dva různí uživatelé ztvárnění dvěma různými konkrétními projektovými rolemi.

hasPurchasers (Project, TeamMember, Purchaser – M:1, 1:N..1) – v každém projektu (*Project*) vystupuje alespoň jeden nákupčí (*Purchaser*) a vytváří tak člena projektového týmu (*TeamMember*), který je jednoznačně určen právě projektem a nákupčím.

hasLeader (Project, ProjectLeader – M:1) – každý projekt (*Project*) má právě jednoho hlavního vedoucího (*ProjectLeader*).

hasAdministrator (Project, ProjectAdministrator – M:1) – každý projekt (*Project*) má právě jednoho projektového administrátora (*ProjectAdministrator*).

hasLogistician (Project, Logistician – M:1) – každý projekt (*Project*) má právě jednoho logistika (*Logistician*).

hasQualityTester (Project, QualityTester – M:1) – každý projekt (*Project*) má právě jednoho kvalitáře (*QualityTester*).

hasSQAs (Project, SQA – M:N) – každý projekt (*Project*) má libovolný počet SQA (*SQA*).

hasConstructors (Project, Constructor – M:1..N) – každý projekt (*Project*) má alespoň jednoho konstruktéra (*Constructor*).

hasMilestones (Project, Milestone – 1:M) – ke každému projektu (*Project*) může být přiřazeno libovolné množství termínů (*Milestone*). Termíny v projektu musejí být unikátní z hlediska svých přiřazených typů (*MilestoneType*). Tedy termín s daným typem se v projektu může vyskytovat pouze jednou. Toto omezení se nevztahuje na termíny bez přiřazeného typu.

hasType (Milestone, MilestoneType – M:0..1) – termín (*Milestone*) může být nejvýše jednoho typu (*MilestoneType*). Pokud není zadán typ termínu, musí být vyplněno pole *CustomType*.

isRemindedBy (Milestone, MilestoneReminder – 1:M) – pro každý termín (*Milestone*) je možné vytvořit libovolné množství připomenutí (*MilestoneReminder*).

isSentTo (MilestoneReminder, Purchaser – M:1..N) – připomenutí (*MilestoneReminder*) je zasláno vždy alespoň jednomu členovi projektového týmu (*Purchaser*). Musí se vždy jednat o VDP, zástupce VDP, nebo nákupčího, kteří jsou přiřazení stejnému projektu, se kterým souvisí termín (*Milestone*), na nějž má být upozorněno.

hasFactories (Project, ProductionPlace, Factory – M:1, 1:1..N) – produkt navrhovaný v průběhu projektu (*Project*) je ve výsledku vyráběn v alespoň jednom závodě (*Factory*). Výrobní místo (*ProductionPlace*) je jednoznačně určeno projektem a závodem.

contains (Project, PartInclusion – 1:M) – každému projektu (*Project*)

může být přiřazeno (*PartInclusion*) libovolné množství dílů.

belongsTo (Part, PartInclusion – 1:M) – díl (*Part*) může být libovolněkrát zařazen (*PartInclusion*) do projektů. Vždy může existovat pouze jedno aktivní přiřazení určitého dílu k určitému projektu.

isResponsibleFor (TeamMember, Part – 1..M:N) – nákupčí (*TeamMember*) může být zodpovědný za libovolné množství dílů (*Part*). Zodpovídat může pouze za díly aktuálně zařazené do projektu, jehož je on sám součástí. Za každý díl přiřazený do projektu zodpovídá alespoň jeden nákupčí.

isSuppliedTo (Part, ProductionPlace – M:N) – díl (*Part*) může být dodáván do libovolného výrobního místa (*ProductionPlace*). Lze mu přiřadit pouze výrobní místa, která souvisejí se závody přiřazenými projektu, v jehož kontextu je díl dodáván.

isCategorizedInto (Part, PartCategory – M:N) – díl (*Part*) může být zařazen do libovolného množství kategorií (*PartCategory*), pro které však platí, že musí každá patřit do jiného projektu. Díl může být zařazen pouze do kategorií.

belongsTo (PartCategory, Project – M:1) – každá kategorie dílu (*PartCategory*) patří do právě jednoho projektu (*Project*).

isCommentedIn (Part, PartNote, Project – M:1, 1:N) – u dílu (*Part*) může být v rámci každého projektu (*Project*), kterému je aktuálně přiřazen, evidována právě jedna poznámka (*PartNote*).

hasPrices (Part, PartPrice – 1:M) – dílu (*Part*) může být přiřazeno libovolné množství cen (*PartPrice*).

initiatesChangeOf (PriceChangeInitiator, PartPrice – 0..1:M) – u každé změny ceny dílu (*PartPrice*), tedy u druhé a další ceny přiřazené dílu, musí buď být uveden její iniciátor (*PriceChangeInitiator*) nebo musí být vyplněno pole *CustomPriceChangeInitiator*.

isCreatedBy (PartPrice, Purchaser – M:1) – každou cenu dílu (*PartPrice*) vytváří nějaký nákupčí (*Purchaser*).

calculatesPricesIn (Part, Currency – M:1) – cena dílu (*Part*) je počítána v právě jedné měně (*Currency*).

hasProfitCenter (Part, ProfitCenter – M:0..1) – každému dílu (*Part*) může být přiřazeno nejvýše jedno profitové centrum (*ProfitCenter*).

hasMetricUnit (Part, MetricUnit – M:0..1) – každému dílu (*Part*) může být přiřazena nejvýše jedna metrická jednotka (*MetricUnit*).

hasMaterialType (Part, MaterialType – M:0..1) – každému dílu (*Part*) může být přiřazen nejvýše jeden typ materiálu (*MaterialType*). Typ materiálu nelze přiřadit, pokud má díl vyplněné pole *CustomMaterialType*.

hasEvaluationClass (Part, EvaluationClass – M:0..1) – každému dílu

(*Part*) může být přiřazena nejvýše jedna třída ocenění (*EvaluationClass*).

hasDeliveryType (Part, DeliveryType – M:0..1) – každému dílu (*Part*) může být přiřazen nejvýše jeden typ dodávek (*DeliveryType*).

hasSpecificFeature (Part, SpecificFeature – M:0..1) – každému dílu (*Part*) může být přiřazen nejvýše jeden zvláštní znak (*SpecificFeature*).

hasFirstSampling (Part, FirstSampling – M:0..1) – každému dílu (*Part*) může být přiřazeno nejvýše jedno první vzorkování (*FirstSampling*). První vzorkování nelze přiřadit, pokud má díl vyplněné pole *CustomFirstSampling*.

hasAttributes (Part, PartAttribute – 1:M) – u každého dílu (*Part*) může být zadáno libovolné množství vlastních atributů (*PartAttribute*).

hasCommodity (Part, Commodity – M:0..1) – každému dílu (*Part*) může být přiřazena nejvýše jedna komodita (*Commodity*).

isSuppliedIn (Part, PartSupply – 1:M) – díl (*Part*) může být dodáván v libovolném počtu dodávek (*PartSupply*) od nějakého dodavatele. Vždy může existovat pouze právě jedna aktivní dodávka nějakého dílu.

provides (Supplier, PartSupply – 1:M) – dodavatel (*Supplier*) zajišťuje libovolné množství dodávek dílu (*PartSupply*). Aktivní dodávku nějakého dílu obstarává vždy právě jeden dodavatel.

offers (Supplier, Commodity – M:N) – dodavatel (*Supplier*) může nabízet libovolné množství komodit (*Commodity*).

isEmployedBy (ContactPerson, Supplier – M:1) – každou kontaktní osobu (*ContactPerson*) zaměstnává právě jeden dodavatel (*Supplier*).

hasAddress (Supplier, Address – M:1) – každý dodavatel (*Supplier*) sídlí na právě jedné adrese (*Address*).

hasTools (Part, Tool – M:N) – dílu (*Part*) může být přiřazeno libovolné množství zařízení (*Tool*). Může se jednat pouze o zařízení vyráběné dodavatelem, který daný díl aktuálně dodává.

isMadeBy (Tool, Supplier – M:1) – každé zařízení (*Tool*) je dodáváno právě jedním dodavatelem (*Supplier*).

isPriceOf (ToolPrice, Tool – M:1) – celková cena zařízení (*Tool*) se může skládat z libovolného počtu dílčích cen (*ToolPrice*).

isComposedOf (ChecklistType, ChecklistItem – M:1..N) – typ Cheklistu (*ChecklistType*) je sestaven z alespoň jedné položky Checklistu (*ChecklistItem*).

consistsOf (ChecklistItemCategory, ChecklistItem – 1:M) – kategorie Checklistu (*ChecklistCategory*) obsahuje libovolné množství položek (*ChecklistItem*).

hasPossibleValues (ChecklistItem, ChecklistItemValue – M:1..N) – u každé

položky Checklistu (*ChecklistItem*) musí být evidována alespoň jedna hodnota (*ChecklistItemValue*), které položka může nabývat.

hasDefaultValue (ChecklistItem, ChecklistItemValue – M:1) – každá položka Checklistu (*ChecklistItem*) má přiřazenu právě jednu výchozí hodnotu (*ChecklistItemValue*).

hasTargetValues (ChecklistItem, ChecklistItemValue – M:1..N) – každá položka Checklistu (*ChecklistItem*) má definovánu alespoň jednu cílovou hodnotu (*ChecklistItemValue*).

shouldBeCompletedBeforeArrivalOf (ChecklistItem, MilestoneType – M:0..1) – položka Checklistu (*ChecklistItem*) může souviset s typem termínu (*MilestoneTypem*), před jehož proběhnutím by měla mít nastavenu jednu z cílových hodnot.

requiresPresenceOf (ChecklistItem, DocumentType – M:N) – položka Checklistu (*ChecklistItem*) může být propojena s více typy dokumentů (*DocumentType*), které musejí být v systému nahrány předtím, než je možné vybrat některou z cílových hodnot položky.

isKindOf (ChecklistValue, ChecklistItem – M:1) – konkrétní hodnota položky Checklistu (*ChecklistValue*) je nějakého typu (*ChecklistItem*).

hasValue (ChecklistValue, ChecklistItemValue – M:1) – konkrétní hodnota položky Checklistu (*ChecklistValue*) má přiřazenou některou hodnotu (*ChecklistItemValue*). Lze vybírat pouze z hodnot asociovaných vazbou *hasPossibleValues* s položkou Checklist, se kterou tato konkrétní hodnota souvisí.

describesWorkProgressOf (ChecklistValue, PartInclusion – M:1) - konkrétní hodnota položky Checklistu (*ChecklistValue*) popisuje postup práce na přiřazení (*PartInclusion*) nějakého dílu v projektu.

isCreatedBy (ChecklistValue, Purchaser – M:1) - konkrétní hodnota položky Checklistu (*ChecklistValue*) je vytvořena právě jedním nákupčím (*Purchaser*).

hasType (Document, DocumentType – M:1) – dokument (*Document*) je vždy právě jednoho typu (*DocumentType*).

dependsOn (SupplierDependentDocument, Supplier – M:1) – dokument závislý na dodavateli (*SupplierDependentDocument*) má vždy přiřazeného právě jednoho dodavatele (*Supplier*).

dependsOn (SupplierFactoryDependentDocument, Supplier – M:1) – dokument závislý na dodavateli a závodu (*SupplierFactoryDependentDocument*) má vždy přiřazeného právě jednoho dodavatele (*Supplier*).

dependsOn (SupplierFactoryDependentDocument, Factory – M:1) – dokument závislý na dodavateli a závodu (*SupplierFactoryDependentDocument*) má vždy přiřazen právě jeden závod (*Factory*).

dependsOn (SupplierProjectFactoryDependentDocument, Supplier – M:1)

– dokument závislý na dodavateli, projektu a závodu (*SupplierProjectFactoryDependentDocument*) má vždy přiřazeného právě jednoho dodavatele (*Supplier*).

dependsOn (SupplierProjectFactoryDependentDocument, Project – M:1) – dokument závislý na dodavateli, projektu a závodu (*SupplierProjectFactoryDependentDocument*) má vždy přiřazen právě jeden projekt (*Project*).

dependsOn (SupplierProjectFactoryDependentDocument, Factory – M:1) – dokument závislý na dodavateli, projektu a závodu (*SupplierProjectFactoryDependentDocument*) má vždy přiřazen právě jeden závod (*Factory*).

dependsOn (SupplierProjectPartDependentDocument, Supplier – M:1) – dokument závislý na dodavateli, projektu a dílu (*SupplierProjectPartDependentDocument*) má vždy přiřazeného právě jednoho dodavatele (*Supplier*).

dependsOn (SupplierProjectPartDependentDocument, Project – M:1) – dokument závislý na dodavateli, projektu a dílu (*SupplierProjectPartDependentDocument*) má vždy přiřazen právě jeden projekt (*Project*).

dependsOn (SupplierProjectPartDependentDocument, Part – M:1..N) – dokument závislý na dodavateli, projektu a dílu (*SupplierProjectPartDependentDocument*) má vždy přiřazen alespoň jeden díl (*Part*).

2.5 Přehled a výběr technologií

V této kapitole představím hlavní technologie, se kterými budu pracovat a uvedu důvody, které mě vedly k jejich výběru.

2.5.1 Microsoft .NET Framework

Microsoft .NET Framework je softwarová aplikační vývojová platforma, která poskytuje služby pro vývoj, nasazení a provozování desktopových, webových a mobilních aplikací a webových služeb především v prostředí operačních systému Windows, ale i v prostředí jiných operačních systému, které nepocházejí z produkce společnosti Microsoft – například Mac OS X a další rozličné distribuce Unix/Linux [39, 58]. První verze s označením 1.0 byla mezi vývojáře uvolněna začátkem roku 2002. Od té doby bylo vydáno pět nových generací, poslední z nich v druhé polovině roku 2012 nesoucí označení 4.5 [41]. Oproti předchozím nástrojům pro vývoj aplikací pod Windows .NET Framework charakterizují následující odlišné rysy, viz [58]:

- Interoperabilita se starším existujícím kódem: Je možné navzájem mísit existující kód vytvořený na předchozí vývojové platformě COM[11] s kódem napsaným pro .NET Framework.

- Podpora pro několik programovacích jazyků: Aplikace pro prostředí .NET mohou být vyvíjeny v jednom z mnoha programovacích jazyků – například C#, Visual Basic, F# nebo S#.

- Společné běhové prostředí pro všechny podporované jazyky: Tato vlastnost je umožněna přesně definovanou sadou typů, které každý jazyk pro .NET Framework podporuje.

- Úplná integrace jazyků: .NET Framework poskytuje mezi-jazykovou dědičnost, zpracování výjimek, a ladění kódu.

- Komplexní knihovna základních tříd: Programátoři mohou při vývoji aplikací vybírat z množiny připravených tříd, které byly vytvořeny pro každý z podporovaných jazyků a obsahují spolehlivý otestovaný znovupoužitelný kód. Mezi nejdůležitější nabízenou funkčnost lze zařadit přístup k databázi, zabezpečení, tvorbu a správu vláken, funkce pro čtení a zápis souborů a prostředky pro vykreslování grafiky a komunikaci s rozličnými hardwarovými zařízeními. Framework kromě tříd nabízí také nástroje a technologie pro vývoj aplikací spadajících do specifických oblastí, viz [37]. Jedná se například o ASP.NET pro webové aplikace, ADO.NET pro přístup k datům a WCF[12] pro tvorbu servisně orientovaných aplikací.

- Zjednodušený model nasazení: .NET Framework nevyžaduje registraci jednotek binárního kódu do systémových registrů.

Mimo knihovnu základních tříd .NET Framework obsahuje tři základní stavební bloky, kterými jsou CLR[13], CTS[14] a CLS[15], viz [58]. CLR, neboli běhové prostředí, se stará o najití, nahrání a řízení .NET tříd, typů a kódu v době běhu aplikace. Jeho dalšími nepostradatelnými funkcemi jsou správa paměti, hostování aplikací, ovládání vláken a kontrola bezpečnosti. CTS

[11] COM – Component Object Model byl aplikační vývojový rámec společnosti Microsoft populární v 90. letech, který předcházel .NET Frameworku. Umožňoval tvorbu takzvaných COM serverů, což zjednodušeně řečeno byly znovupoužitelné celistvé oddíly binárního kódu. Více viz [58].

[12] WCF – Windows Communication Foundation

[13] CLR – Common Language Runtime

[14] CTS – Common Type System

[15] CLS – Common Language Specification

plně popisuje všechny možné typy dat a programové konstrukty podporované běhovým prostředím. Uvádí, jak spolu navzájem souvisí a interagují a jaký formát metadat je reprezentuje. Typ je v tomto kontextu obecné pojmenování pro třídu, interface, strukturu, výčet a delegáta. CLS je poslední blok, který specifikuje, jaké typy z CTS jsou společné a známé všem podporovaným .NET jazykům. Používání pouze těchto typů tak zajistí, že zkompilovaný kód bude všem těmto jazykům srozumitelný.

Kód napsaný v určitém .NET jazyku je zkompilován příslušným kompilátorem. Například pro kód v jazyce C# je použit C# kompilátor. Výstupem kompilace jsou takzvané .NET Assemblies, neboli sestavení, což jsou binární soubory typu *.dll nebo *.exe. Obsahují platformově nezávislé instrukce a metadata v CIL[16]. Právě tento způsob kompilace umožňuje integraci mnoha jazyků a přítomnost jediného společného běhového prostředí pro všechny z nich. Navíc zajišťuje platformovou nezávislost celého .NET Frameworku, podobně jako tomu je u jazyku Java. Již v dnešní době existují různé implementace pro podporu .NET Frameworku na jiných operačních systémech než Windows. K dispozici je také mezinárodní standard pro jazyk C#, viz [10]. Před samotným zpracováním zkompilovaného kódu v CLR dochází již za běhu programu k ještě jedné kompilaci do platformově specifického kódu, kterou provádí JIT[17] kompilátor. Jeho implementace je přizpůsobena typu zařízení a operačnímu systému, na kterém má být program spouštěn. Takto zkompilovaný kód je uložen do paměti přístroje a používán, dokud není jeho zdroj změněn. Celý postup kompilace je znázorněn na obrázku 2.10.

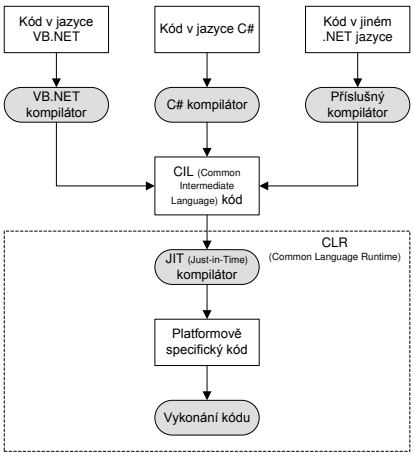
V prostředí .NET mohou vývojáři připravovat aplikace mnoha typů, od konzolových aplikací, přes desktopové (Windows Forms a Windows Presentation Foundation – WPF) a webové (ASP.NET), až po mobilní a aplikace orientované na služby (ASP.NET Web Services, Windows Services a Windows Communication Foundation), viz [40].

2.5.2 Jazyk Visual C#

Programovací jazyk Visual C# je jedním z jazyků podporovaných .NET Frameworkem. Společnost Microsoft jej vytvořila specificky pro účely této platformy a obvykle vydává novou verzi jazyka s každou její generací. Patří do skupiny programovacích jazyků odvozených z jazyku C. Jeho syntaxe se proto velice podobá například jazyku Java. Nejrůznější jeho prvky však naznačují inspiraci i mnoha dalšími jazyky. Z C++ si propůjčil přetěžování

[16]CIL – Common Intermediate Language

[17]JIT – Just-in-Time



Obrázek 2.10: Kompilace v .NET Frameworku – převzato z [30]

operátorů a vytváření struktur a výčtů. Od VB6[18] převzal způsob definování veřejných třídních vlastností, anglicky properties, namísto tradičních metod typu getter a setter, známých třeba z Javy. Nabízí také funkce pocházející především z funkcionálních programovacích jazyků (například LISP), kterými jsou lambda výrazy a anonymní typy. C# však pouze nepřejímá existující stavební prvky, ale přidává i své vlastní. Unikátní knihovna LINQ[19] umožňuje transformovat a vytvářet dotazy nad daty různých struktur, ať už se jedná o obyčejná pole, seznamy, XML dokumenty, či data uložená v SQL databázi.

Jazyk C# se podle [59] vyznačuje následujícími klíčovými vlastnostmi:

- Programy psané v C# nevyžadují, ale umožňují, manipulaci s přímými ukazateli (pointry).
- Garbage collector zajišťuje automatickou správu paměti.
- Jsou definovány formální konstrukty pro třídy, rozhraní, struktury, výčty a delegáty.
- Existuje jednoduchý způsob přetěžování operátorů.

[18]VB6 – Programovací jazyk Visual Basic 6.0 byl vytvořen jako jednodušší a přístupnější alternativa k C a C++. Umožňoval pohodlnější tvorbu aplikací, včetně uživatelských rozhraní i knihoven kódu, a zároveň zapouzdřoval složitosti aplikačního programového rozhraní Windows. Za jeho největší nedostatek lze považovat fakt, že se nejednalo o plně objektově orientovaný jazyk, ale pouze o objektově založený [58].

[19]LINQ – Language Integrated Query

- Je podporováno programování s využitím atributů, v Javě známých jako anotace, které umožňují dodatečné specifikování funkčnosti tříd a jejich prvků.
- Je možné vytvářet generické typy a jejich členy.
- Jeden typ může být za pomoci klíčového slova „partial“ definován na mnoha místech kódu, a to i v samostatných zdrojových souborech.
- Existuje možnost doplňovat funkčnost existujících typů bez nutnosti dědění za použití rozšiřujících metod.
- Přítomnost lambda operátoru (`=>`) ulehčuje práci s delegáty.

Troelsen v [59] detailně popisuje všechny tyto a mnohé další vlastnosti nejnovější verze jazyka s označením C# 5.0.

Jazyk C# jsem si zvolil pro implementaci aplikace pro jeho bohatost, syntaktickou čistotu a především kvůli množství zkušeností, které s ním mám. S jazykem Visual Basic .NET, který se pro tyto účely nabízí jako jediná vhodná alternativa, jsem nikdy ve větší míře nepracoval. Od jeho výběru mě mimo to nejvíce odrazuje jeho, dle mého názoru, velice nepřehledná a „upovídaná“ syntaxe.

2.5.3 Webové aplikace v prostředí .NET

Microsoft .NET Framework v současné době poskytuje čtyři hlavní technologie pro vývoj webových aplikací založených na této platformě, kterými jsou ASP.NET, ASP.NET MVC, ASP.NET Dynamic Data a Silverlight.

Poslední zmíněný produkt se svou podstatou nejvíce podobá platformě Flash od společnosti Adobe. Hodí se zejména pro tvorbu webových aplikací a jejich částí, které vyžadují přehrávání multimédií nebo zobrazování komplexních dvourozměrných a třírozměrných grafických prvků. Žádná z těchto vlastností nebude ve vyvíjené aplikaci přítomna, a proto se technologií Silverlight nebudu dále zabývat. Případní zájemci se o ní mohou více dočíst například v [29].

ASP.NET Dynamic Data je framework určený pro snadný a rychlý vývoj aplikací orientovaných na data (v angličtině se označují slovním spojením „data-driven“). Ve spojení s knihovnou LINQ umožňuje ze zadaných databázových schémat generovat hotové webové aplikace, které nabízejí funkce pro prohlížení, úpravu, vkládání a mazání dat. Výsledný kód je postaven na plně přizpůsobitelných šablonách a komponentách. Dynamic Data nabízí také validační funkce založené na omezeních definovaných ve schématu databáze a filtrování dat vycházející z cizích klíčů relací. Ani tento framework

není pro mé potřeby vhodný a více prostoru mu ve své práci tudíž nebudu věnovat. Více se o něm lze dočíst například v [55].

V další části kapitoly představím a porovnám platformy ASP.NET a ASP.NET MVC a uvedu důvody, které mě vedly k výběru druhé z nich.

2.5.4 ASP.NET

První verze ASP.NET se objevila spolu s příchodem .NET Frameworku a představovala v té době revoluční nástroj pro tvorbu webových aplikací. Zatím poslední verze ASP.NET 4.5 byla vydána v roce 2012. Základ ASP.NET tvoří samotný .NET Framework, se kterým je platforma plně integrována a může využívat všech jeho předností včetně základní knihovny tříd. Nad tímto základem je postavena vrstva ASP.NET, která zajišťuje hostování .NET aplikací na serveru IIS[20] a umožňující interakci s HTTP dotazy a odpověďmi. Poslední vrstvu tvoří ASP.NET Web Forms, které představují skupinu komponent a programovací model pro tvorbu stavových uživatelských rozhraní. Stránky jsou ve Web Forms modelovány jako sestava vnořených a propojených objektů. Poté, co server obdrží dotaz na konkrétní stránku, je nejprve vytvořena instance objektu, který ji reprezentuje. Jeho následnou činností je naplnění stránky všemi ostatními objekty, které představují její dílčí prvky. Takto sestavená skupina objektů prochází specifickým životním cyklem, jenž se skládá z událostí navazujících v pevně daném pořadí. Po zpracování stránky je sestaven a odeslán zpět výsledný HTML kód a objekty jsou uvolněny z paměti serveru, viz [30].

Autoři ASP.NET se zavedením popsaného modelu, a především způsobem tvorby uživatelského rozhraní, snažili o odstínění vývojářů od bezestavové podstaty HTTP i nutnosti znát HTML, které v době představení platformy nebylo příliš rozšířené [16]. Zmíněné dílčí prvky stránek jsou v ASP.NET implementovány takzvanými HTML Controls a Web Controls. Jedná se o objektové reprezentace tradičních HTML značek, jako jsou například formulářová pole, obrázky a popisky, ale i komplexní ovládací prvky, z kterých lze uvést kalendář, navigační menu, nebo stromovou strukturu. Každá komponenta umožňuje úplnou manipulaci se svým obsahem, vlastnostmi i vzhledem prostřednictvím kódu. Navíc si uchovává informace o svém stavu i v průběhu více dotazů a odpovědí zaslaných mezi klientem a

[20] IIS – Internet Information Services je webový server od společnosti Microsoft, který je součástí většiny distribucí operačních systémů Windows. Poskytuje veškeré funkce, které lze v dnešní době u webových serverů považovat za standardní. Zmínit lze například integrované zabezpečení a podporu pro FTP (File Transfer Protocol) či správu elektronické pošty. Charakteristická je pro IIS především vestavěná podpora pro provozování ASP.NET webových aplikací [59].

serverem. Sama si také zajišťuje správné vyrenderování příslušného HTML a propojení událostí na straně klienta s odpovídajícím blokem programu na straně serveru. Web Forms tak v podstatě představují abstraktní vrstvu navrženou k tvorbě klasických, událostmi řízených, grafických uživatelských rozhraní ve webovém prostředí. Cílem autorů bylo vývoj internetových aplikací co nejvíce přiblížit vývoji stavových desktopových aplikací na platformě Windows Forms [16].

ASP.NET mimo HTML a Web Controls nabízí také tyto další nástroje, viz [30]:

- Master stránky (Master pages): Umožňují tvorbu společných šablon, které lze využít pro neomezené množství konkrétních stránek.
- Motivy: Představují způsob jak definovat společný vzhled všem prvkům z jednoho místa.
- Navigace: K dispozici jsou nástroje pro definování navigačních struktur v podobě map stránek (site maps), které mohou následně být napojeny na komponenty reprezentující různá menu, drobečkovou navigaci a podobně.
- Zabezpečení a správa uživatelských účtů: Zahrnuta je podpora pro správu bezpečnosti včetně autentizace a autorizace uživatelů na základě rolí. Součástí je i sada souvisejících komponent.
- Ovládání pro datové zdroje (Data source controls): Vybrané komponenty umožňují přímé napojení na datové zdroje v podobě polí, seznamů, ale i databázových tabulek.
- Ajax: ASP.NET 3.5 přineslo rozsáhlou podporu pro Ajax[21], cože je technologie umožňující pomocí kódu na straně klienta zaslat dotaz na server a obdrženou odpověď použít pro aktualizaci pouze části stránky.
- Automatická detekce klientských zařízení: ASP.NET dokáže automaticky rozeznat klientská zařízení, ze kterých byl dotaz na server zaslán, a na základě toho přizpůsobit vygenerovaný HTML kód, aby byla zajištěna veškerá požadovaná funkčnost.
- Správa cache: Je nabízena nativní podpora pro ukládání sestaveného HTML do cache, která je k dispozici v několika podobách.

[21] Ajax – Dříve psáno velkými písmeny jako AJAX, protože se jednalo o akronym slov „Asynchronous JavaScript and XML“. Dnes už však vnímáno jako samotné slovo [30].

ASP.NET prošlo za dobu své více než desetileté existence značným vývojem. Jeho podstata založená na stavovém UI se však nezměnila. Mnoho vývojářů začalo toto původně zamýšlené ulehčení vnímat jako nedostatek a obtíž. Freeman v [16] uvádí následující hlavní nedostatky platformy ASP.NET. Z vlastní zkušenosti s platformou musím s mnohými souhlasit:

- Paměťová a přenosová náročnost View State: View State představuje způsob uložení stavu všech UI komponent na straně klienta. Jedná se o zakódovaný řetězec znaků, který je na každé vygenerované stránce uložen do HTML značky `<input type="hidden">`. Ze své podstaty musí být zasílán s každým dotazem zpět na server, kde je rozkódován, použit při sestavování objektů a nakonec s odpovědí opět zaslán zpět. Tato data však mohou běžně dosahovat velikosti několika stovek kilobytů, což ani v dnešní době rychlého internetu není zanedbatelné.

- Životní cyklus stránek: Mechanismus zajišťující propojení událostí na straně klienta se serverovými událostmi se v některých případech může jevit jako velice záhadný a nepochopitelně selhávající. Mnozí vývojáři pak často zdlouhavě a neúspěšně hledají příčiny vzniklých problémů.

- Chybná interpretace zamýšleného oddělení zájmů: Autoři se snažili o oddělení navzájem nesouvisejícího HTML kódu pro definici vzhledu a kódu pro zachycení a zpracování událostí například při stisknutí tlačítka. Tato druhá část označovaná jako „code-behind“ je psána do tříd reprezentujících jednotlivé stránky. Úplnému oddělení prezentace a logiky se jim však nepodařilo docílit, neboť vývojáři jsou často nuceni manipulovat s prvky stránek pomocí kódu.

- Omezená možnost ovlivnit generované HTML: Vývojáři aplikací mají velice omezené možnosti, jak ovlivnit HTML kód, jejž renderuje každá komponenta umístěná na stránce. Teprve ASP.NET 4 zajišťuje vygenerování kódu, který je v souladu se standardem XHTML. Nicméně ani to nezaručuje, že bude v takové podobě, jakou si vývojáři přejí. Například nemají žádnou možnost, jak ovlivnit hodnoty atributů `id`.

- Nespolehlivá abstrakce: Vývojáři při snaze implementovat vlastní chování prvků narážejí na nutnost opustit od abstrakce HTTP, kterou ASP.NET nativně nabízí a musejí k němu pracně přistupovat neobvyklými metodami.

- Omezená možnost testování kódu: ASP.NET není nijak uzpůsobené pro tvorbu automatických jednotkových či integračních testů a jejich implementaci dělá mnohem složitější, či ji zcela znemožňuje.

2.5.5 ASP.NET MVC

Cesta k ASP.NET MVC

Výše uvedené nedostatky platformy ASP.NET přinutili její autory zamyslet se nad modernizací, jejíž provedení bylo nezbytné, chtěl-li si Microsoft udržet dobré postavení na poli webových aplikací. Zatímco se platforma stále držela stejných konceptů stanovených hned na počátku, vývoj webu i tendence ve vývoji software šly rychle kupředu, viz [16].

Rozšíření mobilního internetu přineslo nutnost vytvářet webové aplikace kompatibilní s dalšími zařízeními a prohlížeči. To vedlo k rozšíření webových standardů v oblasti HTML, CSS, JavaScriptu a podobně, a především ke kladení většího důrazu na jejich dodržování. Architektura REST[22] v oblasti distribuovaného prostředí začala pomalu vytlačovat do té doby převládající SOAP, který v ASP.NET sloužil jako hlavní nástroj pro práci s webovými službami. REST je založen na odlehčeném modelu, který aplikaci popisuje jako seskupení zdrojů reprezentujících entity. Zdroj je v tomto kontextu pouze zkrácený význam pro URI[23]. K manipulaci se zdroji se používá výhradně protokol HTTP a jím poskytované metody související s konkrétními operacemi (například GET pro čtení, POST pro vkládání, PUT pro editaci a DELETE pro smazání). Webové stránky a aplikace v dnešní době nesestávají pouze z HTML souborů. Čím dál častěji jsou mezi klientem a serverem zasílána data ve formátu XML a JSON, nejen kvůli velikému rozšíření technologie Ajax.

Nezanedbatelný rozvoj bylo možné zaznamenat i v oblasti metodik vývoje software. Programátoři se od těžkých metodik začali uchylovat k agilnímu způsobu vývoje, který je založen na snaze dosáhnout kvalitního výsledku v co nejkratším čase bez nutnosti zdlouhavých příprav a plánování. Pro tyto účely mohou používat ověřené nástroje a postupy [16]. Veliké oblibě se těší zejména metodiky založené na testování kódu, konkrétně vývoj řízený testy[24] a vývoj řízený chováním[25]. Jsou založené na postupu, kdy je nejprve popsáno požadované chování programu pomocí testu. Následně je implementována pouze ta část kódu, která zajistí jeho splnění. Nesmí však samozřejmě dojít k porušení dříve definovaných testů. Existuje mnoho nástrojů pro .NET Framework určených výhradně k usnadnění tohoto způsobu vývoje aplikací.

Jak je vidět z popisu uvedeného výše, chápání podstaty webu a jeho možností se za poslední dobu výrazně změnilo. Vývojáři se začali obracet

[22]REST – Representational State Transfer
[23]URI – Uniform Resource Identifier
[24]TDD – Test-Driven Development
[25]BDD – Behaviour-Driven Development

zpět k jeho jádru – protokolu HTTP. Jedna z prvních technologií reflektujících tuto tendenci byly Ruby on Rails. Jedná se o open source framework postavený na programovacím jazyce Ruby. Mezi jeho přednosti patří využití návrhového vzoru MVC[26], orientace na HTTP protokol, převzetí principu „konvence má přednost před konfigurací“[27] a integrace nástroje pro objektově-relační mapování[28] do jeho jádra. Více o Ruby on Rails je možné se dočíst v [18, 19, 51].

2.5.5.1 Framework ASP.NET MVC

Společnosti Microsoft se na kritiku ASP.NET a úspěch Ruby on Rails podařilo odpovědět až v roce 2007, kdy byla představena a na trh uvedena první verze frameworku ASP.NET MVC postavená na jádru platformy ASP.NET. Od té doby byla zatím třikrát aktualizována. Dosud poslední čtvrtá verze byla uvolněna v roce 2012. ASP.NET MVC přináší kombinaci návrhové vzoru model-view-controller, nejnovějších myšlenek z oblasti agilního vývoje software a nejlepší a osvědčené části z ASP.NET. Klade důraz na čistou architekturu, soulad s návrhovými vzory, testovatelnost a především se naopak od tradičního ASP.NET nesnaží zakrýt, jak web funguje [16]. Dále uvádím hlavní výhody, které dle [16] ASP.NET MVC přináší:

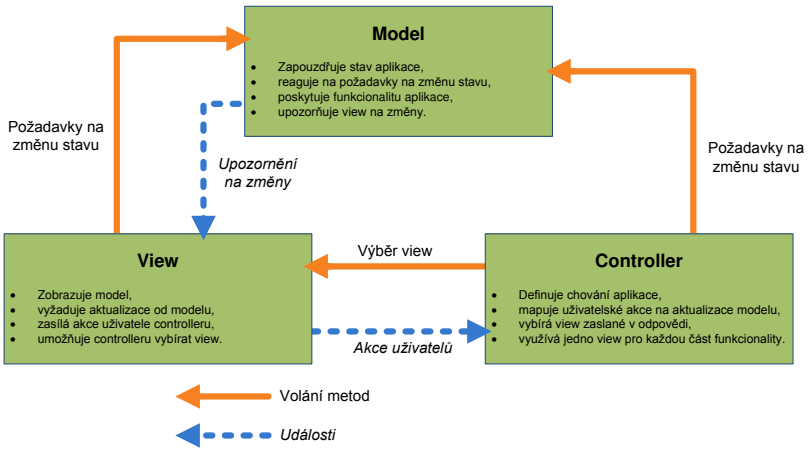
MVC architektura
Framework ASP.NET MVC implementuje architektonický návrhový vzor MVC, čímž je dosaženo značně lepšího oddělení zájmů, než v případě klasického ASP.NET. Vzor MVC sestává ze tří oddělených objektů, což zvyšuje znovupoužitelnost a flexibilitu celého řešení. Model obstarává logiku aplikace, view představuje prezentaci výsledků na obrazovce a controller předzpracovává příkazy od uživatele a definuje způsob, jak na ně uživatelské rozhraní reaguje, viz [50]. Vztahy mezi těmito objekty mohou být dobře ukázány na následujícím cyklu (viz také obrázek 2.11): Uživatel vykoná nějakou akci, aplikace v odpovědi na ni upraví model a vrátí uživateli příslušné aktualizované view.

Rozšiřitelnost

[26] MVC – Model-View-Controller

[27] Convention over configuration – (do češtiny lze přeložit jako „Konvence má přednost před konfigurací“) je návrhové paradigma založené na myšlence, že výhodnější je dodržovat a využít jasně dané postupy, zákonitosti a zjednodušení (konvence) před nutností vše složitě konfigurovat.

[28] ORM – Object-Relational Mapping



Obrázek 2.11: Schéma vzoru MVC – převzato z [4]

Framework je sestaven z nezávislých komponent, které jsou propojené pouze prostřednictvím pečlivě definovaných rozhraní nebo abstraktních tříd. Veškeré konkrétní třídy je možné nahradit vlastním kódem. Vývojářům jsou v podstatě nabídnuty tři možnosti: mohou použít výchozí implementaci, definovat novou třídu děděním od výchozí a pouze pozměnit její chování nebo vytvořit zcela novou implementaci od základu. Tímto způsobem je například možné upravit chování směrovacího systému nebo poskytovatele controllerů.

Zaručená kontrola nad HTML a HTTP

Framework ASP.NET MVC na rozdíl od ASP.NET podporuje tvorbu čistého HTML, jehož vzhled je definován pouze pomocí kaskádových stylů[29]. Nabízí vlastní šablonovací systém, který umožňuje tvorbu view obsahujících minimální množství programové logiky. View je možné provázat s objekty – modelem a libovolně do sebe zanořovat. Implementace veškerého programového chování na straně klienta pomocí JavaScriptu je závislá pouze na vývojáři. Framework je připraven na provázání s knihovnou jQuery, kterou rozšiřuje o mnoho funkcí, jež lze využít mimo jiné pro validaci modelů.

Vývojáři mají plnou kontrolu nad dotazy a odpověďmi zasílanými mezi klientem a serverem a mohou využívat jejich objektové reprezentace. Odpovědi je možné zaslat v libovolném formátu včetně HTML, XML a JSON, což velice usnadňuje implementaci Ajaxových prvků aplikací.

Testovatelnost

[29] CSS – Cascading Style Sheet

Testování je v ASP.NET MVC vývojářům aplikací ulehčeno na dvou úrovních. Jednotlivé části aplikace jsou rozděleny do nezávislých softwarových komponent. Každá z komponent je navíc navržena tak, aby co nejblíže odpovídala požadavkům testovacích nástrojů. Snadné je i použití nástrojů pro automatické testování uživatelských rozhraní, protože struktura výsledného HTML kódu je vždy podle představ vývojáře.

Směrovací systém

Framework ASP.NET MVC poskytuje nástroje zajišťující vývojářům úplnou kontrolu nad URL[30] schématem aplikace. Jednotlivé stránky a další zdroje je možné lehce zpřístupnit pod čistými, čitelnými a snadno zapamatovatelnými URL adresami, které jsou také prvním důležitým krokem k optimalizaci pro internetové vyhledávače. Navíc ukrývají technické detaily adresářové struktury aplikace, kterou je tak možné libovolně měnit bez obav o ztrátu funkčnosti.

Využití nejlepších částí ASP.NET

Z platformy ASP.NET a samotného .NET Frameworku si toto rozšíření přineslo možnost programovat v libovolném podporovaném jazyce a přistupovat k celému obsahu základní knihovny tříd. Mezi další zděděné vlastnosti patří zabezpečení včetně autorizace a autentizace, internacionalizace, profily a podobně. Zachována byla také podpora ze strany webového serveru IIS včetně jednoduchého a rychlého nasazení aplikace na cílový server.

Moderní API

Každá nová verze ASP.NET MVC je upravena tak, aby odpovídala standardům nejnovější generace .NET Frameworku a využívala všechny jeho přednosti a inovace, jako jsou například rozšiřující metody, lambda výrazy, anonymní a dynamické typy a knihovna LINQ.

Distribuce pod open source licencí

Zdrojový kód ASP.NET MVC je volně dostupný na stránkách `http://aspnetwebstack.codeplex.com` a může být libovolně upraven podle potřeb každého vývojáře. Framework je vydáván pod licencí Microsoft Public License[31].

[30] URL – Uniform Resource Locator

[31] Ms-PL – Microsoft Public License – viz `http://opensource.org/licenses/ms-pl.html`

2.5.6 Výběr webového frameworku pro implementaci

Svou práci jsem se rozhodl implementovat ve frameworku ASP.NET MVC, protože jsem s ním dosud měl pouze lepší a pozitivnější zkušenosti než s druhou variantou. Při tvorbě klasických ASP.NET aplikací jsem obvykle dříve nebo později narazil na situaci, kdy se požadavky musely přizpůsobit možnostem platformy nebo řešení problému vyžadovalo krkolomné obejití zavedených pravidel. Naopak framework ASP.NET MVC vždy nabídl více možností řešení problematických situací, a to především díky své otevřenosti a neskrývání podstatných součástí webu jako jsou HTTP dotazy a odpovědi. Upřednostňuji také naprostou svobodu při vývoji před množstvím dostupných rozšíření a prvků UI, která jsou pro ASP.NET k dispozici. Osobně si nakonec také myslím, že mnou zvolená platforma má větší budoucnost než tradiční ASP.NET, a proto rád využiji každou příležitost se zdokonalit v práci s ní.

V době, kdy jsem začínal aplikaci implementovat, bylo vydání čtvrté verze ASP.NET MVC teprve připravováno. Ve své práci proto používám třetí verzi tohoto frameworku.

2.5.7 jQuery

jQuery je rychlá, málo objemná a na funkce bohatá knihovna postavená na jazyce JavaScript [56], který je nejrozšířenějším skriptovacím jazykem pro webové stránky. JavaScript za posledních několik let nabyl veliké popularity, protože vzrostl zájem o bohaté internetové aplikace, známé také pod zkratkou RIA[32], a o technologii Ajax. Problémem je, že různé prohlížeče mohou kód napsaný v tomto jazyce interpretovat rozdílně. Knihovny jako jQuery se tento a mnohé další problémy snaží řešit a odstínit tak vývojáře od nutnosti psát skripty přizpůsobené na míru každému prohlížeči.

jQuery nabízí jednoduché, ale velice bohaté API, které především usnadňuje procházení a manipulaci s objektovým modelem dokumentu[33]. Pro výběr elementů na stránce používá syntaxi selektorů založenou na CSS3 a dále rozšířenou o vlastní funkce. Manipulace s vybranými elementy je prováděna pomocí příkazů, které lze za sebe libovolně řetězit. Takovýto druh rozhraní se nazývá fluent API (plynulé API) a jeho autorem je Matin Fowler [15],

[32]RIA – Rich Internet Application – Je webová aplikace, která se vzhledem, funkčností i použitelností snaží co nejvíce přiblížit desktopovým aplikacím.

[33]DOM – Document Object Model – Objektový model dokumentu je podle [62] platformě i jazykově nezávislé rozhraní umožňující programům a skriptům dynamicky přistupovat k a měnit obsah, strukturu a styl (především HTML) dokumentů. Dokument může být dále zpracován a výsledky zpracování mohou být promítnuty zpět do prezentované stránky.

který jej blíže popisuje v [14]. jQuery dále umožňuje například zachytávání a zpracovávání událostí a tvorbu animací. V neposlední řadě poskytuje funkce usnadňující Ajaxová volání. Veškeré funkce, které jsou vývojářům k dispozici, fungují bez rozdílu na všech nejrozšířenějších prohlížečích. O knihovně jQuery, jejíž slogan zní „write less, do more“, je možné se více dočíst například v [3].

Další předností knihovny jQuery je její rozšiřitelnost. Na stránce `http://plugins.jquery.com/` je volně k dispozici veliké množství pluginů, kterými je možné doplnit základní rozhraní o požadované funkce. V případě neexistence rozšíření vývojářům nic nebrání ve vytvoření vlastního a v jeho sdílení s ostatními programátory.

Knihovna jQuery byla integrována do frameworku ASP.NET MVC. Přímo pro jeho potřeby bylo autory připraveno několik rozšíření knihovny, které obstarávají především validaci formulářových polí na straně klienta a Ajaxová volání. Osobně velice oceňuji právě funkce pro validaci formulářů. Použijeme-li atributy ze jmenného prostoru System.ComponentModel.DataAnnotations k definování podmínek, které musejí splňovat jednotlivé členy modelu, framework zajistí automatické vygenerování příslušného kódu a napojení validačních funkcí na straně klienta.

2.5.8 jQuery UI

jQuery UI rozšiřuje jádro jQuery o sadu interakcí s uživatelským prostředím, efektů, widgetů a motivů vzhledu, které lze použít k vytvoření webových aplikací obsahujících dynamičtější ovládací prvky a disponující modernějším vzhledem [57]. Jedná se v podstatě o seskupení mnoha oficiálních pluginů od týmu, který vyvíjí knihovnu jQuery.

Rozšíření, která jQuery UI přináší, lze rozdělit do tří hlavních kategorií:

- Efekty: Rozšiřují množinu efektů poskytovaných jQuery o nové a doplňují některé již existující efekty převážně o možnosti animace. Jedná se například o postupné skrytí a zviditelnění elementů, postupnou aplikaci nebo změnu CSS třídy, postupnou změnu barvy a podobně.

- Interakce: Přinášejí uživatelům nové možnosti jak interagovat s prvky stránek pomocí myši. Součástí je přenášení prvků („drag and drop“), změna velikosti, jejich označování a řazení.

- Widgety: Obsahují sadu často užívaných ovládacích prvků, které jsou běžnou součástí desktopových aplikací, ale neexistuje jejich nativní implementace v prostředí webu. Patří sem například tlačítka, pole pro výběr data, dialogová okna, menu, taby a tooltip.

Nepostradatelnou součástí jQuery UI jsou předdefinované kaskádové styly, které zajišťují správnou funkčnost prvků. Styly je samozřejmě možné upravit a vzhled prvků tak přizpůsobit celkovému designu webové aplikace. Podrobnější informace o jQuery UI jsou uvedené například v [3, 63].

jQuery UI není zdaleka jediným frameworkem nabízejícím popsané funkce. Z populárních open source alternativ lze jmenovat například MooTools (`http://mootools.net/`) nebo Dojo Toolkit (`http://dojotoolkit.org/`). K dispozici jsou také placené platformově nezávislé frameworky, které však nabízejí mnohem více možností. Příkladem může být Kendo UI (`http://www.kendoui.com/`) od společnosti Telerik poskytující vlastní jQuery widgety, programovací rozhraní, mapování datových zdrojů, validaci, internacionalizaci, motivy, šablony a podobně. Kendo UI obsahuje i pomocné rozšiřující metody vytvořené přímo pro framework ASP.NET MVC.

Pro implementaci uživatelského rozhraní aplikace jsem se rozhodl použít framework jQuery UI, protože práce s ním je jednoduchá a rychlá, framework je zdarma i pro komerční využití a myslím si, že obsahuje veškeré komponenty, které budu potřebovat.

2.5.9 Jazyk T-SQL

Transact-SQL (T-SQL) je jazyk patřící do rodiny strukturovaných dotazovacích jazyků (SQL). Přesněji se jedná o proprietární rozšíření standardizovaného jazyka SQL. Byl vyvinut společností Microsoft a Sybase speciálně pro potřeby relačních databázových serverů Microsoft SQL Server a Sybase Adaptive Server Enterprise. Zatím poslední verze Microsoft SQL Server nese označení 2012.

T-SQL je deklarativní programovací jazyk, což znamená, že programátor zápisem kódu definuje pouze požadovaný cíl, kterého má být dosaženo. Počítač, v případě SQL databázový stroj, si sám určí nejlepší cestu k tomuto cíli. Druhou skupinu tvoří imperativní programovací jazyky (C++, C#, Java a podobně). V jejich případě musí programátor určit nejen výstup programu, ale také počítači zadat instrukce jak ho krok za krokem dosáhnout [5]. T-SQL má všechny standardní vlastnosti jazyka SQL, které lze rozdělit do následujících podmnožin:

- Dotazování (Querying): K dotazování T-SQL používá tradiční příkaz `SELECT` rozšířený o mnoho volitelných klauzulí.
- Manipulace s daty (DML[34]): DML je podmnožinou jazyka SQL, jejíž příkazy se používají k manipulaci s daty uloženými v databázi. Čtyři

[34]DML – Data Manipulation Language

základní příkazy `INSERT`, `UPDATE`, `DELETE` a `MERGE` doplňují příkazy spojené s kurzory, které umožňují procedurální styl programování.

- Definice dat (DDL[35]): DDL slouží především ke tvorbě a úpravě struktury databáze, tedy provádění operací s tabulkami. Řadí se sem příkazy `CREATE`, `ALTER` a `DROP`.
- Kontrola nad daty (DCL[36]): Účelem DCL je aplikovat různá omezení na přístup uživatelů k tabulkám a dalším databázovým objektům. Obsahuje variace příkazů `GRANT` a `REVOKE`.
- Transakční zpracování dat (TCL[37]): TCL se používá ke spouštění a potvrzování nebo vracení výsledků transakcí, což je atomická dále nedělitelná jednotka práce prováděná serverem, která dodržuje princip ACID (atomicita, konzistence, izolace a trvanlivost). Pro tyto účely slouží příkazy `BEGIN TRANSACTION`, `COMMIT` a `ROLLBACK`.

T-SQL dále rozšiřuje standard SQL o možnosti procedurálního programování, schopnost deklarování lokálních proměnných, funkce pro práci s textovými řetězci a daty, matematické funkce a upravuje výrazy `DELETE` a `UPDATE`.

K dispozici je také možnost vytvářet vlastní uložené procedury (SP[38]) a uživatelsky definované funkce (UDF[39]). Uložené procedury jsou v podstatě samostatné programy psané v jazyce T-SQL a uložené na databázovém serveru, které mohou být opakovaně spouštěny. Mohou obsahovat zápis libovolného množství příkazů manipulujících s daty i se strukturou databáze. Jejich výhodou je možnost lepšího zabezpečení a vyšší rychlost jejich vykonávání. Uživatelsky definované funkce se jim velice podobají, ale vždy vracejí nějaký výsledek buď v podobě skalární veličiny, nebo tabulkových dat. Jejich dalším omezením je, že nemohou vykonávat DML a DDL příkazy.

Podrobný popis T-SQL a manuál k jeho použití nabízí například [5] nebo [24].

[35] DDL – Data Definition Language
[36] DCL – Data Control Language
[37] TCL – Transaction Control Language
[38] SP – Stored Procedure
[39] UDF – User-Defined Function

Kapitola 3

Návrh

Ve fázi návrh jsou logické modely vytvořené během analytické části vývoje software specifikovány do podoby, kterou lze využít jako přesný podklad pro implementaci systému. Zatímco v první fázi je kladen důraz na zachycení funkcí, které systém musí poskytovat, v této etapě je určen způsob, jak je přesně implementovat. Na systém již není nahlíženo z pohledu uživatelů a jejich potřeb, ale z pohledu možností zvolených technologií, knihoven tříd, úložišť dat a podobně, viz [2].

V kapitole představím celkovou architekturu systému, jeho rozdělení do vrstev a každou vrstvu blíže popíšu. Podrobně se budu věnovat vrstvě doménového modelu, vrstvě určené pro přístup k datům, vrstvě služeb a prezentační vrstvě. Na závěr kapitoly pomocí sekvenčního diagramu ukážu, jak probíhá komunikace mezi objekty napříč vrstvami.

3.1 Architektura systému

Pod architekturu systému se v případě .NET webových aplikací řadí návrh tříd a jejich vzájemných vztahů, rozdělení tříd do jmenných prostorů a jednotlivých sestavení (assembly) a závislosti mezi sestaveními. Při návrhu aplikace jsem se inspiroval především radami od Milletta, který v [45] uvádí návrhové vzory a osvědčené postupy pomáhající vývojářům docílit robustních aplikačních architektur. Systém doporučuje rozdělit do vrstev s přesně danými zodpovědnostmi, mezi které se řadí vrstva business logiky (*Business Logic Layer*), vrstva aplikačních služeb (*Service Layer*), vrstva pro přístup k datům (*Data Access Layer*) a prezentační vrstva (*Presentation Layer*). Jádro aplikace má tvořit vrstva business logiky, kterou je možné organizovat čtyřmi různými způsoby:

- Transaction Script: Pro každou business transakci či operaci je obvykle vytvořena jedna metoda, která obsahuje veškerý potřebný kód včetně business pravidel, validace a persistence. Metody jsou shromážděny v jedné servisní třídě.

- Active Record: Využívá se v případě, kdy business model přesně kopíruje datový model a každá databázová tabulka tak má svou objektovou reprezentaci. Každý business objekt obsahuje implementaci svých členů, chování i operací pro uložení, čtení a mazání.

- Domain Model: Co nejpřesněji se snaží zachytit nějakou doménu z reálného světa. Definuje entity, jejich chování i validační pravidla. Konkrétní způsob persistence je od modelu abstrahován. Business model může obsahovat servisní třídy implementující chování, které nezapadá do žádné z entit modelu.

- Anemic Domain Model: Podobně jako Domain Model obsahuje třídy reprezentující modelovanou doménu. Veškeré chování je však vyčleněno z definic entit, které tak získávají podobu jednoduchých datových tříd.

Podstatu a principy objektově orientovaného návrhu nejlépe vystihuje třetí způsob organizace business modelu – Domain Model. Ostatní způsoby se více zaměřují na procedurální styl programování a v rozsáhlých aplikacích jsou obtížněji udržovatelné, viz [45].

Při návrhu aplikace jsem se nejprve snažil vytvořit business model, který by respektoval pravidla pro organizaci Domain Model, ale nakonec jsem přistoupil k organizaci, která mísí prvky Domain Model, Active Record i Transaction Script. Důvodem pro toto rozhodnutí byl především požadavek IT oddělení společnosti Magna E&I na použití uložených procedur pro veškerou komunikaci s databází. Domain Model je možné implementovat, pokud je pro persistenci dat využito objektově relační mapování[40]. ORM framework je totiž schopen zajistit automatické získání dat z databáze v momentě, kdy je s nimi pracováno, aniž by to programátor musel jakkoliv explicitně specifikovat (tzv. „lazy loading“). V případě čtení dat pomocí

[40]ORM – Object-Relational Mapping – Objektově relační mapování je technologie, která vývojáře odstiňuje od propojení objektového business modelu systému se strukturou relační databáze, kde jsou veškerá data uložena. Umožňuje jim pracovat s objektovou reprezentací dat a používat jednoduché příkazy pro jejich čtení, vytváření, ukládání a mazání. Zvolený ORM framework se sám postará o vygenerování odpovídajících databázových (nejčastěji SQL) dotazů, jejich zaslání na databázový server, přijmutí odpovědi a vytvoření struktury objektů přesně reflektující obdrženou odpověď.

uložených procedur však musejí být předem získány všechny objekty, ke kterým má být v dané operaci přistupováno. Podobně po dokončení operace musejí být zvlášť uloženy všechny objekty, které byly pozměněny.

Navrhl jsem business model, kde stejně jako v organizaci Active Record každá třída a její členy reprezentují jednu tabulku databáze. Tato struktura modelu mi umožnila použít knihovnu Entity Framework Extensions pro snadné vytváření objektů reprezentujících výsledky procedur, podrobněji viz kapitola 4.2.1. Uvnitř tříd jsem ponechal definici jejích členů a chování, které je závislé pouze na členech té třídy a nevyžaduje načtení vnořeného objektu. Ostatní chování jsem vyčlenil do samostatných servisních tříd obsahujících téměř veškerou business logiku aplikace, podobně jako v organizaci Transaction Script. Definoval jsem procedury zajišťující manipulace s daty a načítání nejčastěji používaných struktur objektů. Implementaci procedur jsem zahrnul do metod samostatných tříd, které jsou volány výhradně z metod servisních a dalších pomocných tříd.

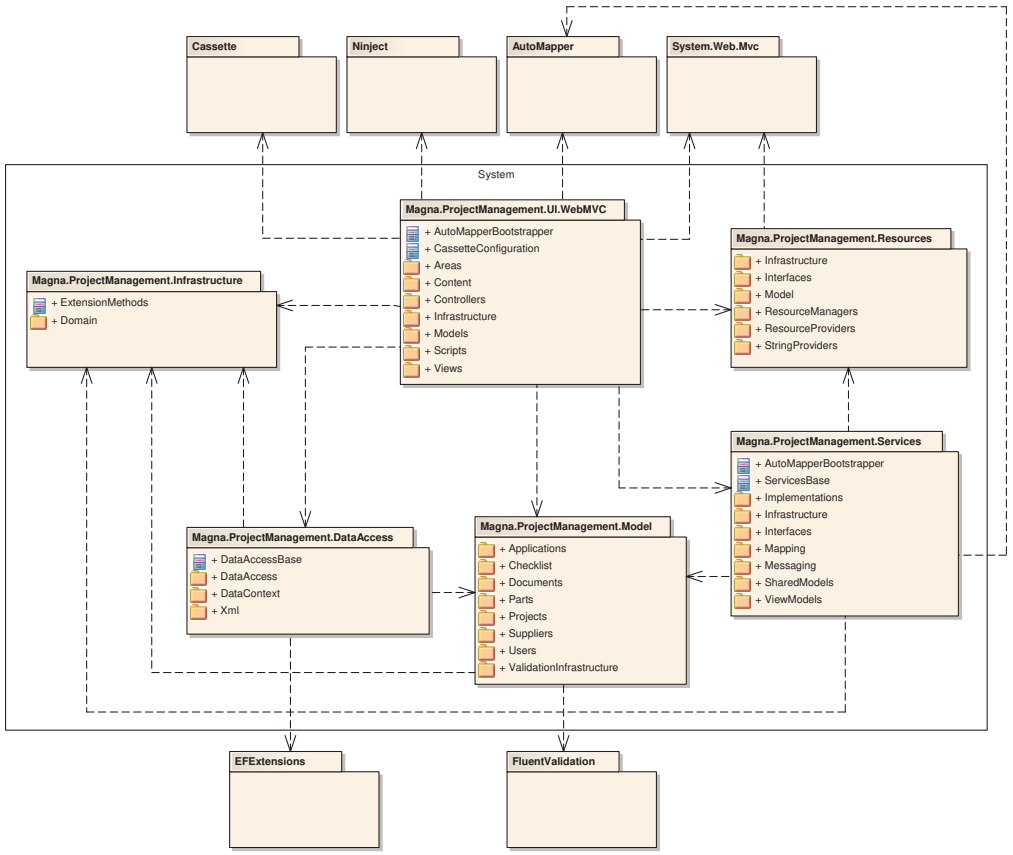
Veškerý kód aplikace jsem rozdělil do šesti samostatných vrstev, přičemž každá je implementována jako samostatné sestavení (assembly). Každé sestavení se odkazuje na co nejméně ostatních sestavení, které umožní jeho správnou funkčnost, a na další potřebné knihovny. Diagram vrstev jako balíčků a závislostí mezi nimi i důležitými knihovnami je zachycen na obrázku 3.1. Význam knihoven a důvod jejich použití uvádím v kapitole 4. Systém tvoří následující vrstvy:

- **Vrstva business modelu:** Obsahuje definice tříd, které tvoří business model aplikace. Třídy jsou podle souvislostí rozděleny do několika balíčků.
  (sestavení: Magna.ProjectManagement.Model)

- **Vrstva pro přístup k datům:** Obsahuje třídy implementující volání uložených procedur a vytváření objektů odpovídajících obdrženým výsledkům procedur.
  (sestavení: Magna.ProjectManagement.DataAccess)

- **Vrstva aplikačních služeb:** Poskytuje množinu služeb, které zajišťují veškerou aplikační logiku a udržují business model v konzistentním stavu.
  (sestavení: Magna.ProjectManagement.Services)

- **Prezentační vrstva:** Jedná se o ASP.NET MVC aplikaci s bohatým uživatelským rozhraním s Ajaxovými prvky.
  (sestavení: Magna.ProjectManagement.UI.WebMVC)

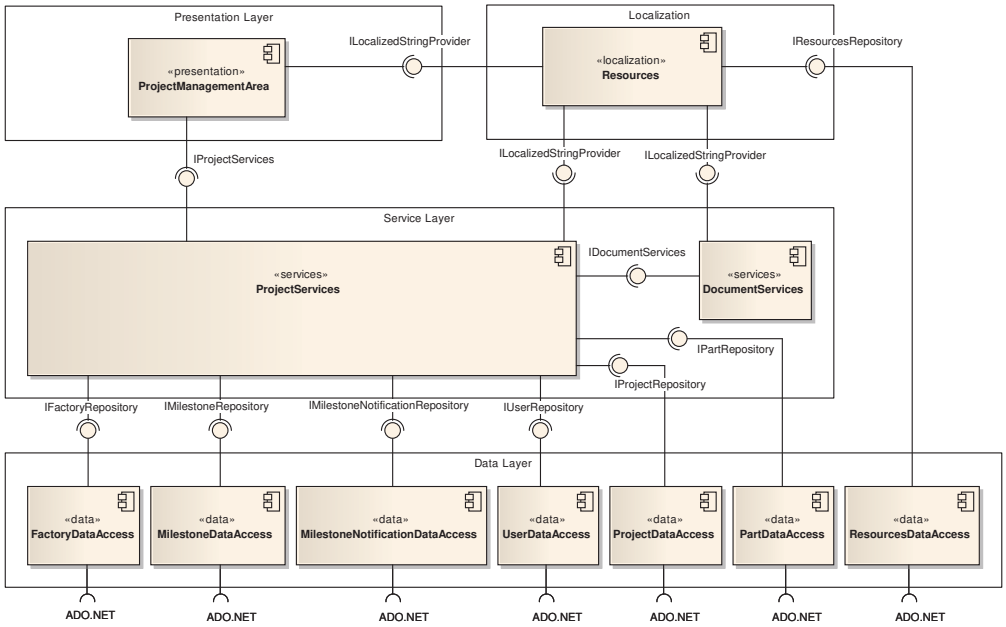


Obrázek 3.1: Závislosti mezi vrstvami aplikace a použitými knihovnami

- **Lokalizační vrstva:** Implementuje logiku obstarávající veškerou lokalizaci aplikace.
  (sestavení: Magna.ProjectManagement.Resources)

- **Infrastrukturní (podpůrná) vrstva:** Obsahuje pomocné třídy a metody používané napříč všemi ostatními vrstvami.
  (sestavení: Magna.ProjectManagement.Infrastructure)

Na servisní třídy, třídy pro přístup k datům, lokalizační vrstvu a prezentační vrstvu lze nahlížet jako na samostatné komponenty, které spolu komunikují prostřednictvím definovaných rozhraní. Takto strukturovaný komponentový systém je dle [2] flexibilnější a dílčí konkrétní implementace mohou být snadno zaměněny za jiné, které však dodržují stejná rozhraní. Ve specifikaci UML 2 [49] je komponenta definována jako „modulární část systému, která zapouzdřuje svůj obsah, a jejíž projev může být okolním



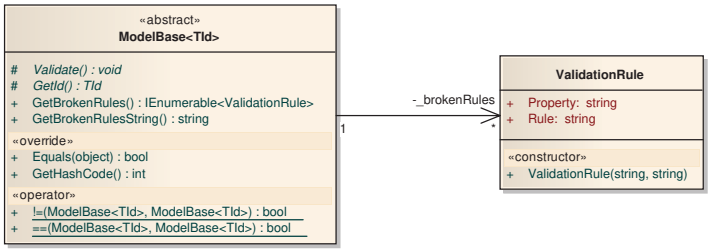
Obrázek 3.2: Diagram komponent – zachycuje pouze komponentu Project-Services, komponenty, na kterých závisí a komponenty na ní závislé

prostředím zaměněn“. Na diagramu 3.2 je na část systému nahlíženo z pohledu soustavy komponent. Jsou zde zobrazeny pouze některé komponenty spadající do vrstvy aplikačních služeb a do vrstvy pro přístup k datům. Komponenta *ProjectServices* komunikuje s několika komponentami, které obstarávají přístup k datům, dále s jinou servisní komponentou *Document-Services* a s lokalizační komponentou *Resources*. Komponenta prezentační vrstvy *ProjectManagementArea* vyžaduje přístup pouze ke komponentám servisní a lokalizační vrstvy. Pro větší přehlednost jsou v diagramu vynechány některé vztahy, například mezi *DocumentServices* a komponentami vrstvy pro přístup k datům.

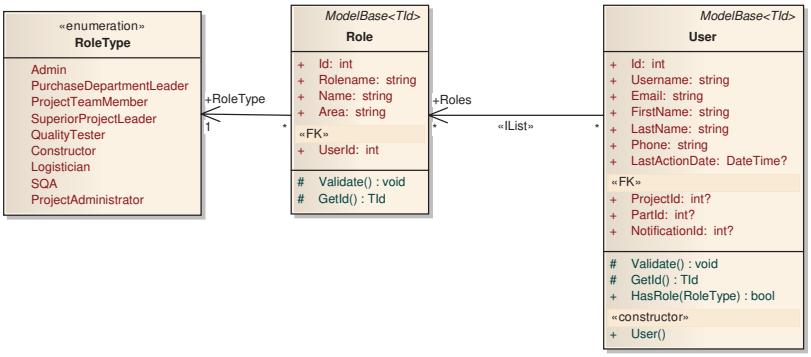
3.2 Vrstva business modelu

Při návrhu tříd business modelu jsem vycházel z konceptuálního modelu uvedeného na diagramech 2.8 a 2.9. Model jsem přizpůsobil tak, aby entity a jejich vztahy bylo možné snadno ukládat do tabulek relační databáze. Výsledný business model je zachycen na diagramech 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 a 3.9.

Názvy tříd a jejich vlastností (v jazyce C# takzvané Properties) přesně



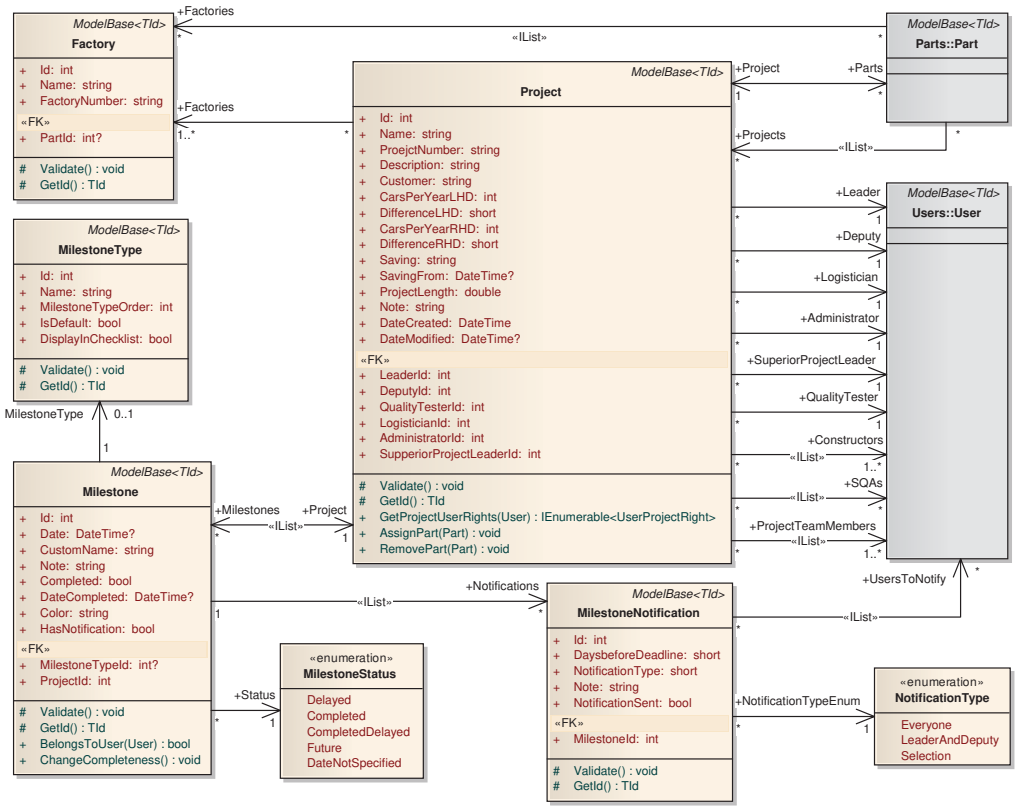
Obrázek 3.3: Diagram tříd business modelu – společná rodičovská třída



Obrázek 3.4: Diagram tříd business modelu – uživatel

odpovídají názvům databázových tabulek a jejich sloupců. Tento způsob strukturování a pojmenování mi umožnil použít knihovnu Entity Framework Extensions k automatickému vytváření objektů z obdržených výsledků volání uložených procedur, viz kapitola 4.2.1. Součástí tříd musejí být vlastnosti reprezentující cizí klíče, které umožní propojování objektů a vytváření jejich vnořených struktur. Cizí klíče však ve třídách nemají jiný význam, po sestavení struktur se již dále nevyužívají a ve třídách spíše překážejí. Rozhodl jsem se proto vytvořit a udržovat další dva velice podobné modely na úrovni vrstvy aplikačních služeb a prezentační vrstvy. Každý z těchto modelů je upraven tak, aby co nejlépe sloužil svému účelu. Význam a přizpůsobení modelů uvádím dále v kapitolách věnovaných příslušným vrstvám.

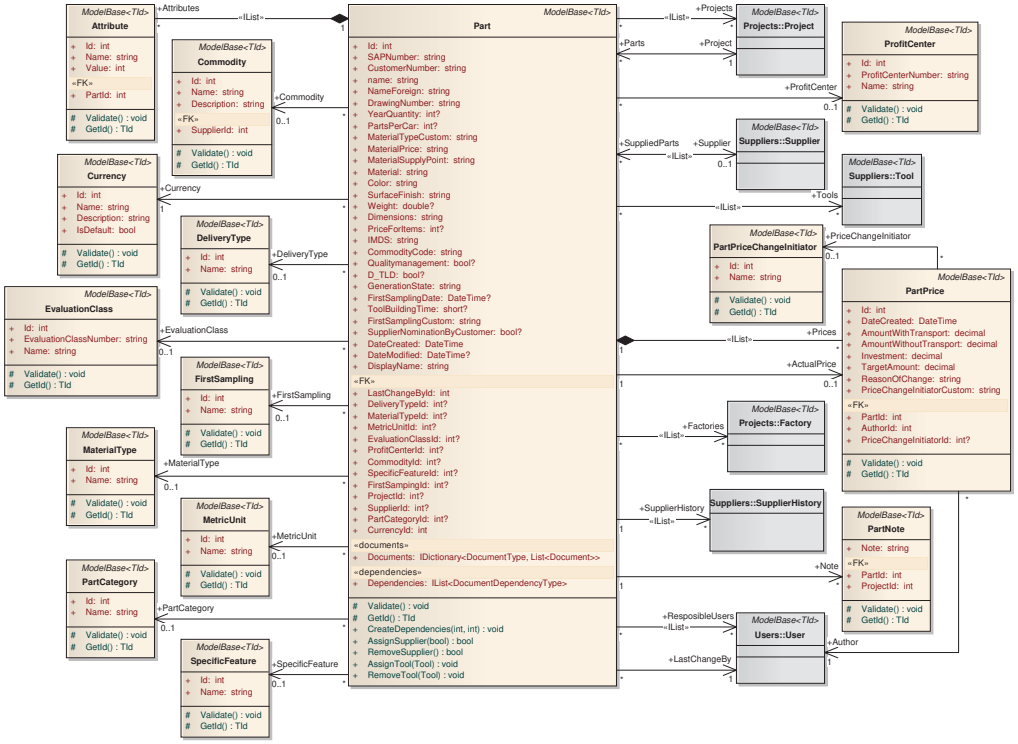
Business model je proto používán pouze na úrovni vrstvy aplikačních služeb a vrstvy pro přístup k datům. Všechny třídy, které reprezentují databázové tabulky, dědí od společného předka – generické třídy *ModelBase*. Ta s využitím návrhového vzoru Template Method poskytuje metody pro validaci a porovnávání objektů. Předepisuje dvě abstraktní metody `GetId()` a `Validate()`, které implementují všechny odvozené třídy. S pomocí metod `GetBrokenRules()` a `Validate()` je možné ověřit validnost objektů



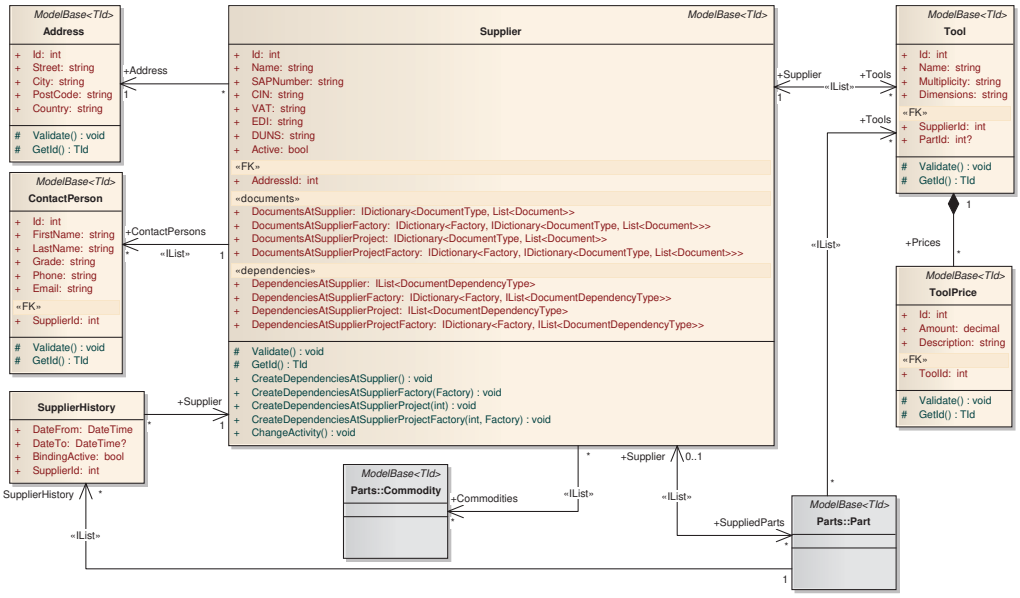
Obrázek 3.5: Diagram tříd business modelu – projekt

na úrovni business modelu. Správnost zadávaných dat je však testována především na úrovni prezentační vrstvy a jejího modelu.

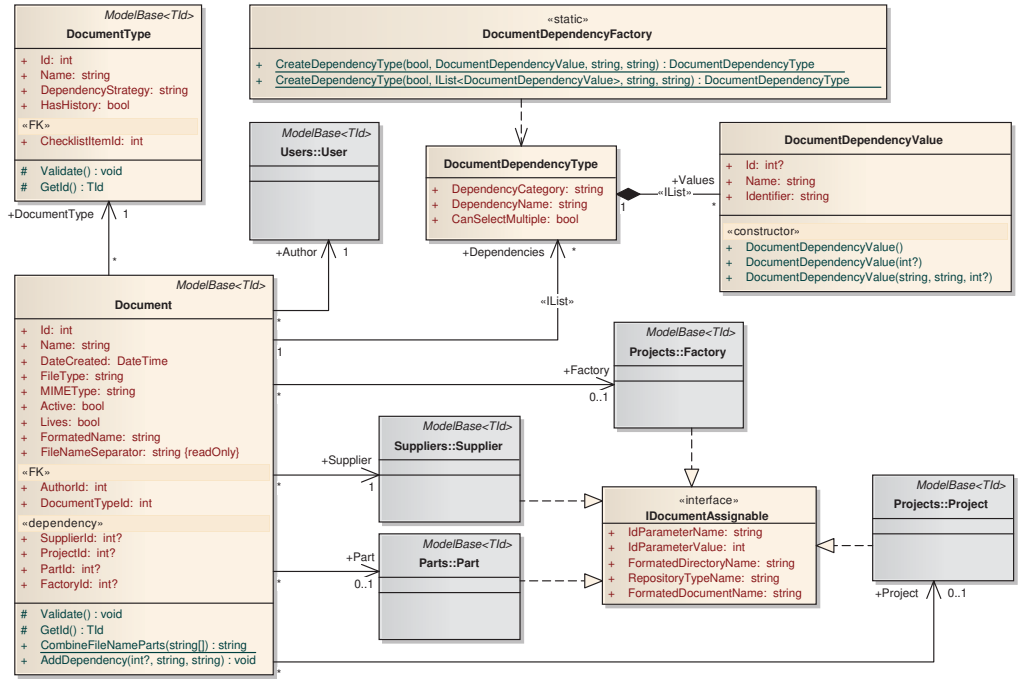
Na diagramu 3.3 je zachycena abstraktní generická třída *ModelBase*, od které je odvozena většina ostatních tříd, a třída *ValidationRule* sloužící k zachycení porušených validačních pravidel modelu. Diagram 3.4 znázorňuje třídy související s uživateli aplikace a jejich rolemi. Hlavní třídou na diagramu 3.5 je projekt (*Project*), ke kterému se váží závody (*Factory*) a termíny (*Milestone*). Diagram 3.6 zobrazuje především třídu díl (Part) a na ní navázané ceny (*PartPrice*) a další přidružené třídy. Třída dodavatel (*Supplier*) představuje stěžejní část diagramu 3.7. K dodavateli se přiřazuje adresa (*Address*), kontaktní osoby (*ContactPerson*) a zařízení (*Tool*). Na diagramu 3.8 jsou zachyceny třídy reprezentující projektovou dokumentaci. Rozhraní *IDocumentAssignable* implementují třídy, na které je možné navázat dokumenty. Konkrétně se jedná o dodavatele (*Supplier*), projekt (*Project*), závod (*Factory*) a díl (*Part*). Rozhraní předepisuje vlastnosti, které se využívají při vytváření adresářové struktury a generování názvů



Obrázek 3.6: Diagram tříd business modelu – díl



Obrázek 3.7: Diagram tříd business modelu – dodavatel

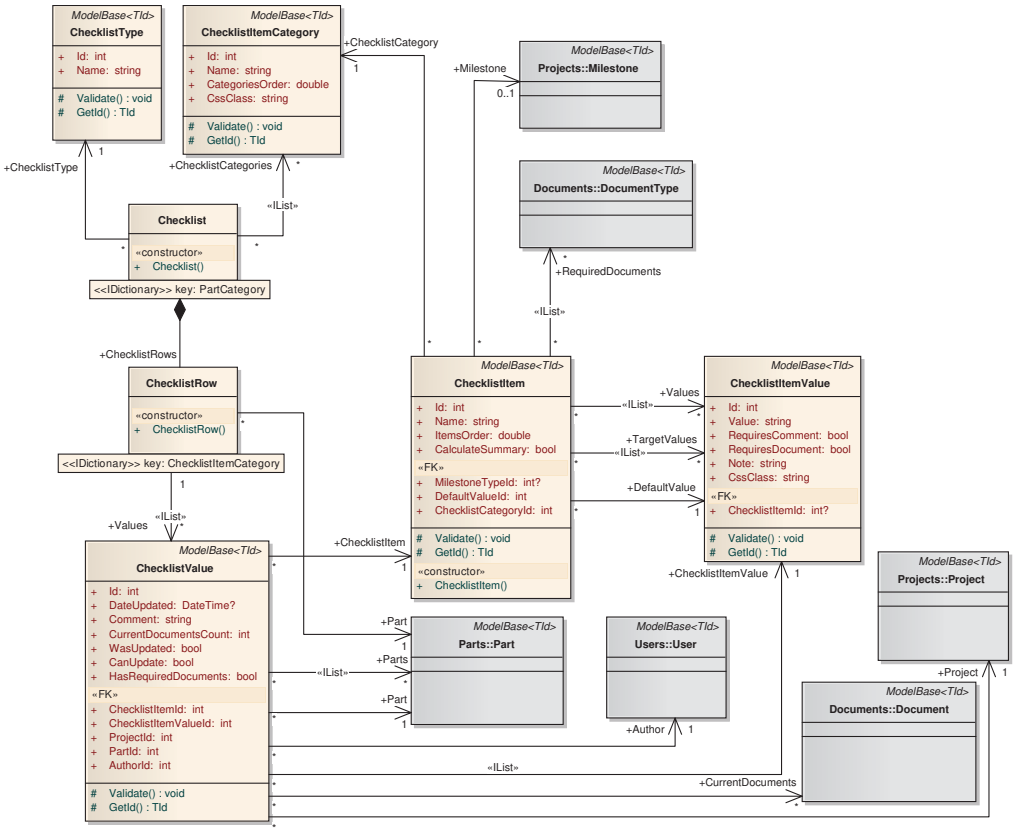


Obrázek 3.8: Diagram tříd business modelu – projektová dokumentace

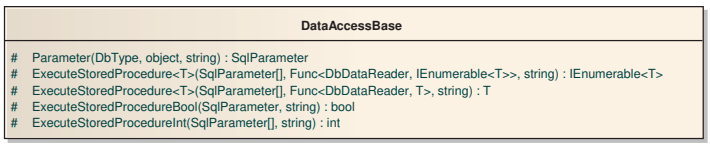
ukládaných dokumentů. O způsobu organizování dokumentů se podrobněji zmiňuji v kapitole 4.3.1. Poslední diagram 3.9 zachycuje třídy sloužící k sestavení Checklistu. Na diagramech jsou zakresleny žluté a šedé třídy. Žluté třídy vždy tvoří hlavní část diagramu. Šedé třídy jsou převzaty z jiných diagramů a slouží ke znázornění vazeb na žluté třídy.

3.3 Vrstva pro přístup k datům

Vrstva pro přístup k datům slouží výhradně ke komunikaci s datovým úložištěm a k ukládání a čtení business objektů. Zajišťuje také řízení transakcí, viz [45]. Datovým úložištěm obecně může být například databáze nebo souborový systém. V mém případě se konkrétně jedná o databázi provozovanou na databázovém serveru Microsoft SQL Server 2005. Třídy této vrstvy neobsahují žádnou aplikační logiku, implementují pouze jednoduché operace pro vytvoření, čtení, změnu a smazání business objektů. Vrstva aplikačních služeb k ní přistupuje pouze prostřednictvím definovaných rozhraní, což podporuje jeden z principů objektově orientovaného návrhu, kterým je takzvané oddělení zodpovědností (Separation of Concerns). Pro každý business objekt, který se ukládá do databáze, jsem definoval jedno rozhraní



Obrázek 3.9: Diagram tříd business modelu – Checklist



Obrázek 3.10: Diagram tříd vrstvy pro přístup k datům – společná rodičovská třída

obsahující operace související výhradně s tímto objektem. Každé rozhraní implementuje jedna třída na úrovni vrstvy pro přístup k datům. Třídy mají společného předka – třídu DataAccessBase (viz obrázek 3.10), která poskytuje metody usnadňující volání uložených databázových procedur. Existují metody pro volání procedur vracejících jeden objekt, kolekci objektů, číslo (například ID při vytváření objektů) a pravdivostní hodnotu značící, zda došlo k modifikaci alespoň jednoho řádku tabulky (například při úpravě nebo mazání objektu).

Pro navázání spojení s databází používám knihovnu ADO.NET Entity Framework, která bývá nejčastěji využívána pro funkce související s objektově-relačním mapováním. Já jsem ji však zvolil proto, že umožňuje automatické vytváření databázových spojení dle údajů zadaných v konfiguračním souboru Web.config, jednoduché volání uložených procedur a také napojení rozšiřující knihovny Entity Framework Extensions.

3.3.1 Návrh databáze

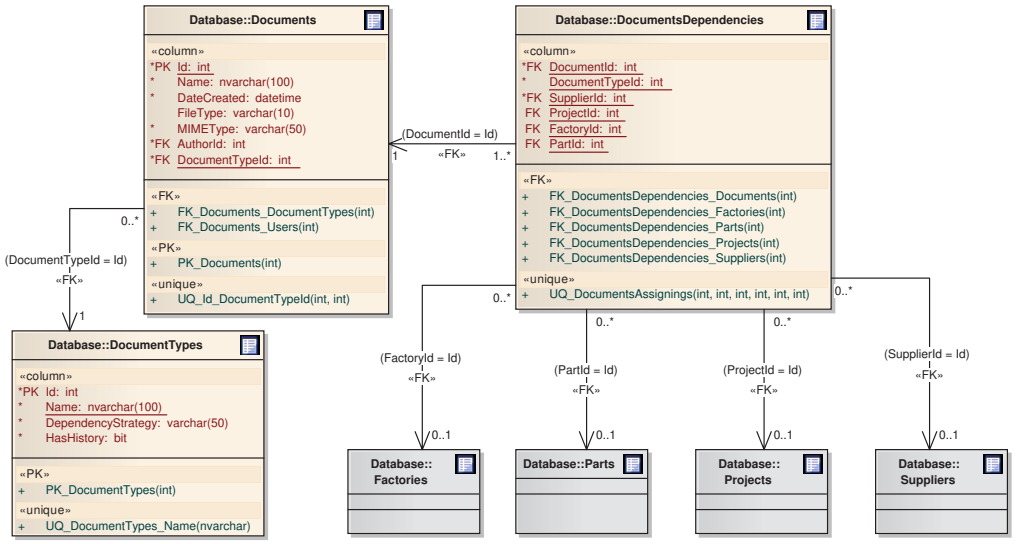
Návrh struktury databáze byl víceméně přímočarý, a proto se podrobněji zmíním pouze o části týkající se ukládání dokumentů, která byla jako jediná problematická a nejednoznačná. Přehled celé struktury rozdělené do několika diagramů lze zhlédnout v příloze F. Vzhledem k tomu, že business objekty reprezentují jednotlivé tabulky, je struktura databáze velice podobná struktuře business modelu.

Projektová dokumentace se ukládá na podnikový server do adresářové struktury, kterou aplikace sama automaticky spravuje. Správu adresářů a souborů podrobněji popisuji v kapitole 4.3.1. Aby bylo možné dokumenty rychle vyhledávat a přiřazovat je k požadovaným entitám (dodavatelé, projekty, závody a díly), je nutné vést přehled nahraných dokumentů také v databázi. V tabulce 2.1 uvádím přehled čtyř možných závislostí dokumentů na entitách. Nabízely se dvě možnosti, jak závislosti uložit. První z nich je vytvoření samostatné tabulky pro každou závislost. Tabulky by obsahovaly vždy pouze vybrané sloupce. Druhou možností, kterou jsem nakonec zvolil, je vytvoření jediné tabulky pro všechny závislosti. Tabulka *DocumentsDependencies* obsahuje všechny sloupce s cizími klíči (*SupplierId*, *ProjectId*, *FactoryId* a *PartId*), které se odkazují na závislé entity. Do sloupců je možné uložit hodnotu `NULL`, pokud dokument na nějaké entitě nezávisí. Tento způsob uchování závislostí umožnil jednodušší implementaci ukládání dokumentů a zejména jejich vyhledávání, neboť není nutné vyhledávat ve více tabulkách současně. Část databázové struktury týkající se projektové dokumentace je zobrazena na diagramu 3.11.

3.3.2 Návrh uložených procedur

Výhradní používání uložených procedur pro komunikaci s databází má tu výhodu, že zabrání případným útokům typu SQL Injection[41]. Zároveň však

[41] SQL Injection je jedním z útoků na webové aplikace, který využívá neošetřených vstupů a spočívá ve vložení vlastního upraveného kódu až do databázové vrstvy aplikace, kde je posléze spuštěn a vykonán. Takový kód může systém poškodit nebo útočníkovi zobrazit skrytá data. [43]



Obrázek 3.11: Diagram tříd vrstvy pro přístup k datům – projektová dokumentace

vyžaduje vytvoření procedury pro každou operaci, která má být systémem prováděna. Existuje možnost vytvářet dynamické SQL dotazy uvnitř procedur. Tato technika je však nebezpečná, neboť je náchylná právě na útoky SQL Injection a procedury navíc ztrácejí na své výkonnosti, protože si databázový server nemůže uložit jejich prováděcí plán.

Pro načítání některých business objektů jsem vytvořil několik verzí procedur. Jedna procedura slouží k načtení dat pouze z tabulky příslušející danému objektu. Ostatní procedury načítají business objekt i s dalšími na něm závislými objekty, což znamená, že procedura vrací více výsledků neboli „result setů“. Pro jejich čtení jsem využil knihovnu Entity Framework Extensions, viz kapitola 4.2.1. Různé verze procedur demonstruji na příkladu čtení entity reprezentující díl (*Part*):

- `Part_GetPartBasics`: Procedura vyhledá díl podle zadaného ID dílu a přečte data pouze z tabulky *Parts*.
- `Part_GetPartBasicsBySAPNumber`: Procedura vyhledá díl podle zadaného SAP čísla dílu a přečte data pouze z tabulky *Parts*.
- `Part_GetPartDetail`: Procedura vyhledá díl podle zadaného ID dílu a přečte data z tabulky *Parts*. Dále vyhledá údaje o uživateli, který díl naposledy upravil, přečte údaje o dodavateli dílu a jeho adrese, vyhledá související údaje o přiřazené třídě ocenění, typu materiálu,

metrické jednotce apod., vyhledá ceny dílu, jejich zadavatele a původce, přečte zařízení přiřazená k dílu a jejich ceny a vyhledá všechny projekty, ve kterých se díl aktuálně vyskytuje.

- `Part_GetPartDetailInProject`: Procedura vyhledá stejná data jako procedura `Part_GetPartDetail` kromě všech projektů. Navíc podle zadaného ID projektu přečte závody, které jsou dílu v daném projektu přiřazeny, vyhledá nákupčí zodpovědné za díl v tomto projektu, přečte poznámku a kategorii, do které je díl v projektu zařazen a vyhledá všechny dokumenty závisející na daném dílu, projektu a dodavateli dílu.

Každá procedura je implementována jako jediná transakce, jejíž výsledky jsou potvrzeny příkazem `COMMIT TRANSACTION` pouze v případě, kdy jsou správně provedeny všechny její části. Pokud tedy procedura sestává z několika samostatných T-SQL příkazů a v průběhu jejího vykonávání dojde k chybě na některém z nich, je databáze vrácena do stavu, ve kterém byla před spuštěním procedury.

3.4 Vrstva aplikačních služeb

Vrstva aplikačních služeb představuje rozhraní umístěné mezi prezentační vrstvou a vrstvou business modelu. Definuje veškeré možnosti aplikace dostupné uživatelům, viz [45]. Je tvořena pomocí návrhového vzoru Facade, jehož účelem je ukrytí složité logiky za jednoduché rozhraní, ke kterému přistupuje klient. Na úrovni vrstvy aplikačních služeb probíhá koordinace mezi business objekty, jejich načítání z datového úložiště a ukládání zpět, jejich validace, vykonávání business logiky a autorizace uživatelů ke kontrole jejich oprávnění volat dané operace.

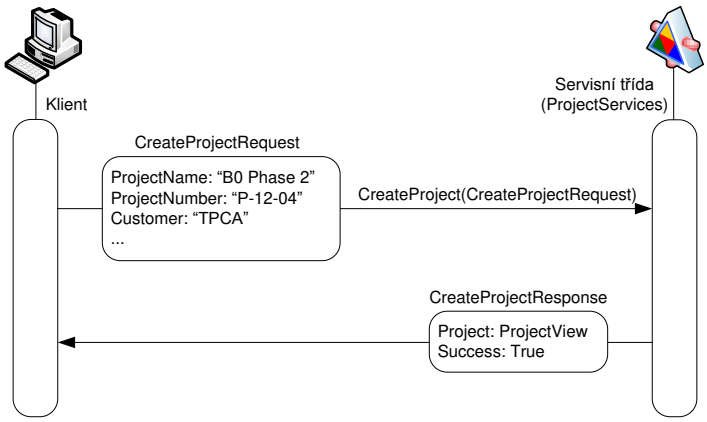
Veškerou logiku jsem rozdělil do několika servisních tříd, které implementují rozhraní s požadovanými metodami. Prezentační vrstva komunikuje s vrstvou služeb pouze přes ně. Servisní třídy seskupují operace týkající se kategoricky souvisejících business objektů. Uvnitř operací dochází k autorizaci uživatele volajícího operaci, odchytávání a zaznamenávání výjimek, validaci vstupu a objektů, aby bylo zabráněno chybám a přechodu aplikace do nekonzistentního stavu a koordinaci business logiky a akcí souvisejících s persistencí business objektů. Granularitu operací jsem volil na takové úrovni, aby byly co nejvíce znovupoužitelné, ale zároveň, aby klient (například prezentační vrstva) nebyl zodpovědný za jejich správnou koordinaci. Operace by jinými slovy měly být autonomní a jejich vnitřní logika

by neměla být závislá na tom, zda před nimi byla provedena jiná operace. V aplikaci jsou k dispozici následující servisní třídy:

- **ApplicationServices:** Obsahuje operace související s prostředím aplikace jako je třeba sestavení položek navigace v závislosti na uživatelské roli přihlášeného uživatele.
- **DocumentServices:** Obsahuje operace starající se o ukládání a zpětné stahování dokumentů a přiřazování a odstraňování závislostí dokumentů. Koordinuje také logiku spravující adresářovou strukturu projektové dokumentace.
- **ChecklistServices:** Obsahuje operace související s generováním a správou Checklistu.
- **PartServices:** Obsahuje operace týkající se správy dílů, jejich cen a souvisejících atributů. Nabízí také operace pro přiřazení (odstranění) dílu k projektu, přiřazení (odstranění) dodavatele k dílu a přiřazení (odstranění) zařízení k dílu.
- **ProjectServices:** Obsahuje operace související se správou projektů, termínů a závodů.
- **ReportServices:** Obsahuje operace vytvářející různé přehledy projektů, dílů a termínů.
- **SearchServices:** Obsahuje operace sloužící k podrobnému vyhledávání dokumentů, dílů a dodavatelů.
- **SupplierServices:** Obsahuje operace související se správou dodavatelů a zařízení.
- **UserServices:** Obsahuje operace související se správou uživatelských účtů a rolí.

3.4.1 Messaging Pattern

Volání operací servisních tříd klientskou stranou jsem navrhl za použití vzoru pro zasílání zpráv, takzvaný Request-Response Messaging Pattern, viz [45]. Jedná se o vzor uplatňovaný především při návrhu servisně orientovaných aplikací. Já jsem se jej však rozhodl použít i zde hlavně pro jeho elegantnost a zjednodušení, které s sebou přináší. Každá zpráva, dotaz i odpověď, zasílaná mezi klientem a vrstvou aplikačních služeb je reprezentována speciálně vytvořenou třídou, která obsahuje všechny požadované



Obrázek 3.12: Schéma vzoru Request-Response Messaging Pattern – částečně převzato z [45]

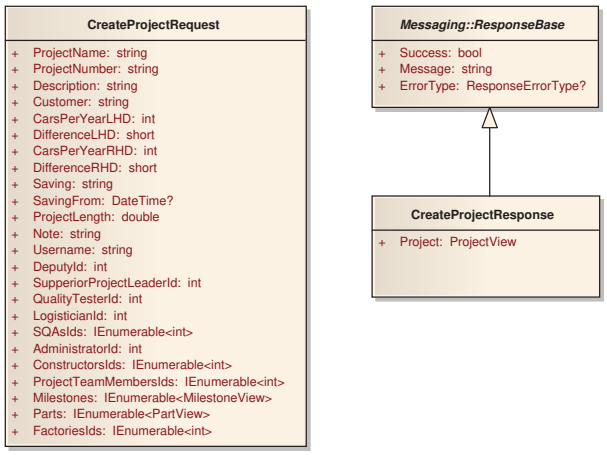
vlastnosti. Odpadá tak například nutnost vytvářet přetěžované metody. Na obrázku 3.12 je zachyceno grafické schéma vzoru.

Každá třída představující odpověď rozšiřuje základní třídu, která obsahuje příznak úspěšného vykonání operace a případně zprávu informující o vzniklé chybě. V odpovědích nejsou klientovi zasílány objekty pocházející z vrstvy business modelu, ale speciálně upravené objekty, které obsahují jen potřebné vlastnosti a často postrádají jakékoliv operace. Tyto třídy tvoří model nazvaný ViewModel, který svou strukturou téměř totožně kopíruje business model. Třídy však neobsahují metody spojené s business logikou a například vlastnosti představující cizí klíče. Na diagramu 3.13 jsou zachyceny třídy reprezentující dotaz a odpověď pro operaci vytvoření projektu.

Výše popsaná architektura vrstvy aplikačních služeb připravuje aplikaci na případnou integraci s jinými systémy společnosti Magna E&I za pomoci webových služeb. Použité návrhové vzory Facade a Request-Response Messaging již v podstatě definují rozhraní služeb. Zbývalo by tedy implementovat vrstvu obstarávající jejich přijímání a zasílání a vytvořit reprezentace tříd zpráv ve vhodném formátu, například XML nebo JSON.

3.5 Prezentační vrstva

Základ prezentační vrstvy tvoří webový framework ASP.NET MVC. Vrstva je navázaná na všechny ostatní vrstvy, ale skutečně využívá pouze třídy z vrstvy aplikačních služeb a lokalizační vrstvy. Reference na ostatní vrstvy existuje pouze pro potřeby principu Dependency Injection (nebo také Inversion Of Control), jehož význam popisuji v kapitole 4.4.1.



Obrázek 3.13: Diagram tříd reprezentujících dotaz a odpověď pro operaci vytvoření projektu

Vrstva obsahuje především třídy a další prvky, které předepisuje použitý framework. Není zde obsažena žádná business logika, kromě validace vstupů a objektů. Veškerá ostatní logika souvisí s přizpůsobením modelů pro potřeby zobrazení na HTML stránkách. Strukturu frameworku ASP.NET MVC tvoří především následující prvky: Controllers, Models, Views, Areas a různá rozšíření.

Controllers

Controllery jsou třídy rozšiřující základní třídu Controller, které obsahují metody představující akce volané uživateli. Každá akce vrací nějaký výsledek, kterým může být klasické view představující HTML stránku, částečné (Partial) view obsahující pouze fragment HTML kódu, přesměrování na jinou akci, data ve formátu JSON, soubor nebo cesta k souboru a podobně.

V každém controlleru by měly být umístěny pouze nějakým způsobem vzájemně související akce. Rozhodl jsem se je rozčlenit podobně jako servisní třídy, tedy podle příslušnosti ke skupině business objektů. Jsou však členěny detailněji a existuje jich tedy více než servisních tříd.

Models

Modely jsou klasické třídy, které mají v ASP.NET MVC aplikacích reprezentovat především business entity. Každému view může být předán nejvýše jeden tento model a následně v něm libovolně pomocí HTML kódu zobrazen. Já jsem však zvolil složitější architekturu a business model jsem umístil do zcela samostatné vrstvy. Na úrovni prezentační vrstvy ale i tak rozlišuji

dva typy modelů.

První z nich, takzvaný UIViewModel, je třetí a poslední variací business modelu. Podobně jako ViewModel na úrovni vrstvy aplikačních služeb svou strukturou téměř totožně kopíruje výchozí business model. Třídy jsou však opět přizpůsobené, tentokrát pro potřeby jejich zobrazení na stránkách. Některé jejich vlastnosti tak slouží k naformátování výstupu složeného z jiných vlastností. Vlastnosti tříd, které při vytváření objektů zadávají uživatelé, a je tedy nutné kontrolovat jejich korektnost, jsou opatřeny atributy z knihovny DataAnnotations. Tyto atributy předepisují požadovanou formu hodnot vlastností a framework se jimi dokáže řídit při validování modelu automaticky vytvářeného z vyplněných formulářových polí. Framework navíc obsahuje hotové jQuery skripty, které zajistí vytvoření a navázání atributů využívaných pro automatickou validaci formulářových polí na straně klienta.

Druhý model jsem vytvořil speciálně pro potřeby jednotlivých view. Většina view totiž zobrazuje složitější strukturu než pouze samostatnou business entitu. Vytvořil jsem tedy jednoduché třídy, jejichž jediným účelem je shromažďovat několik business entit a dalších vlastností, které je nutné zobrazit, nebo kterými se zobrazování řídí. Třídy jsou předávány dílčím view.

Views

Views jsou klasické HTML stránky rozšířené o schopnost interpretace kódu zapsaného v libovolném .NET programovacím jazyce. Mohu být přiřazená konkrétní akci určitého controlleru nebo sdílená napříč celou aplikací. Jak jsem již uvedl výše, každému view může být přidělen nejvýše jeden model (třída), který má zobrazit. Framework nabízí funkci libovolného vnořování view, a je tedy možné definovat znovupoužitelná view, která jsou zobrazena na více místech uživatelského rozhraní aplikace. Odpadá tím také nutnost udržovat případný duplicitní kód.

Pro tvorbu všech view jsem využil jazyk HTML 5. Tuto nejnovější verzi HTML jsem zvolil především proto, že umožňuje definování vlastních atributů ke všem elementům [61]. Jedná se o takzvané data-* atributy, kde * je nahrazena vlastním požadovaným názvem atributu. Slouží k ukládání vlastních dat, která nemají žádný vliv na zobrazení elementu, ale mohou být přečtena pomocí jazyka JavaScript. Další technologií, kterou jsem použil k zápisu view, je šablonovací systém Razor [42]. Poprvé se objevil v ASP.NET MVC 3, kde doplnil výchozí ASPX systém převzatý z klasického ASP.NET. Funkcí šablonovacího systému je zpracovat obsah dokumentů a na místo definovaných instrukcí vložit nějaký dynamický kód. Výsledek je zaslán klientovi k zobrazení.

Areas
Areas jsou prostředkem k rozdělení rozsáhlých ASP.NET MVC aplikací do jednotlivých funkčních [44] skupin. Každá Area obsahuje vlastní strukturu controllerů, modelů a view. Areas jsem využil k rozdělení aplikace na sekce pro skupiny uživatelů. Do každé sekce mají přístup pouze přihlášení uživatelé s přidělenými požadovanými rolemi. Prozatím existuje administrační sekce aplikace a sekce pro oddělení projektového nákupu. V případě rozšíření aplikace do dalších oddělení podniku budou vytvořeny nové Areas.

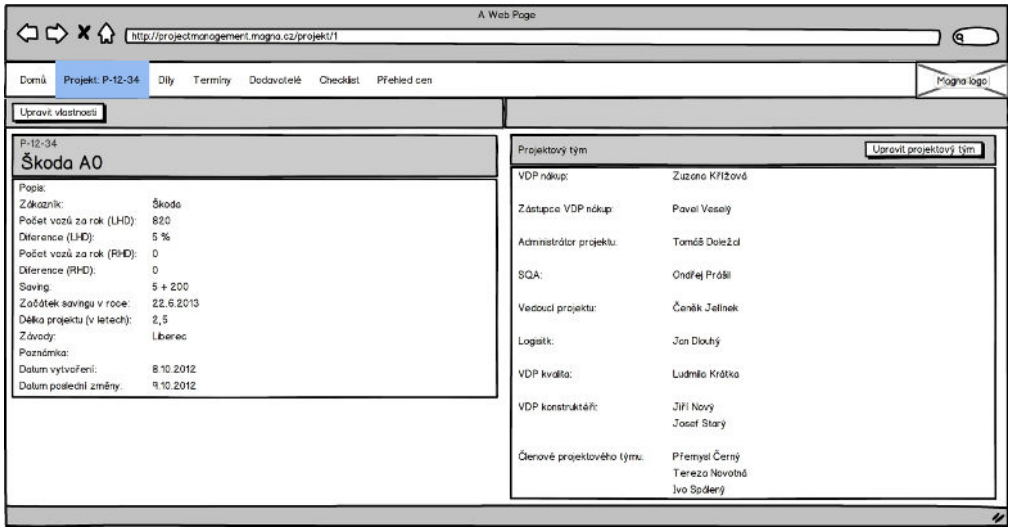
Rozšíření
Důležitou vlastností frameworku ASP.NET MVC je jeho otevřenost a rozšiřitelnost. Druhé jmenované vlastnosti jsem využil k implementaci vlastních:

- binderů (Binders), které slouží k předávání hodnot z formulářových polí do vlastností tříd,
- providerů, které zpracovávají atributy vlastností tříd,
- validátorů, které jsou volány na základě validačních atributů definovaných u vlastností tříd,
- poskytovatelů rolí (RoleProvider), které se starají o autentizaci a autorizaci,
- filtrů, které mohou být vykonávány před vybranými akcemi,
- a metod rozšiřujících třídu HtmlHelper, které slouží ke generování často používaného HTML kódu.

Detaily implementace vybraných rozšíření popisuji v kapitole 4.4.4.

3.5.1 Návrh uživatelského rozhraní

Součástí návrhu uživatelského rozhraní aplikace byl návrh navigace mezi stránkami a návrh vzhledu a rozložení ovládacích prvků. Aplikaci mají používat především zaměstnanci OPN společnosti Magna E&I, a proto proces návrhu probíhal tak, aby byly co nejlépe splněny jejich požadavky a zapracovány jejich připomínky. Z důvodu velké časové vytíženosti zaměstnanců oddělení byl počáteční návrh konzultován pouze s jedním vybraným zástupcem. Na počátku procesu mu byl předložen hrubý návrh uživatelského



Obrázek 3.14: Prototyp uživatelského rozhraní – obrazovka s detailem projektu

rozhraní ke schválení, viz příklad na obrázku 3.14, který jsem vytvořil na základě doménového modelu a požadavků na systém. Zároveň mu bylo vysvětleno, jakým způsobem bude aplikace ovládána, jak bude probíhat vytváření a úprava entit a jakým způsobem se budou uživatelé aplikací pohybovat. Poté jsem již přistoupil k implementaci, aby uživatelé měli co nejdříve k dispozici testovací verzi aplikace, na které by byli schopni odhalit nedostatky rozhraní.

Jak jsem již zmínil, vytíženost zaměstnanců OPN mi nedovolila provést testování návrhu uživatelského rozhraní. Zvolil jsem tedy alespoň techniku Nielsenovy heuristiky, kterou popisuji v kapitole 5.3.

Navigační struktura aplikace

Aplikace ve velké míře využívá technologii Ajax pro asynchronní komunikaci klienta a serveru. Veškerá úprava vlastností business entit probíhá v dialogových oknech. Po zanesení změn do formuláře v dialogovém okně a jejich uložení je okno automaticky zavřeno a provedené změny jsou zobrazeny na stránce.

Pro znázornění navigační struktury aplikace a dostupnosti různých akcí z jednotlivých stránek jsem vytvořil diagramy 3.15 a 3.16, jejichž vzhled vychází z [27], ale je přizpůsobený mým potřebám. Každý uzel diagramu představuje klasickou stránku (šedé a žluté), dialogové okno (zelené) nebo

pouhou akci (modré). Speciální případem stránek jsou stránky s výstupem přizpůsobeným pro tisk (fialové) a wizardy (červené a oranžové), které jsou použité pro rozložení složitého postupu vytváření nové entity do více kroků. Přechody mezi klasickými stránkami jsou znázorněny plnou čárou, možnost vyvolání dialogového okna nebo provedení akce ze stránky je znázorněna přerušovanou čárou. V diagramech nejsou zachycena některá propojení mezi podstránkami, která by jej učinila velice nepřehledným.

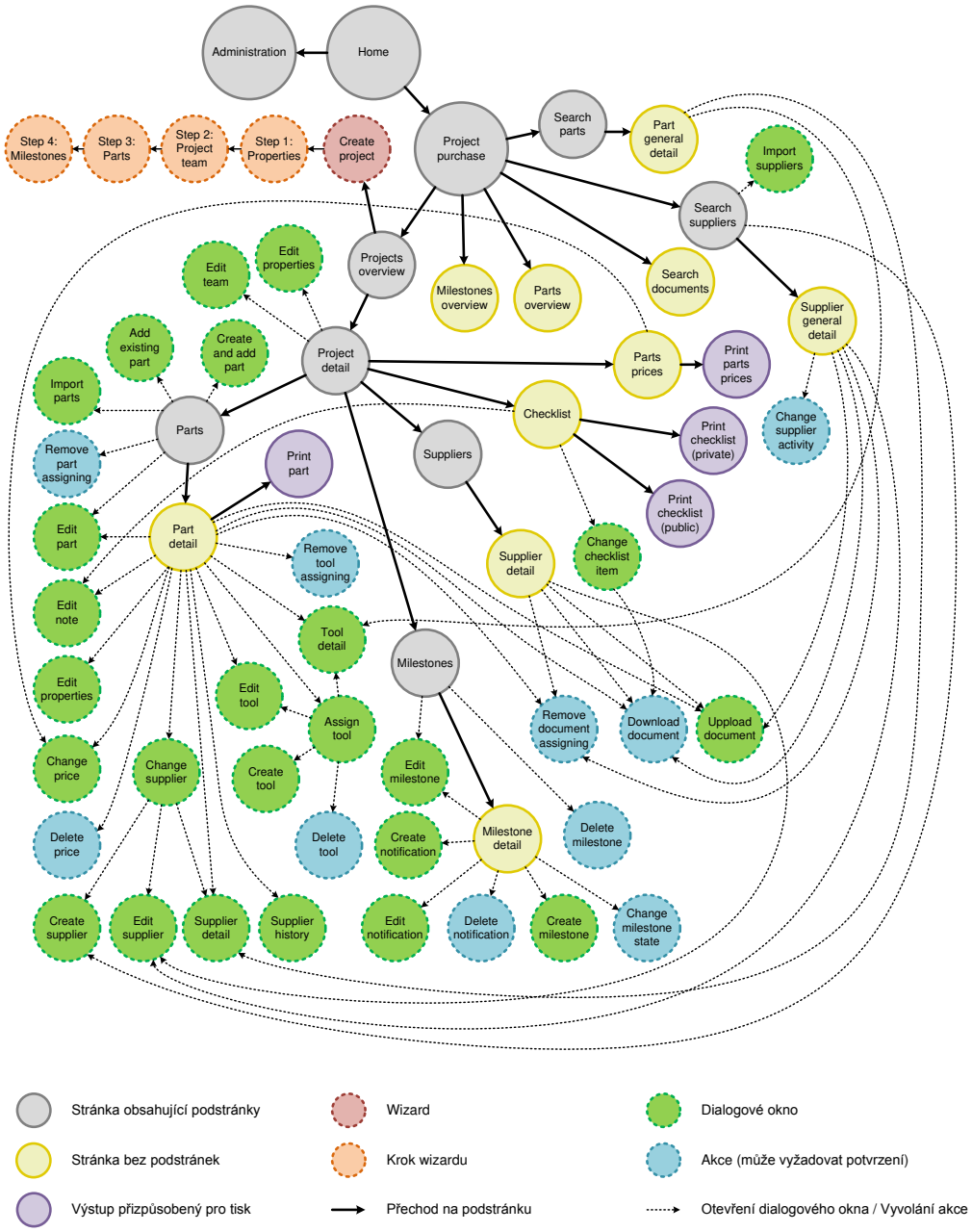
Z výchozí stránky *Home* může oprávněný uživatel přistoupit do sekce administrace (*Administration*) nebo projektového nákupu (*ProjectPurchase*). Na diagramu 3.15 jsou zachyceny veškeré podstránky sekce projektového nákupu, které byly dosud vytvořeny. Jak je z diagramu patrné, některé podstránky jsou hodně zanořené, a proto jsem navrhl speciální menu, které v sobě kombinuje prvky standardního horizontálního menu a drobečkové navigace. Zobrazené položky menu se mění na základě stránky, na které se uživatel právě nachází. Je-li aktuální stránkou detailní pohled na nějakou entitu, pak se v menu objeví položka zachycující tuto skutečnost. V kombinaci se zvýrazněním a barevným odlišením aktuálních položek má uživatel neustále přehled o tom, kde se v aplikaci právě nachází.

Existují dvě základní podoby menu, jejichž varianty jsou zachyceny na obrázcích 3.17 a 3.18. První podoba menu (obrázek 3.17) je přítomná na úvodní straně sekce projektového nákupu, na stránkách přehledu projektů, dílů a termínů, a na stránkách souvisejících s vyhledávání dokumentů, dodavatelů a dílů. Druhá podoba menu (obrázek 3.18) je zobrazena vždy, když se uživatel pohybuje v detailu projektu a s ním souvisejících entit, jako jsou například díly termíny, dodavatelé a Checklist. Horizontální menu je na některých stránkách doplněno vertikálním menu, které uživateli například umožňuje rychle se přepínat mezi detaily jednotlivých dílů, termínů a dodavatelů.

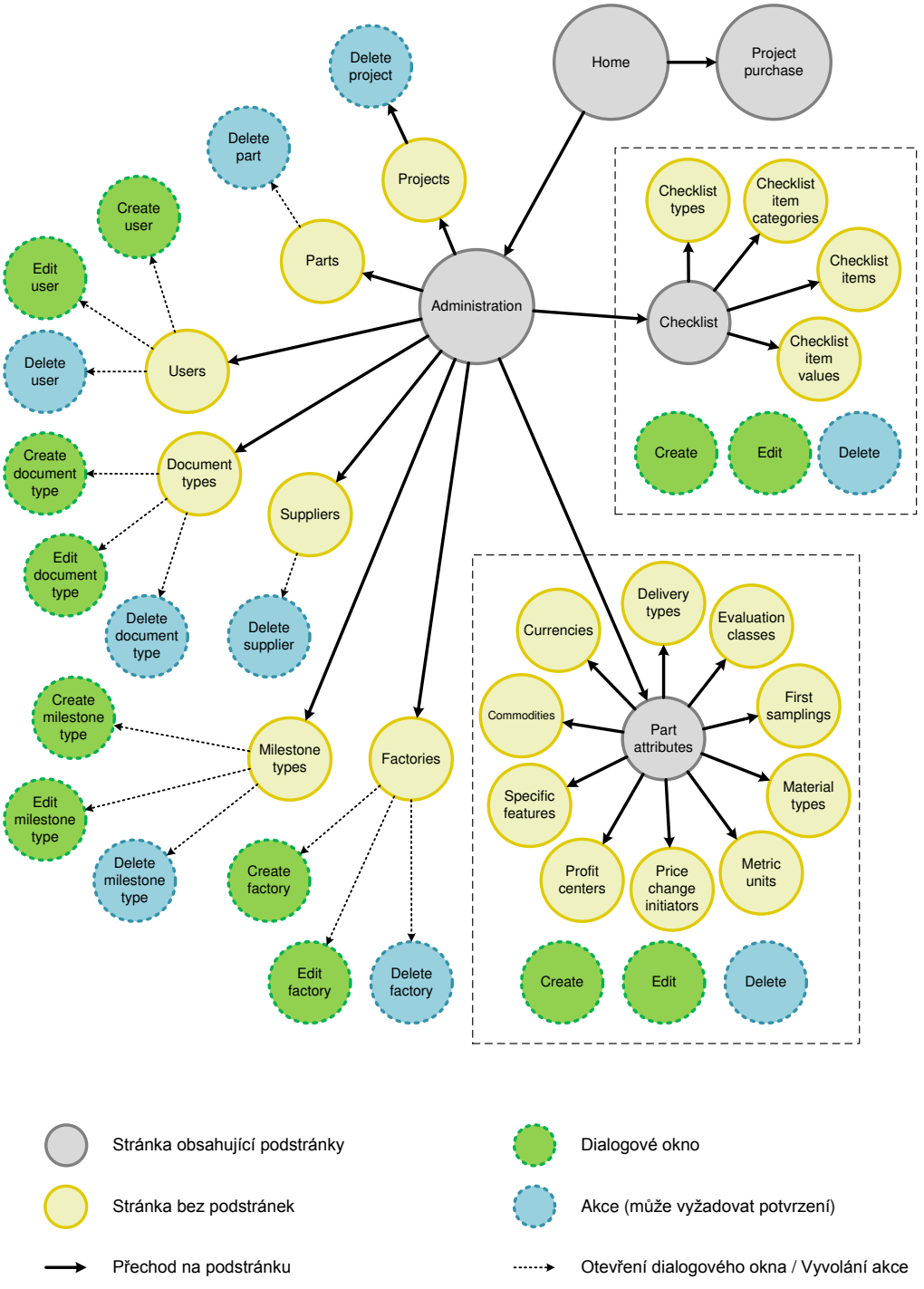
Na diagramu 3.16 je zachycen návrh navigační struktury administrační části aplikace. Stránky zde většinou představují přehledy existujících business entit s možností jejich vytváření, úpravy a mazání. V případě uzlů *Part attributes* a *Checklist* souvisí zobrazené akce se všemi žlutě znázorněnými podstránkami. Administrační sekce dosud nebyla implementována, protože bylo upřednostněno dokončení a odladění sekce projektového nákupu. Je tedy možné, že dojde ke změně uvedeného návrhu.

Použité grafické prvky

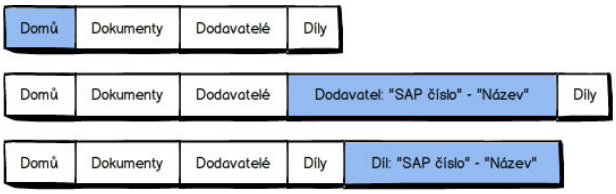
Během návrhu uživatelského rozhraní jsem se snažil používat osvědčené grafické prvky, které uvádí například [12, 11, 13]. Patří mezi ně mimo jiné:



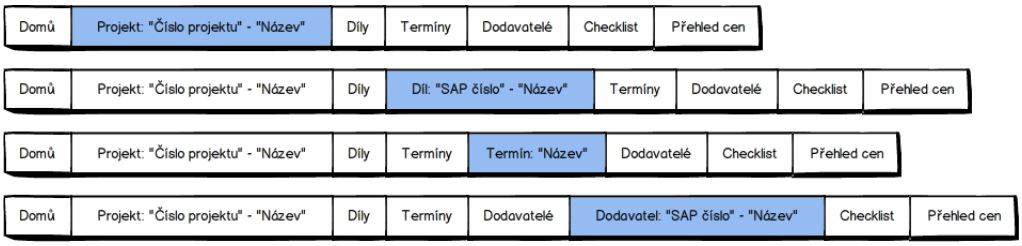
Obrázek 3.15: Diagram navigační struktury – sekce projektového nákupu



Obrázek 3.16: Diagram navigační struktury – administrační sekce

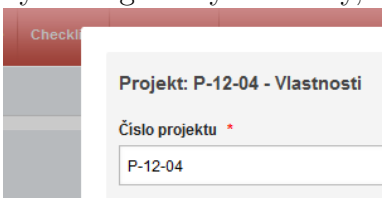


Obrázek 3.17: Prototyp hlavního menu – ukázka podoby menu na hlavní stránce, přehledu dodavatelů a přehledu dílů

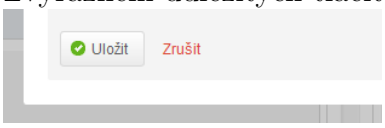


Obrázek 3.18: Prototyp hlavního menu – ukázka podoby menu na stránkách detailu projektu

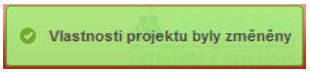
- Používání modálních dialogových oken, které jsou od zbytku stránky výrazně graficky odděleny, například ztmavením okolí.



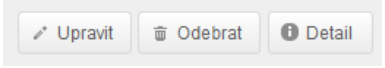
- Zvýraznění důležitých tlačítek a potlačení těch méně důležitých.



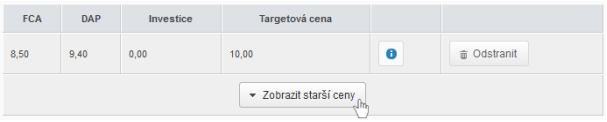
- Informování uživatele o výsledku provedené akce.



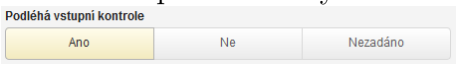
- Zamezení opětovného odeslání formulářů zablokováním tlačítka.
- Používání vhodných ikon u tlačítek.



- Skrývání nedůležitých nebo neaktuálních informací s možností zobrazit je.

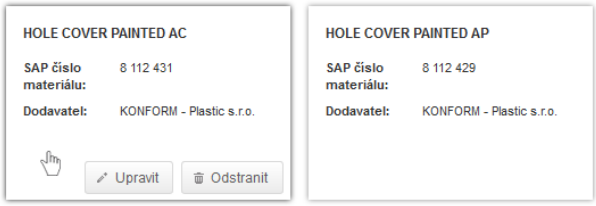


- Používání specializovaných ovládacích prvků.

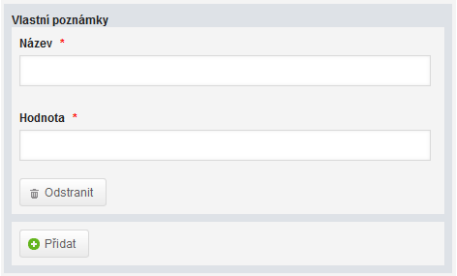


- Zobrazení ovládacích prvků v závislosti na aktuálním kontextu aplikace.

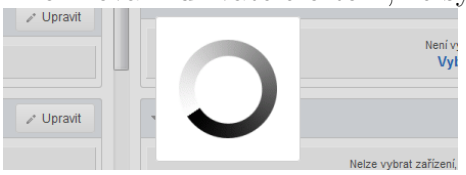
- Zobrazení skrytých ovládacích prvků při najetí kurzorem myši.



- Formuláře s variabilním množstvím zadávacích polí.

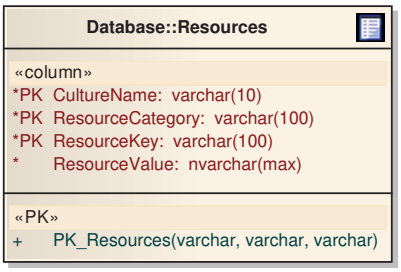


- Informování uživatele o tom, že systém provádí nějakou akci.



3.6 Lokalizační vrstva

Aplikace bude s vysokou pravděpodobností nasazena i v zahraničních pobočkách společnosti Magna E&I, a proto je nezbytné, aby nabízela možnost přepnutí do jiného jazyku, a to i za běhu. K těmto účelům slouží lokalizační



Obrázek 3.19: Schéma databázové tabulky pro uložení lokalizovaných zdrojů

vrstva, která spravuje lokalizované zdroje použité v prezentační vrstvě a ve vrstvě aplikačních služeb.

Framework ASP.NET, a tedy i ASP.NET MVC, je na potřeby lokalizace připraven a umožňuje rychlé napojení na .resx soubory obsahující definici klíčů a k nim přidružených překladů. V aplikaci mohou existovat soubory se stejným obsahem, avšak určené pro různé jazyky. Překlady se pak vyhledávají v souboru, který odpovídá zvolenému jazyku. Soubory .resx jsou založené na formátu XML a vývojové prostředí Visual Studio k nim automaticky generuje a udržuje třídy, jejichž vlastnosti odpovídají zadaným klíčům a jejich hodnotám. Jejich nevýhodou však je, že jsou primárně kompilovány spolu s ostatním kódem, a jakákoliv změna v jejich obsahu tak vyžaduje zásah programátora.

Popsané chování je však pro potřeby této aplikace nepřijatelné, neboť uživatelé musejí být schopni překlady upravovat za chodu aplikace a provedené změny musí být ihned viditelné. Důvodem je například uchovávání některých atributů přiřazovaných dílům v databázi (třída ocenění, typ materiálu, metrická jednotka a podobně) a potřeba možnosti upravovat je. Rozhodl jsem se proto opustit tradiční styl ukládání zdrojů v .resx souborech a přistoupil jsem k řešení, kdy jsou zdroje zapisovány do databáze. Musel jsem navrhnout a implementovat vlastního poskytovatele zdrojů (Resource Provider), přičemž jsem se inspiroval v [17]. Schéma tabulky, kde jsou zdroje uloženy, je zobrazeno na diagramu 3.19. Každý zdroj je jednoznačně určen jazykem, kategorií a klíčem.

Lokalizační vrstva nabízí rozhraní *ILocalizedStringProvider*, které poskytuje metody k získání překladu podle zadané kategorie a klíče, podle zadané třídy (typu) a jména její vlastnosti, a podle zadané třídy, jména její vlastnosti a jména atributu uvedeného u té vlastnosti.

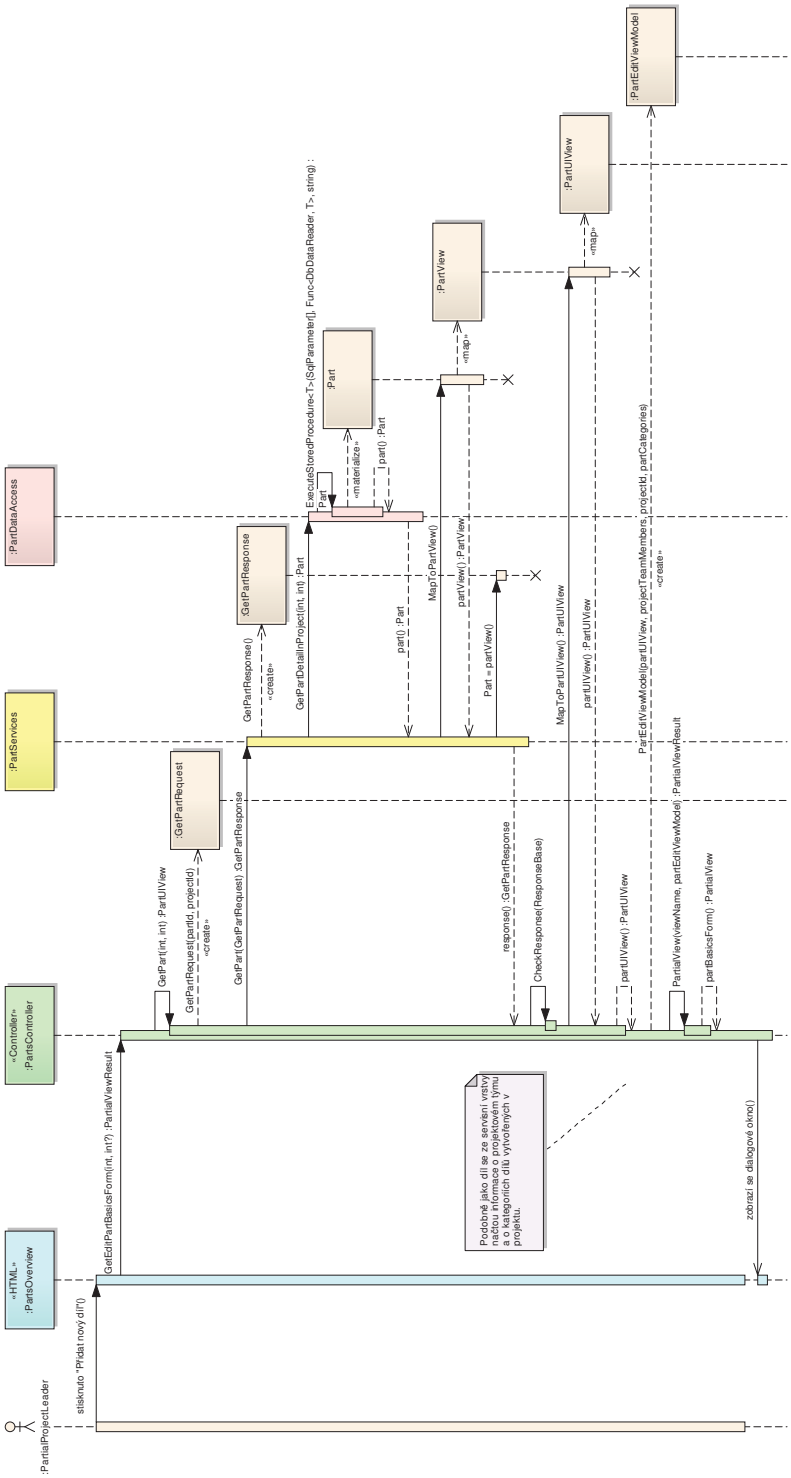
3.7 Komunikace napříč vrstvami

Příklad komunikace mezi objekty napříč vrstvami ukážu na sekvenčních diagramech 3.20 a 3.21. Sekvenční diagramy (sequence diagrams) patří spolu s diagramy spolupráce (collaboration diagram) do skupiny diagramů interakce, které popisují spolupráci skupin objektů pro dosažení určitého chování. Typicky zachycují chování jednoho případu užití, viz [2].

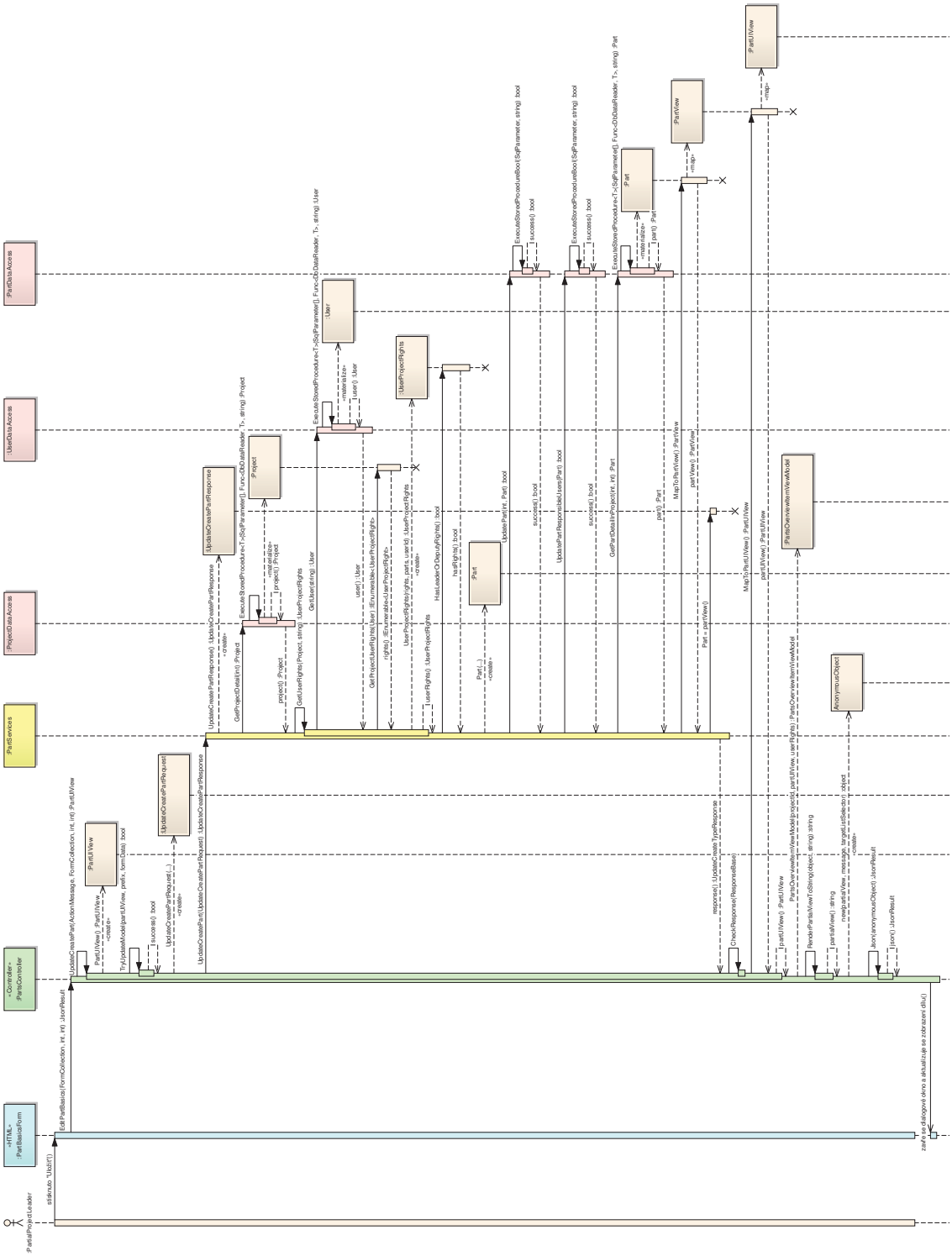
Na diagramech 3.20 a 3.21 je zachycena realizace případu užití *UC401: Edit part (Upravit díl zařazený do projektu)*. První diagram (3.20) zobrazuje proces, který proběhne při zahájení případu užití a skončí zobrazením dialogového okna s formulářem pro upravení vlastností dílu. V jeho průběhu jsou načteny informace o dílu, projektovém týmu a kategoriích dílů existujících v projektu. Poslední dva zmíněné kroky nejsou na diagramu pro zjednodušení ukázány. Proběhly by velice podobně jako získání dílu.

Na druhém diagramu (3.21) je zachycen proces probíhající po vyžádání uložení provedených změn uživatelem. Servisní třída *PartServices* zkontroluje práva uživatele, který změnu provádí a případně zajistí aktualizaci vlastností dílu a zodpovědných nákupčí. Na konci procesu je dialogové okno uzavřeno a pomocí JavaScriptu jsou provedené změny zobrazeny na stránce.

Diagramy také kvůli lepší přehlednosti nezachycují alternativní kroky prováděné v případě výskytu nějaké chyby.



Obrázek 3.20: Sekvenční diagram – otevření dialogového okna pro úpravu dílu



Obrázek 3.21: Sekvenční diagram – uložení formuláře pro úpravu dílu

Kapitola 4

Realizace

V této kapitole popisuji implementační detaily vybraných částí aplikace a význam a způsob použití některých knihoven. Uvádím pouze složitější problémy a nepříliš často používané postupy. Zmiňuji zde také některá rizika spojená s návrhem a implementací. Závěr kapitoly věnuji nasazení aplikace na server společnosti Magna E&I.

Implementace aplikace dosud nebyla dokončena a očekává se, že práce na ní budou probíhat minimálně do konce roku 2013. První verze aplikace byla zprovozněna a uživatelům předložena k testování začátkem února 2013. Od té doby byla několikrát opravena a rozšířena o nové funkce. První verze obsahovala kompletní správu projektů, dílů, termínů, dodavatelů, Checklistu a přehledu cen dílů. Poté byly na základě požadavků a priorit zaměstnanců OPN doplněny funkce jako import dílů a dodavatelů ze souboru, různé možnosti vyhledávání, filtrování a řazení, výstupy přizpůsobené pro tisk a podobně.

Během testovacího provozu aplikace bylo také zjištěno, že některé požadavky byly chybně formulovány nebo se na ně zcela zapomnělo. Požadované úpravy byly vždy upřednostněny před jakýmikoliv jinými zásahy do aplikace. Některé si vyžádaly změnu business modelu a tedy i struktury databáze. Až dlouhodobější používání aplikace v ostrém provozu odhalilo některé drobné nedostatky v návrhu uživatelského prostředí. Na základě připomínek uživatelů byly přesunuty nebo upraveny některé ovládací prvky.

4.1 Vrstva business modelu

4.1.1 Fluent Validation

Business objekty se při vytváření či úpravě validují nejen na úrovni prezentační vrstvy, ale také na úrovni vrstvy business modelu. Pro účely validace jsem využil knihovnu Fluent Validation (FluentValidation.dll dostupné z `http://fluentvalidation.codeplex.com/`). Obsahuje předdefinovanou sadu pravidel a zároveň umožňuje její rozšiřování o vlastní pravidla. Funguje na principu definování vlastní třídy, která rozšiřuje generickou třídu *AbstractValidator<T>*. Za generický parametr se dosadí typ objektu, který chceme validovat. Validační pravidla se následně zadávají do konstruktoru nové třídy. Na ukázce kódu 4.1 lze vidět definování validátoru pro třídu *Supplier* a následné vykonání validace. Formát chybových zpráv se může zdát nic neříkající, jedná se však pouze o přizpůsobení pro potřeby lokalizace. Zprávy jsou parsovány a následně je přes rozhraní *ILocalizedStringProvider* vyhledán odpovídající překlad.

```
1  public class SupplierValidator : AbstractValidator<Supplier>
2  {
3      public SupplierValidator()
4      {
5          RuleFor(s => s.SAPNumber).Matches(@"\d{8}").WithMessage("SAPNumber"
               );
6          RuleFor(s => s.Name).NotEmpty().WithMessage("Required").Length(0,
               100).WithMessage("StringLength;100");
7          RuleFor(s => s.CIN).Length(0, 10).WithMessage("StringLength;10");
8          RuleFor(s => s.VAT).Length(0, 20).WithMessage("StringLength;20");
9          RuleFor(s => s.EDI).Length(0, 50).WithMessage("StringLength;50");
10         RuleFor(s => s.DUNS).Length(0, 50).WithMessage("StringLength;50");
11         Custom(s =>
12         {
13             return s.Address.Valid()
14                 ? null
15                 : new ValidationFailure("Address", s.Address.
                       GetBrokenRulesString());
16         });
17     }
18 }
19 ...
20 Supplier supplier = new Supplier();
21 SupplierValidator validator = new SupplierValidator();
22 ValidationResult results = validator.Validate(supplier);
```

Zdrojový kód 4.1: Definice validátoru pro třídu Supplier

4.1.2 AutoMapper

V předchozí kapitole Návrh jsem uvedl, že v aplikaci existují celkem tři velice podobné modely. Každý je ale přizpůsobený svému konkrétnímu účelu.

Modely se vykytují na úrovni vrstvy business modelu, vrstvy aplikačních služeb a prezentační vrstvy. Objekty modelů je přitom mezi sebou nutné převádět, jak je naznačeno na diagramech 3.20 a 3.21. Knihovnu AutoMapper (AutoMapper.dll dostupné z `http://automapper.codeplex.com/`) jsem použil pro automatické převádění typů. Jeden převod probíhá mezi business modelem a ViewModelem a druhý mezi ViewModelem a UIViewModelem. Při prvním spuštění aplikace jsou ve statické třídě nastaveny jednotlivé převody, viz ukázka 4.2.

```
1  public class AutoMapperBootstrapper
2  {
3      public static void Bootstrap()
4      {
5          ...
6          Mapper.CreateMap<Project, ProjectView>();
7          ...
8          Mapper.CreateMap<Part, PartView>()
9              .ForMember(p => p.ResponsibleBuyers, o=> o.MapFrom(p => p.
                   ResponsibleUsers))
10             .ForMember(p => p.ForeignName, o=> o.MapFrom(p => p.
                   NameForeign));
11         ...
12     }
13 }
```

Zdrojový kód 4.2: Nastavení mapování mezi třídami pro knihovnu AutoMapper

Pro snadnější provádění již konkrétních převodů mezi objekty jsem vytvořil rozšiřující třídy, viz ukázka 4.3.

```
1  public static class ProjectMapper
2  {
3      public static ProjectView MapToProjectView(this Project project)
4      {
5          return Mapper.Map<Project, ProjectView>(project);
6      }
7
8      public static IList<ProjectView> MapToProjectViews(this IEnumerable<
           Project> projects)
9      {
10         return Mapper.Map<IEnumerable<Project>, IList<ProjectView>>(
               projects);
11     }
12 }
```

Zdrojový kód 4.3: Pomocná třída využívající knihovnu AutoMapper pro automatický převod mezi třídami

4.2 Vrstva pro přístup k datům

4.2.1 Entity Framework Extensions

Jak jsem již uvedl v předchozí kapitole, Veškeré operace v databázi provádějí uložené procedury. Aby bylo možné z databáze jednoduše získávat složité struktury business objektů bez několikanásobného dotazování, vytvořil jsem procedury vracející několik výsledků (takzvaných result setů). Pro jejich čtení jsem použil knihovnu Entity Framework Extensions (EFExtensions.dll dostupné z `http://efextensions.codeplex.com/`), která rozšiřuje populární ORM knihovnu Entity Framework.

Všechny třídy přistupující k datům rozšiřují základní abstraktní třídu *DataAccessBase*, která obsahuje šablony metod volajících procedury. Jedna z metod je uvedena na ukázce 4.4. Metoda získá aktuální kontext připojení k databázi, zavolá proceduru podle zadaného názvu a spustí vloženou anonymní funkci pro přečtení výsledků (takzvané materializování objektů).

```
1  ...
2  protected T ExecuteStoredProcedure<T>(string spName, Func<DbDataReader, T>
       materializer, params SqlParameter[] parameters)
3  {
4      DbCommand cmd = DataContextFactory.GetDataContext().CreateStoreCommand(
           spName, CommandType.StoredProcedure, parameters);
5      T result = default(T);
6      using (IDisposable connectionScope = cmd.Connection.
           CreateConnectionScope())
7      {
8          using (DbDataReader reader = cmd.ExecuteReader())
9          {
10             result = materializer(reader);
11         }
12     }
13     return result;
14 }
15 ...
```

Zdrojový kód 4.4: Ukázka metody sloužící jako šablona pro volání uložených procedur

Na ukázce kódu 4.5 lze vidět použití výše popsané metody v již konkrétní třídě. Účelem funkce `GetUser(string username)` je nalézt uživatele včetně jeho rolí podle zadaného uživatelského jména. Generická metoda `Materialize<T>()` volaná na objektu *DbDataReader* je součástí knihovny Entity Framework Extensions. Zajistí vytvoření nové instance typu, který je generickým parametrem, a naplnění jeho vlastností hodnotami vrácenými procedurou na základě shodujících se jmen. Metoda `NextResult()` provede načtení dalšího result setu.

```
1  public User GetUser(string username)
2  {
```

```
3       return ExecuteStoredProcedure<User>("User_GetUserByUsername",
4           (reader) =>
5           {
6               User user = reader.Materialize<User>().SingleOrDefault<User>();
7               reader.NextResult();
8               if (user != null)
9                   user.Roles = reader.Materialize<Role>().ToList();
10              return user;
11          },
12          Parameter("Username", username, DbType.String));
13  }
```

Zdrojový kód 4.5: Ukázka použití šablonové metody pro volání uložených procedur

4.2.2 Uložené procedury

Některé procedury vyžadují zadání velkého množství vstupních parametrů, či dokonce kolekcí dat. Tuto problematiku jsem vyřešil předáváním dat ve formátu XML. Za pomoci metod z knihovny LINQ to XML je nejprve vytvořena reprezentace objektu ve formátu XML, viz ukázka 4.6. Na ukázce 4.7 je pak vidět způsob čtení XML v jazyce T-SQL. Parametr `@User` je typu XML.

```
1  public static XDocument ConvertToUserXml(this User user)
2  {
3      return new XDocument(
4          new XElement("User",
5              new XElement("Username", user.Username),
6              new XElement("FirstName", user.FirstName),
7              new XElement("LastName", user.LastName),
8              new XElement("Phone", user.Phone),
9              new XElement("Email", user.Email),
10             new XElement("Roles", user.Roles.Select(x =>
11                 new XElement("Rolename", x.RoleType.ToString()))));
12 }
```

Zdrojový kód 4.6: Ukázka metody převádějící objekt do XML

```
1  INSERT INTO Users (Username, FirstName, LastName, Phone, Email)
2  SELECT
3    U.Item.value('(Username/text())[1]', 'varchar(50)'),
4    U.Item.value('(FirstName/text())[1]', 'nvarchar(50)'),
5    U.Item.value('(LastName/text())[1]', 'nvarchar(50)'),
6    U.Item.value('(Phone/text())[1]', 'varchar(20)'),
7    U.Item.value('(Email/text())[1]', 'varchar(50)')
8  FROM @User.nodes('/User') AS U(Item)
```

Zdrojový kód 4.7: Ukázka čtení XML v jazyce T-SQL

Zasílání emailových upozornění na blížící se termíny je také řešeno pomocí uložené procedury. Pro rozesílání programově vygenerovaných zpráv používá IT oddělení liberecké pobočky společnosti Magna E&I databázi

s tabulkou, kam jsou ukládány všechny emaily k odeslání. V co nejkratším možném čase je zajištěno jejich automatické odeslání a vymazání z tabulky. Vytvořil jsem proceduru, která je automaticky spouštěna každý den chvíli po půlnoci. Procedura vyhledá upozornění, která ten den mají být odeslána, vybere příjemce, sestaví zprávu a výsledek uloží do tabulky pro odesílání.

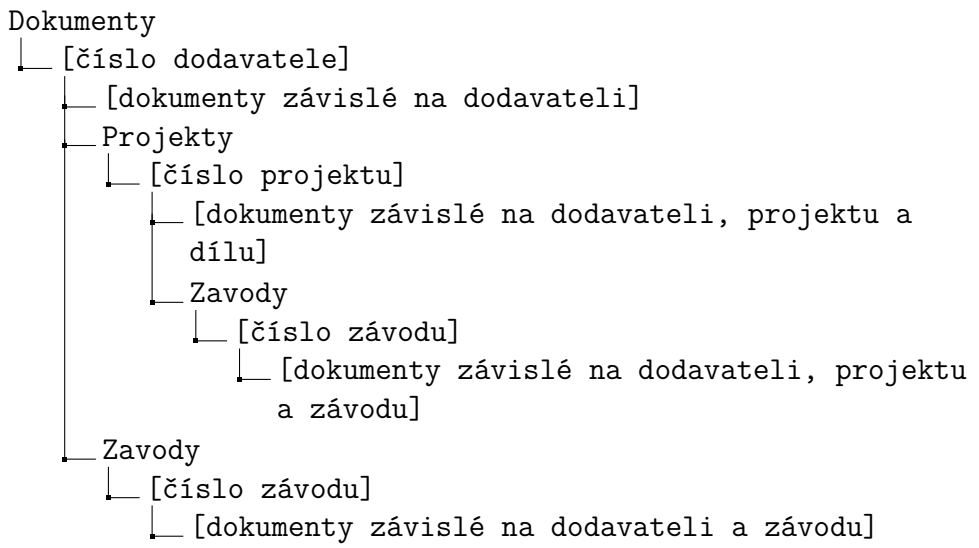
4.3 Vrstva aplikačních služeb

4.3.1 Organizace projektové dokumentace

Aplikace dle požadavků musí zajistit ukládání a organizování projektové dokumentace. V kapitole 3.3.1 jsem popsal způsob ukládání dokumentů v databázi, nyní podrobněji popíšu implementační řešení této problematiky. Mým cílem bylo vytvořit řešení co nejjednodušší na implementaci i následné použití. Využil jsem proto návrhový vzor Strategy. Vytvořil jsem strategii pro každý druh závislosti dokumentu na dalších entitách (dodavatel, projekt, závod a díl), přičemž všechny spojuje společné rozhraní a základní abstraktní třída, definující společné metody. Dílčí strategie zajišťují vytváření, ukládání a načítání dokumentů, což také zahrnuje správné pojmenování dokumentu a vytvoření odpovídající adresářové struktury. Mezi další funkčnost, kterou strategie poskytují, patří přiřazování nových závislostí k existujícím dokumentům a naopak odebírání závislostí, či vyhledání existujících dokumentů stejných typů.

Důležitou součástí je organizace dokumentů do adresářů. Vyhledávání dokumentů bude v naprosté většině případů probíhat přes uživatelské rozhraní aplikace, ale i přesto může být ojediněle nutné vyhledat dokument manuálně. Navrhnul jsem proto adresářovou strukturu, která odpovídá závislostem dokumentů. Její obecná podoba je ukázána na obrázku 4.1. Dokumenty se primárně dělí podle dodavatelů, neboť každý dokument je napojen na alespoň jednoho dodavatele. Další dělení může probíhat podle projektů a závodů. Dokumenty se nedělí podle příslušnosti k dílům, protože jeden dokument může být napojen na více dílů. Dokumenty stejných typů jsou shromážděny v adresáři, jehož název odpovídá názvu typu.

Názvy adresářů jsou přeloženy do jazyka, který je vybrán při instalaci systému a je uchován v konfiguračním souboru Web.config. Je zde rovněž uložena cesta ke kořenovému adresáři, kam jsou dokumenty nahrávány. Názvy souborů se liší pro různé druhy závislostí, viz tabulka 4.1 a s ní související vysvětlivky. Názvy souborů ani adresářů neobsahují znaky s diakritikou, všechny české znaky jsou nahrazeny za znaky bez diakritiky, stejně tak mezery jsou nahrazeny podtržítky.



Obrázek 4.1: Obecná adresářová struktura pro uložení projektové dokumentace

Tabulka 4.1: Formát názvů dokumentů pro různé závislosti

Závislost na				**Formát názvu dokumentu**
Dodavatel	**Projekt**	**Závod**	**Díl**	
×				[NDod]-[ČDod]-[NTD]-[ID]-[datum]
×		×		[NDod]-[ČDod]-[NZáv]-[ČZáv]-[NTD]-[ID]-[datum]
×	×	×		[NDod]-[ČDod]-[NPro]-[ČPro] -[NZáv]-[ČZáv]-[NTD]-[ID]-[datum]
×	×		×	[NDod]-[ČDod]-[NPro]-[ČPro] -[NTD]-[ID]-[datum]

Vysvětlivky k tabulce 4.1:

- [NDod], [NPro], [NZáv]: název dodavatele / projektu / závodu. Bere se vždy nejvýše prvních deset znaků názvu.
- [ČDod], [ČPro], [ČZáv]: SAP číslo dodavatele / číslo projektu / číslo závodu.
- [NTD]: Název typu dokumentu, který je lokalizován do zvolného jazyka.
- [ID]: Unikátní identifikační číslo, které je dokumentu přiřazeno při ukládání do databáze.
- [datum]: Datum uložení dokumentu ve formátu YYYY_MM_DD, kde YYYY je rok, MM je měsíc a DD je den.

V případě, že dojde ke změně čísla nebo názvu dodavatele, projektu nebo závodu, systém zajistí přejmenování příslušných adresářů a dokumentů na novou aktuální hodnotu.

Mimo vytváření dokumentů je nutné počítat také s jejich mazáním, aby se disková kapacita nezaplňovala nepotřebnými daty. Sami uživatelé dokumenty nemažou, pouze odstraňují jejich přiřazení k business entitám. Mazání zajišťuje systém automaticky. Odstraní-li se poslední přiřazení nějakého dokumentu, je dokument smazán. Je-li smazán poslední dokument v adresáři, je vymazán tento adresář i všechny rodičovské, které obsahují pouze jediný adresář.

Použití vzoru Strategy mi umožnilo vytvořit vždy jedinou implementaci komponent uživatelského rozhraní k nahrávání a stahování dokumentů pro všechny typy závislostí. Stejně tak pro všechny závislosti existuje jediné rozhraní servisní třídy *DocumentServices*. Součástí request objektu zasílaného metodám této třídy je mimo jiné vždy název typu dokumentu, kterého se prováděná operace týká. Na základě typu dokumentu je vyvozen typ závislosti, podle níž je pomocí reflexe vytvořena instance odpovídající strategie.

4.4 Prezentační vrstva

4.4.1 Dependency Injection

Dependency Inversion Principle je jedním z návrhových principů, který by měl být dodržen během objektově orientovaného návrhu aplikací. Jednotlivé třídy by měly být abstrahovány od konkrétních implementací a měly by záviset pouze na rozhraních nebo abstraktních třídách, což zvyšuje flexibilitu systému a podporuje zásadu volného párování (loose coupling), viz [45]. S principem úzce souvisí DI[42], jinak také označované jako IoC[43]. DI spočívá v poskytnutí závisející třídy přes konstruktor, metodu nebo vlastnost třídy, která ji používá. Třída se přitom neodkazuje na konkrétní implementaci, ale na rozhraní, což je v souladu s principem Dependency Inversion. IoC container má za úkol poskytovat programátorem určené konkrétní třídy klientskému kódu, aniž by klientský kód specifikoval, které implementace přesně vyžaduje.

Nevytvářel jsem vlastní IoC container, ale vybral jsem jednu z mnoha již hotových implementací, kterou je Ninject (Ninject.dll dostupné z `http://www.ninject.org/`). Pro začlenění knihovny jsem se řídil pokyny uvedenými v [16]. Vytvořil jsem vlastní implementaci rozhraní *IDependencyResol-*

[42]DI – Dependency Injection
[43]IoC – Inversion of Control

ver, jež ASP.NET MVC framework využívá k vytvoření jakékoliv instance třídy, kterou potřebuje. Kde to bylo možné, využil jsem předání závisející třídy v konstruktoru. V ostatních případech jsem použil předání přes vlastnost.

4.4.2 Zabezpečení

K implementaci zabezpečení aplikace včetně autorizace a autentizace jsem využil mechanismus převzatý z klasického ASP.NET, který je velice jednoduchý na použití a ke zprovoznění v podstatě vyžaduje pouze upravení obsahu konfiguračního souboru Web.config. Jak bylo řečeno v kapitole 2.1.1, systém má ověřovat identitu uživatele na základě jeho přihlášení v operačním systému Windows. Toto omezení po uživateli vyžaduje, aby v případě, že chce aplikaci využívat, byl připojen k podnikové síti.

Pro účely aplikace jsem musel vytvořit vlastního poskytovatele uživatelských rolí, aby bylo možné ověřovat práva uživatele k provádění akcí. Začal jsem implementací třídy, která rozšiřuje abstraktní třídu *RoleProvider*, ale přepisuje jen některé podstatné metody. Jedná se o metodu vracející role přihlášeného uživatele a metodu ověřující, zda je přihlášený uživatel v dané roli. V souboru Web.config jsem následně nastavil, aby systém tohoto poskytovatele používal namísto výchozí implementace. Na ukázce 4.8 je část obsahu souboru Web.config týkající se bezpečnosti.

```
1  <configuration>
2    ...
3    <system.web>
4      ...
5      <authentication mode="Windows" />
6      <roleManager defaultProvider="MagnaRoleProvider" enabled="true">
7        <providers>
8          <clear />
9          <add name="MagnaRoleProvider" type="Magna.ProjectManagement.UI.
               WebMVC.Infrastructure.Security.MagnaRoleProvider" />
10       </providers>
11     </roleManager>
12     ...
13   </system.web>
14   ...
15 </configuration>
```

Zdrojový kód 4.8: Nastavení zabezpečení v souboru Web.config

4.4.3 Cassette

Knihovnu Cassette (balík několika závislých DLL[44] dostupných z `http://getcassette.net/`) jsem použil pro usnadnění správy JavaScript a CSS

[44] DLL – Dynamic-link Library

souborů. Knihovna umí převést zmíněné typy souborů do minifikované podoby, zřetězit je a výsledek vložit do HTML stránek. Zajistí také jejich uložení do cache a verzování. Klient díky tomu načítá menší množství souborů, jejichž velikost je navíc minimální.

4.4.4 Providers

Zvláštní návrh lokalizace zdrojů uložených v databázi si vyžádal implementaci providerů, které zpracovávají atributy vlastností tříd UIViewModelu. Framework ASP.NET MVC je na lokalizaci připraven tím, že umožňuje zadání atributů ve formě zobrazené na ukázce kódu 4.9. Vlastnost *FirstName* zde má přiřazené atributy *Display* a *Required*. První z nich je využita k získání textové hodnoty pro značku label při použití pomocné metody *LabelFor* pro její vygenerování, viz ukázka 4.10. Parametr *Name* obsahuje hodnotu klíče, který definuje překlad zapsaný v souboru UserResources.resx (a tedy v související třídě *UserResources*). Validační atribut *Required* říká, že vlastnost *FirstName* musí být vyplněna. Při porušení této podmínky je chybové hlášení načteno ze souboru ValidationResources.resx podle klíče RequiredMessage. Hlášení je zobrazeno v HTML kódu, který generuje pomocná metoda *ValidationMessageFor*, viz ukázka 4.10.

```
1 public class User
2 {
3     ...
4     [Display(Name = "FirstName", ResourceType = typeof(UserResources))]
5     [Required(ErrorMessageResourceName = "RequiredMessage",
          ErrorMessageResourceType = typeof(ValidationResources))]
6     public string FirstName { get; set; }
7     ...
8 }
```

Zdrojový kód 4.9: Tradiční způsob zápisu vlastností a atributů s použitím lokalizace v ASP.NET MVC 3

Jak je vidět z ukázky 4.9, popsaný způsob vyžaduje zapsání a následné udržování vcelku velkého množství kódu. Vytvořil jsem proto providery, které automatizují získávání překladů pro názvy vlastností a pro chybová hlášení.

```
1 @model User
2 ...
3 <div>
4   @Html.LabelFor(u => u.FirstName)
5   @Html.EditorFor(u => u.FirstName)
6   @Html.ValidationMessageFor(u => u.FirstName)
7 </div>
8 ...
```

Zdrojový kód 4.10: Použití pomocných metod v HTML pro vytváření formulářů

První provider rozšiřuje třídu *DataAnnotationsModelMetadataProvider* a přepisuje její metodu *CreateMetadata*. Umožňuje vynechání atributu *[Display]*, protože sám zajistí získání přeloženého popisku vlastnosti z databáze podle jejího názvu. Druhý provider rozšiřuje třídu *DataAnnotationsModelValidatorProvider* a přepisuje její metodu *GetValidators*. Provider automaticky získává přeložená chybová hlášení pro všechny zadané atributy bez nutnosti specifikovat klíče jako ve výše uvedeném případě. Přeložená hlášení nastaví také pro potřeby validace na straně klienta. Na ukázce 4.11 je vidět konečný zápis vlastnosti s použitím vlastních providerů. Překlad popisku vlastnosti *FirstName* je v databázové tabulce *Resources* vyhledán pod kategorií *User* s klíčem *FirstName*. Pokud není nalezen, je vyhledán klíč *FirstName* v obecné kategorii *Shared*. Překlad chybového hlášení pro validační atribut *Required* je postupně vyhledáván pro následující kombinace, přičemž je použita první nalezená:

- kategorie: *User*, klíč: *Msg.FirstName.Required*,
- kategorie: *User*, klíč: *Msg.FirstName*,
- kategorie: *Shared*, klíč: *Msg.FirstName.Required*,
- kategorie: *Shared*, klíč: *Msg.FirstName*,
- kategorie: *Shared*, klíč: *Msg.Required*.

```
1 public class User
2 {
3     ...
4     [Required]
5     public string FirstName { get; set; }
6     ...
7 }
```

Zdrojový kód 4.11: Způsob zápisu vlastností a atributů s použitím vlastních providerů

4.4.5 Komponenty uživatelského rozhraní

Snažil jsem se navrhnout moderní uživatelské rozhraní, které bude ve velké míře využívat JavaScript a technologii Ajax pro usnadnění práce uživatelů. Využil jsem rozšiřitelnosti knihovny jQuery a vytvořil jsem několik pluginů představujících různé komponenty uživatelského rozhraní, které se v aplikaci vyskytují. Rozšíření jsem implementoval tak, aby je bylo možné použít nejen na více místech aplikace, ale i v budoucích projektech.

Pluginy jsou přizpůsobeny k použití v kombinaci s frameworkem ASP.NET MVC, který například vyžaduje určité formátování atributů `name` a `id` u elementů představujících formulářová pole. Snažil jsem se co nejvíce vyvarovat generování HTML kódu pomocí jazyka JavaScript na klientské straně.

Proto jsem všude, kde to bylo možné, využil schopnost frameworku vytvářet a vracet částečná view (takzvaná Partial Views). Všechny uživatelské komponenty, které jsem vytvořil, komunikují se serverem téměř výhradně prostřednictvím Ajaxu, což znamená, že obdržené odpovědi mohou být pouze v jednom zvoleném formátu (například HTML, XML, JSON a podobně). Často je však kromě vygenerovaného HTML nutné zaslat i další data, například zprávu informující uživatele o úspěchu či neúspěchu provedené akce. Zvolil jsem tedy přístup, kdy jsou všechny odpovědi zasílány ve formátu JSON, přičemž vygenerované HTML je jejich součástí. Pro tyto účely jsem musel třídy controllerů rozšířit o metodu generující částečná view do textového řetězce (string), který je následně vložen do JSON objektu odpovědi. Metodu lze vidět na ukázce 4.12.

```
1  protected string RenderPartialViewToString(string viewName, object model)
2  {
3      try
4      {
5          ViewData.Model = model;
6          using (StringWriter sw = new StringWriter())
7          {
8              ViewEngineResult viewResult = ViewEngines.Engines.
                   FindPartialView(ControllerContext, viewName);
9              ViewContext viewContext = new ViewContext(ControllerContext,
                   viewResult.View, ViewData, TempData, sw);
10             viewResult.View.Render(viewContext, sw);
11             viewResult.ViewEngine.ReleaseView(ControllerContext, viewResult
                   .View);
12             return sw.ToString().Trim();
13         }
14     }
15     catch (Exception e)
16     {
17         return e.ToString();
18     }
19 }
```

Zdrojový kód 4.12: Metoda pro vyrenderování částečného view do textového řetězce

Dále uvádím příklady komponent uživatelského rozhraní, které jsem implementoval:

Univerzální komponenty

- **Wizard:** Slouží k rozdělení dlouhého formuláře do několika kroků, mezi kterými se uživatel může pohybovat oběma směry. Příklad použití: vytváření nového projektu.
- **Editace entity v dialogovém okně:** Editace probíhá ve dvou krocích. Nejprve je do dialogového okna načteno částečné view s odpo-

vídajícím formulářem. Po odeslání formuláře je okno zavřeno a vybraná část stránky je nahrazena novým částečným view, ve kterém jsou promítnuty provedené změny. Příklad použití: úprava vlastností projektu.

- **Editace kolekce entit v dialogovém okně:** Funguje podobně jako předchozí komponenta. Částečné view zde představuje každou entitu z kolekce. Je možné přidávat nové entity a stávající upravovat a mazat. K přidání i úpravě slouží stejné částečné view s formulářem zobrazené v dialogovém okně. Příklad použití: správa dílů a termínů v projektu.
- **Výběr několika prvků ze seznamu:** Využívá se nejčastěji v místech, kde je potřeba přiřadit několik entit k jiné entitě. Přiřazované entity jsou zobrazeny v dialogovém okně a po jejich výběru a potvrzení jsou zobrazeny na cílovém místě stránky. Příklad použití: výběr členů projektového týmu.
- **Komponenta pro práci s variabilním množstvím formulářových polí:** Umožňuje přidávat (a odebírat) formulářová pole do formuláře. Přidávaná pole jsou definována v šabloně, do které je možné umístit i skupinu několika souvisejících polí. Příklad použití: vlastní atributy dílu.
- **Tabulka se „zmraženým“ záhlavím a sloupci:** Tato funkce je známá z kancelářské aplikace MS Excel. Při vertikálním „scrollování“ ve velké tabulce zůstávají vybrané řádky záhlaví tabulky na místě. Podobně při horizontálním „scrollování“ se nehýbou vybrané sloupce. Příklad použití: Checklist a přehled vývoje cen dílů.
- **Vyhledávání v tabulce:** Buňky záhlaví tabulky jsou doplněny o textové pole, kam uživatel zadává hledaný text. V tabulce se pak zobrazují pouze řádky, které mají v buňce odpovídajícího sloupce hodnotu obsahující zadaný text. Příklad použití: výběr dodavatele.

Komponenty přizpůsobené této aplikaci

- **Nahrání a stažení dokumentu:** Komponenty slouží k nahrávání dokumentů a ke stahování nebo odstraňování nahraných dokumentů. Při nahrávání uživatel vidí, k jakým entitám bude dokument přiřazen.
- **Úprava položky Checklistu:** Komponenta se využívá pro změnu hodnoty položky Checklistu. Nabízí k výběru pouze relevantní hodnoty a kontroluje, zda jsou zadány všechny požadované dokumenty. Zadané dokumenty nabízí ke stažení.

- **Vyhledávání entity:** Jedná se o filtrovací textové pole napojené na konkrétní typ entity. Psaním v poli dochází k automatickému vyhledávání mezi entitami daného typu. Entity, jejichž název (případně číslo) obsahuje zadaný text, jsou zobrazeny pod vyhledávacím polem a kliknutím mohou být vybrány. Filtrovacích polí se může na stránce vyskytovat více a mohou být na sobě závislá. To znamená, že je-li v některém poli již vybraná nějaká entita, pak vyhledávání v ostatních polích probíhá pouze mezi entitami, které se vztahují k té vybrané. Kliknutím na tlačítko „vyhledat“ pak dojde k vyhledání entit, které odpovídají zadaným filtrům.

4.5 Rizika spojená s návrhem a implementací

Výše popsaný návrh a implementace aplikace představuje určitý kompromis mezi složitostí a flexibilitou výsledného programu. Procesy OPN a s nimi související dokumentace, formuláře a tiskové výstupy nejsou ustálené a může dojít k jejich změně. Aplikace by na případné změny měla být připravena reagovat a přizpůsobit se jim nejlépe bez nutnosti zásahu programátora. Možnost libovolné customizace aplikace z uživatelského prostředí by však zároveň vyžadovala její vysokou složitost a tedy i delší dobu implementace. S návrhem aplikace a možnými změnami souvisí několik následujících rizik, kterých jsem si vědom, a která by v některých případech mohla být odstraněna s budoucím rozšiřováním aplikace.

Změna vlastností entit: OPN může požadovat sledování nových vlastností především projektů a dílů, se kterými se v návrhu nepočítalo. Tato změna by si vyžádala programátorský zásah do databáze i kódu aplikace a neexistuje žádný jednoduchý způsob, jak se této nutnosti vyvarovat.

Změna rozložení tiskových výstupů: Struktura a vzhled tiskových výstupů (Kmenová věta materiálu, Checklist a Přehled vývoje cen) jsou dány nařízením vedení OPN a mohou být kdykoliv upraveny. Důvodem může být například přidání nové sledované vlastnosti. Měly by být definovány šablony těchto dokumentů, které by bylo možné upravovat v uživatelském rozhraní aplikace nejlépe pomocí nějakého grafického editoru. Šablony by se ukládaly například do databáze a tiskové výstupy by se podle nich dynamicky sestavovaly.

Vznik nové položky Checklistu: Checklist může být kdykoliv rozšířen o nové sledované položky. Aplikace je na toto riziko již připravena, protože struktura Checklistu je kompletně ukládána v databázi a jeho zobrazení v uživatelském rozhraní dynamicky sestavováno.

Vznik nového typu dokumentu: Projektová dokumentace může být rozšířena o nový typ dokumentu, který bude třeba v aplikaci ukládat. Riziko je rovněž minimalizováno tím, že všechny typy dokumentů, včetně jejich typu závislosti a příznaku určujícím zda má být uchovávána jejich historie, jsou ukládány v databázi.

Vznik nového typu závislosti dokumentu: V současnosti jsou definovány čtyři typy závislostí dokumentů (na dodavateli, na dodavateli a závodu, na dodavateli, závodu a projektu a na dodavateli, projektu a dílu). Není však vyloučené, že si nový typ dokumentu vyžádá vznik nového typu závislosti. V takovém případě by musela být programátorem vytvořena nová strategie zajišťující všechny potřebné operace, viz kapitola 4.3.1. Vzhledem ke složitosti jednotlivých strategických tříd a jejich rozsahu není možné toto riziko nijak potlačit.

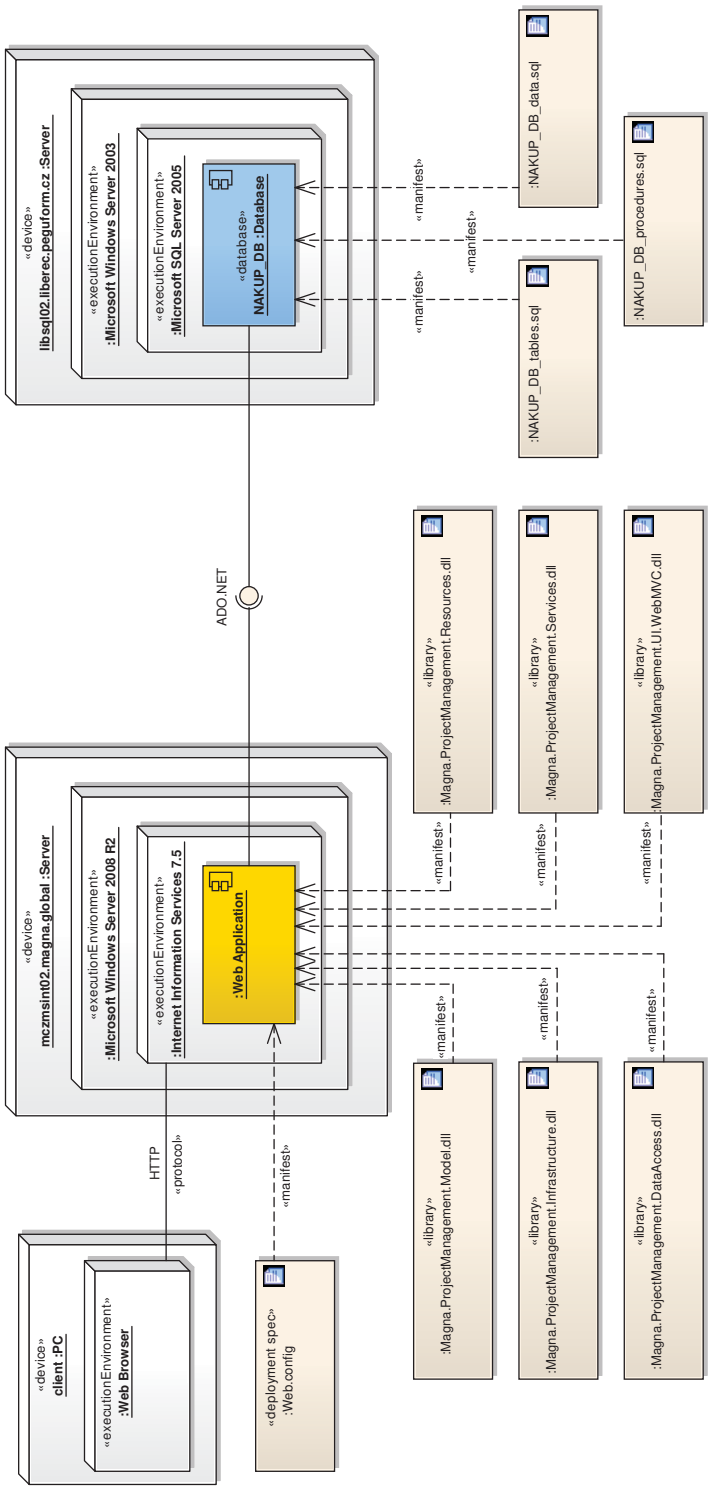
4.6 Nasazení

Aplikace je provozována na serverech společnosti Magna E&I, které jsou připojeny do podnikové sítě (intranet). Webová aplikace a databáze jsou umístěny na rozdílných serverech. Webová aplikace běží na webovém serveru Micrososft IIS 7.5, který je umístěný na serveru *mczmsint02.magna.global*. Požadované nastavení webového serveru muselo provést IT oddělení společnosti, protože mi nebyl ke konfiguračnímu rozhraní umožněn přístup. Databáze NAKUP_DB je umístěna na serveru *libsql02.liberec.peguform.cz* na databázovém stroji Microsoft SQL Server 2005. Vytvoření databáze a uživatelských účtů opět provedlo IT oddělení společnosti. Nasazení aplikace (deploy) i vytvoření databáze jsem provedl vzdáleně přes virtuální privátní síť[45].

Diagram nasazení 4.2 ukazuje fyzické rozmístění aplikace, databáze a pracovních stanic, ze kterých se k aplikaci přistupuje. Diagram nasazení se obecně skládá z uzlů, artefaktů a vztahů mezi nimi, viz [2]. Uzly představují typ výpočetního prostředku a dělí se na dva typy. Zařízení (device) zastupuje fyzické zařízení (například server nebo osobní počítač). Prostředí zpracování (execution environment) reprezentuje běhové prostředí, ve kterém je zpracováván software (například webový nebo databázový server),

[45] VPN – Virtual Private Network

a který běž na nějakém zařízení (výpočetním uzlu). Artefakty specifikují například konkrétní zdrojové soubory, spustitelné soubory, databázové tabulky, konfigurační soubory a podobně.



Obrázek 4.2: Diagram nasazení aplikace na servery společnosti Magna E&I

Kapitola 5

Testování

Cílem testování je odhalit chyby v softwarovém systému, změřit jeho kvalitu a přispět k jejímu zvýšení. Jak uvádí [6], jedná se v podstatě o proces porovnávání „toho co je“ s „tím jak by to mělo být“. Standard IEEE 610.12-1990 [20] definuje pojem testování jako: „proces řízení systému nebo komponenty za specifických podmínek, pozorování či zaznamenávání výsledků a vyhodnocování nějakých aspektů systému nebo komponenty“.

Systémy je možné testovat mnoha různými ověřenými technikami a metodami v průběhu celého životního cyklu jejich vývoje. Testy lze kategorizovat několika způsoby. Podle [6] se rozlišují čtyři testovací úrovně:

- **Jednotkové (Unit) testy:** Jednotka je nejmenší programová část testovaného softwaru. V objektově orientovaných jazycích se za jednotku obvykle považuje třída. V procedurálních jazycích to nejčastěji bývá funkce. Testování jednotek je založeno na přesné znalosti jejich implementace a funkce. Testy by měly být co nejjednodušší, aby byl programátor schopen správně ověřit jejich bezchybnou funkčnost.

- **Integrační testy:** Podstatou integračních testů je ověření bezchybné spolupráce několika jednotek. Jednotky mohou samy o sobě fungovat správně, ale podsystémy nebo systémy z nich sestavené se mohou chovat neočekávaně.

- **Systémové testy:** Systém tvoří všechny softwarové i hardwarové komponenty dodané zákazníkovi a jeho testování tak probíhá na nejvyšším stupni integrace. Spadají sem funkční testy, testy použitelnosti, zabezpečení, spolehlivosti, dostupnosti, výkonnosti a podobně.

- **Akceptační testy:** Akceptační testy provádí zákazník po obdržení hotového produktu, když ověřuje, jestli systém splňuje všechny definované funkční i nefunkční požadavky.

Dalším významným kritériem pro dělení testů je úroveň potřebné znalosti systému pro vykonání testu, viz [6]. Rozlišují se takzvané white box a black box testování. White box testování je založené na znalosti interních struktur, cest a implementačních detailů testovaného softwaru. Zároveň vyžaduje vysokou úroveň znalostí programátora provádějícího testování. Využívá se na úrovni jednotkových, integračních i systémových testů. Je důležité na něj nahlížet jako na testování cest, nikoliv kódu. Tyto cesty se mohou vykytovat uvnitř jednotek, uvnitř podsystémů, ale rovněž mezi podsystémy i systémy. Black box testování naopak vychází pouze ze znalostí požadavků a specifikací. Může být aplikováno na všech čtyřech výše zmíněných úrovních.

V rámci testování vyvíjené aplikace jsem se věnoval především funkčnímu white box a black box testování. Pro alespoň částečné otestování uživatelského rozhraní jsem využil Nielsenovu heuristiku. Mým cílem bylo co nejdříve vytvořit fungující aplikaci obsahující nejdůležitější funkce, kterou bych mohl předložit zaměstnancům OPN k otestování na několika reálných projektech a skutečných datech. Nevěnoval jsem se například zátěžovým testům, protože aplikaci bude používat pouze relativně malé množství uživatelů, kteří k ní budou přistupovat pouze prostřednictvím interní podnikové sítě.

5.1 White box testování

White box testování jsem využil zejména pro ověření správného chování business modelu. Testoval jsem metody pro validaci business objektů a jejich další důležité operace, viz ukázka 5.1 kde je testováno přiřazování zařízení k dílu. Dalšími testovanými jednotkami byly třídy zajišťující čtení souborů pro automatický import dílů a dodavatelů. Pro implementaci testů jsem využil framework NUnit (nunit.framework.dll dostupné z `http://www.nunit.org/`).

```
1  [TestFixture]
2  public class BusinessModelTests
3  {
4      ...
5      [Test]
6      public void AssignToolToPartTest()
7      {
8          Part part = new Part { Id = 1 };
9          Tool tool1 = new Tool { Id = 1 };
10         Tool tool2 = new Tool { Id = 2 };
11         part.AssignTool(tool1);
```

```
12            Assert.AreEqual(part.Tools.Count, 1);
13            Assert.AreEqual(part.Tools[0], tool1);
14            part.AssignTool(tool2);
15            Assert.AreEqual(part.Tools.Count, 2);
16            Assert.AreEqual(part.Tools[0], tool1);
17            Assert.AreEqual(part.Tools[1], tool2);
18            Assert.Throws<ModelException>(delegate { part.AssignTool(tool1); })
                  ;
19            Assert.Throws<ModelException>(delegate { part.AssignTool(null); });
20        }
21        ...
22    }
```

Zdrojový kód 5.1: Ukázka jednotkového testu – přiřazení zařízení k dílu

5.2 Black box testování

V rámci black box testování jsem testoval funkčnost jednotlivých vrstev a především jejich vzájemnou integraci. Využil jsem nástroj pro automatizaci testování webových aplikací Selenium (`http://www.seleniumhq.org/`). Za pomoci Selenium IDE[46], které je dodáváno v podobě rozšíření pro internetový prohlížeč Firefox, je možné zaznamenat manuálně prováděné akce ve webové aplikaci a následně je již automaticky zopakovat či exportovat do kódu v libovolném programovacím jazyce. Využil jsem možnosti exportu v jazyce C# do tříd používajících framework NUnit.

Vytvořil jsem několik testů, které reprezentují vybrané případy užití. Jejich vykonáním dojde k otestování funkčnosti uživatelského rozhraní včetně validování modelu na úrovni prezentační vrstvy, otestování vrstvy aplikačních služeb i business modelu, jeho operací a validování, a otestování vrstvy pro přístup k datům spolu s uloženými procedurami. Testy je nutné vykonávat v pevně daném pořadí, aby pro každý test byla k dispozici data vytvořená v průběhu některého z předcházejících testů. Využil jsem vlastnosti frameworku NUnit, který testy spouští v abecedním pořadí. Otestovány byly následující případy užití:

- Projekty: vytvoření projektu, úprava vlastností projektu, úprava projektového týmu.
- Termíny: vytvoření termínu, úprava termínu, smazání termínu, změna stavu termínu, vytvoření upozornění, úprava upozornění, smazání upozornění.
- Dodavatelé: vytvoření dodavatele, úprava vlastností dodavatele.

[46]IDE – Integrated Development Environment

- Díly: úprava vlastností dílu, úprava ceny dílu, přiřazení dodavatele, vytvoření nástroje, úprava nástroje, smazání nástroje, vytvoření dílu, odebrání dílu z projektu, přiřazení existujícího dílu k projektu.

- Dokumentace: nahrání dokumentu závislého na dodavateli, nahrání dokumentu závislého na jednom i více dílech, odebrání dokumentu, změna názvu dodavatele či projektu a s tím související přejmenování dokumentových adresářů a souborů.

- Checklist: Úprava položek Checklistu.

Další integrační testy, které jsem vytvořil, se soustředí především na otestování vrstvy aplikačních služeb, business modelu a vrstvy pro přístup k datům. Spočívají ve volání metod servisních tříd a ověřování jejich výsledků. Jejich součástí jsou také testy zabezpečení aplikace z pohledu rolí a oprávnění uživatelů.

5.3 Testování uživatelského rozhraní

Klasické testy použitelnosti jsem neprováděl. Aplikace je určena konkrétní úzké skupině uživatelů, a proto si myslím, že jejich připomínky z dlouhodobějšího používání jsou přínosnější, než test s anonymním uživatelem, který navíc není seznámený s procesy OPN. Uživatelské prostředí proto bylo po dobu testovacího i následného ostrého provozu upravováno na základě požadavků uživatelů.

Provedl jsem alespoň test bez uživatele založený na Nielsenově heuristické analýze. Ta slouží k odhalení chyb a nedostatků v návrhu a použitelnosti uživatelského rozhraní. Byla vyvinuta Jakobem Nielsenem a Rolfem Molichem a poprvé publikována v [47] již v roce 1994. Přestože se jedná o téměř dvacet let starou metodiku, je možné ji v pouze mírně pozměněné podobě stále stejně dobře uplatnit pro otestování současných aplikací. Heuristická analýza se skládá z deseti pravidel, které by měly být dodrženy při návrhu uživatelského rozhraní. Jejich přehled lze nalézt na [48]. Mohou ji provádět sami vývojáři či jiní experti procházením aplikace a kontrolou naplnění všech bodů. Dále uvádím seznam pravidel, jejich význam a popis, jak je uživatelské rozhraní vyvíjené aplikace splňuje či porušuje.

Viditelnost stavu systému
Systém by měl uživatele informovat o tom, co právě vykonává prostřednictvím vhodné zpětné vazby a v rozumném čase.

Aplikace uživatele informuje o výsledcích provedených akcí dočasně zobrazenými zprávami v pravém horním rohu uživatelského prostředí. Zprávy jsou barevně rozlišeny podle toho, zda byla akce vykonána úspěšně či nikoliv. Pokud aplikace provádí déle trvající Ajaxový dotaz, uživatelské prostředí je ztmaveno a je zobrazen tradiční animovaný obrázek představující probíhající práci systému.

Shoda systému s realitou
V systému by měly být použity termíny, fráze a koncepty, které uživatel důvěrně zná. Informace by měly být předkládány v podobě známé z reálného světa.

V aplikaci jsou použity termíny pro pojmenování entit a jejich vlastností tak, jak si je definovali sami uživatelé. S těmito termíny se uživatelé denně setkávali a používali je již před zavedením aplikace. Tlačítka jsou opatřena vhodnými ikonami, které nejlépe vystihují související akci.

Minimální zodpovědnost
Uživatel by měl mít možnost vrátit akce, které vykonal nechtěně. Měla by existovat možnost „nouzového východu“ z částí systému, kam se uživatel dostal nechtěně. Uživatel by měl mít pocit, že má systém pod kontrolou.

Tento krok bývá v aplikacích nejčastěji splněn zahrnutím tlačítka pro vykonání kroku zpět a vpřed, takzvané „undo“ a „redo“. V této aplikaci jsem však podobnou funkci neimplementoval. Veškeré změny entit je však možné opakováním provedené akce vrátit do původního stavu. Před odstraněním entity nebo jejího přiřazení k jiné entitě je uživatel požádán o potvrzení.

Konzistence a standardy
Používané výrazy, ikony a ovládací prvky uživatelského rozhraní by měly mít stejný význam napříč celým systémem. Měly by být dodrženy zaběhnuté standardy pro danou platformu.

V aplikaci jsou všude použity podobné ovládací prvky. Veškeré úpravy entit probíhají v dialogových oknech s podobnou strukturou – okno má záhlaví s názvem, obsah s formulářem a zápatí s tlačítky pro uložení a zavření. Stránky s přehledy entit a detaily entit mají vždy stejné rozložení.

Prevence chyb
Systém by měl být navržen tak, aby uživatelé dělali co nejméně chyb. Uživatelské rozhraní by mělo obsahovat prvky, které zamezí vzniku chyb.

Povinná formulářová pole jsou označená hvězdičkou. Měl by však být doplněn popis požadovaného tvaru či rozsahu zadané hodnoty. Prozatím je v aplikaci implementována alespoň validace na straně klienta, která ihned

zobrazí hlášení v případě chybně zadané hodnoty. Po odeslání formuláře dojde buď k zablokování potvrzovacího tlačítka, nebo k okamžitému zavření dialogového okna, což zamezí případnému dvojímu odeslání. Nevratné akce, jako je smazání entity, vyžadují potvrzení od uživatele.

Rozpoznání místo rozvzpomínání
Měly by být minimalizovány nároky na paměť uživatele. Informace o stavu systému, o tom, kde se uživatel právě nachází a další důležité informace by měly být vždy viditelné nebo alespoň snadno dohledatelné.

Ve speciálním menu aplikace je vždy zřetelně vyznačeno, na jaké stránce či v detailu jaké entity se uživatel právě nachází. Dialogová okna jsou opatřena vhodným titulkem. Při nahrávání dokumentu, či změně položky Checklistu je v dialogovém okně uvedeno, k jakým entitám se prováděná akce vztahuje.

Flexibilita a efektivita použití
Systém by měl být stejně dobře použitelný pro začátečníka i zkušeného uživatele. Měly by být přítomny ovládací prvky urychlující navigaci a vykonávání častých akcí.

Formuláře vyskytující se v aplikaci jsou relativně jednoduché, ale postrádají implementaci některých klávesových zkratek, které by zkušeným uživatelům výrazně ulehčily a urychlily práci. Měla by být doplněna možnost ukládat formuláře v dialogových oknech pomocí klávesy Enter. Okna je již nyní možné zavírat pomocí klávesy Esc. Při vyhledávání se během psaní zobrazují pod vyhledávacím textovým polem nalezené výsledky se zvýrazněným zadaným textem. Mezi těmito výsledky se uživatel může pohybovat pomocí kláves šipek a vybraný výsledek potvrdit klávesou Enter.

Estetický a minimalistický design
Nadbytečné či zřídkakdy potřebné informace by měly být z uživatelského rozhraní odstraněny, aby neomezovaly ty důležité.

V uživatelském prostředí se vykytuje minimální množství textu a pouze potřebná a relevantní ovládací tlačítka. Implementovány byly pouze užitečné a požadované funkce, které uživatelé budou využívat.

Smysluplné chybové hlášení
Chybová hlášení by měla být zobrazena v podobě, které porozumí běžný uživatel. Měla by obsahovat popis příčiny chyby a návod, jak ji odstranit.

Validační chybová hlášení zobrazená při zadání neplatné hodnoty do formulářového pole jsou vždy zobrazena v blízkosti toho pole a uživatele jasně informují, proč je hodnota neplatná. Hlášení zobrazená v případě

chyby na straně serveru by měla být doplněna o příčinu vzniklé chyby, aby uživatel mohl zjednat nápravu. Nyní ho pouze informují, že prováděná akce se nezdařila. V případě, že dojde k chybě při importu dílů nebo dodavatelů ze souboru, je zobrazeno, o kterou položku se jedná a na jakém řádku se vyskytuje.

Help a dokumentace
Systém by měl obsahovat přehlednou a snadno dostupnou dokumentaci a nápovědu.

K aplikaci byla vytvořena uživatelská příručka (viz příloha H), která uživatele seznamuje s prostředím aplikace a popisuje jak postupovat v případě zakládání projektů, vytváření dílů a podobně. Obsah tohoto manuálu by však měl být vložen i do aplikace a být dostupný na všech relevantních stránkách.

5.4 Akceptační testy

Aplikace byla budoucím cílovým uživatelům (zaměstnancům OPN) předložena k prvnímu testování v únoru 2013. Testování probíhalo s reálnými daty – v systému byly vytvořeny uživatelské účty a založeny dva projekty, na kterých OPN v té době začínalo pracovat. Postupně byly doplněny údaje o dodavatelích a k nim přiřazeny existující dokumenty. Zároveň byly v systému vytvořeny další nové i již rozpracované projekty. Během testování i následného ostrého provozu aplikace byly od uživatelů shromažďovány a zpracovávány připomínky a nové požadavky. Nebyly nalezeny žádné zásadní nedostatky či chyby, které by znemožnily používání aplikace. Více o názorech a hodnocení uživatelů uvádím v následující kapitole 6. Aplikace v tuto chvíli (duben 2013) není zcela dokončena a pokračuji v jejím vývoji. Některé počáteční funkční požadavky dosud nebyly implementovány. Důvodem jsou zejména nově vzniklé požadavky, které dostaly vyšší prioritu a byly tedy zpracovány přednostně. Testování aplikace a jejích nových funkcí tak bude nadále pokračovat.

Kapitola 6

Zhodnocení aplikace

Závěrečnou kapitolu věnuji zhodnocení přínosu aplikace pro OPN společnosti Magna E&I. Popisuji zde také další postup vývoje aplikace.

Jak jsem již uvedl v předchozí kapitole, aplikace byla uživatelům zpřístupněna k prvnímu testování v únoru 2013. Po několika týdnech testovacího provozu ji uživatelé postupně začali plně využívat a plnit novými i starými daty. V tuto chvíli (duben 2013) v aplikaci existuje 20 uživatelských účtů s rolí *Nákupčí*, jejichž vlastníci do ní mohou přistupovat. Aplikaci využilo 18 z nich alespoň jednou a 15 uživatelů ji využívá pravidelně. V aplikaci bylo dosud založeno 20 projektů a k nim přiřazeno téměř 800 unikátních dílů. Rovněž bylo do databáze vloženo více než 360 záznamů o dodavatelích. Pro naplnění systému projektovou dokumentací vztahující se především k dodavatelům a závodům byla najata brigádnice. Jejím úkolem bylo shromáždit dokumenty v elektronické podobě umístěné na různých serverech a osobních počítačích OPN a vložit je do aplikace přes uživatelské rozhraní. Celkem bylo do systému dosud nahráno více než 2 800 dokumentů.

Abych získal zpětnou vazbu od uživatelů a zjistil, jak jsou s dosavadní podobou a funkčností aplikace spokojeni, případně jaké mají výhrady, požádal jsem je o vyplnění dotazníku uvedeného v příloze I. Dotazník odevzdalo 13 oslovených zaměstnanců OPN. Výsledky dotazníku a názory uživatelů lze označit za pozitivní. Sumarizaci výsledků lze vidět v tabulce 6.1, kde je uvedeno všech sedm otázek, výčet možných odpovědí, ze kterých mohli dotazovaní vybírat, a počet zaškrtnutých odpovědí. Téměř všichni uživatelé, kteří dotazník vyplnili, aplikaci již využívají, a to buď denně, nebo alespoň několikrát týdně. Většina uživatelů věří, že jim aplikace ulehčuje práci a považuje ji za přínosnou. S grafickou stránkou uživatelského rozhraní jsou spokojeni a většina z nich se v něm dokáže rychle orientovat. Aplikaci čím dál více využívají jako poskytovatele informací o dodavatelích, projektech a

dílech namísto dosavadních prostředků, kterými jsou papírové dokumenty a elektronické dokumenty na projektovém serveru. Jako největší nedostatek vnímají fakt, že v aplikaci dosud nejsou uloženy veškeré dokumenty, projekty a díly, což jim prozatím brání v jejím větším užívání. Tento nedostatek by však měl být do několika týdnů odstraněn.

V kapitole 4.5 jsem uvedl několik rizik spojených s návrhem a implementací aplikace. Za nejčastěji se vyskytující problém lze označit změnu vlastností entit. Zaměstnanci OPN používáním aplikace postupně odhalují chybějící vlastnosti, na které se při specifikaci požadavků zapomnělo, a požadují jejich doplnění. Očekávám však, že brzy budou do systému promítnuty všechny zbývající vlastnosti a tento problém tak bude vyřešen.

6.1 Budoucí práce

Aplikace dosud není zcela dokončena a je neustále rozšiřována o nové funkce. Dosud nebyly implementovány některé původně požadované prvky systému, ale byly naopak zahrnuty nově vzniklé a upřednostněné požadavky na funkčnost. Uživatelské rozhraní průběžně doplňuji o nové prvky usnadňující a urychlující práci s aplikací – jedná se například o různé způsoby řazení a filtrování zobrazených entit. Do aplikace by v nejbližší době mělo být zahrnuto i několik dalších větších rozšíření:

Administrační rozhraní: Aplikace musí být rozšířena o administrační sekci, odkud budou spravovány některé atributy dílů, mazány projekty a díly, spravovány lokalizované zdroje a podobně, viz kapitola 2.3.7. Oprávnění pro přístup do sekce budou přidělena vybranému zaměstnanci OPN.

Automatizace Checklistu: Aplikace by měla automaticky zajistit změnu hodnot položek Checklistu při nahrání příslušných dokumentů, které s položkami souvisí. Tato funkce zaměstnancům uspoří mnoho času, který nyní musejí věnovat manuální správě Checklistu.

Úprava více dílů najednou: Bude doplněna funkčnost umožňující úpravu vlastností a cen více dílů najednou. V projektech se často vyskytují párové díly, které mají mnoho společných vlastností a uživatelé nyní musejí spravovat každý zvlášť.

Exporty entit do souborů: Aplikace musí poskytovat možnost exportovat přehledy vybraných entit (zejména projektů, dílů a dodavatelů) do souborů formátu PDF a XLSX (pracovní sešit aplikace Microsoft

Tabulka 6.1: Vyhodnocení dotazníku

1) Používáte aplikaci?					
Ano	Ne, ale brzy začnu	Ne a ani nehodlám	Ne a dosud jsem se nerozhodl/a jestli začnu		
12	1				
2) Jak často aplikaci používáte?					
Neustále	Denně	Několikrát týdně	Jednou týdně	Jednou za měsíc	
	7	6			
3) Ulehčuje Vám aplikace práci?					
Určitě ano	Spíše ano	Nevím, zatím nedokážu posoudit	Spíše ne	Určitě ne	Určitě ne, práce s ní je zdlouhavá
10	2	1			
4) Využíváte aplikaci jako poskytovatele informací namísto dosavadních prostředků?					
Ano, neustále	Ano, občas	Ne, nezkoušel/a jsem to	Ne, nemůžu tam nic najít	Ne, nejsou tam informace, které potřebuji	
4	8	1			
5) Jak hodnotíte uživatelské prostředí aplikace?					
Líbí se mi a je přehledné	Líbí se mi, ale je nepřehledné	Nelíbí se mi, ale je praktické	Nelíbí se mi a je nepřehledné	Je strašné, nedá se s ním pracovat	
13					
6) Jak se v aplikaci orientujete?					
Rychle a bez zaváhání	Pomaleji, občas něco hledám	Pomalu, pořád něco hledám	Pomalu, musím hodně klikat, než se někam dostanu	Pomalu, funkce a odkazy jsou nesmyslně umístěné	
8	5				
7) Považujete novou aplikaci za přínosnou?					
Určitě ano	Spíše ano, ale představoval/a jsem si něco jiného	Zatím nevím, uvidí se časem	Ne, je zbytečná	Ne, nepracuje se mi s ní dobře	
12		1			

Excel). Uživatel by přitom měl mít možnost zvolit si exportované vlastnosti entit a jejich pořadí, ve kterém budou v souboru uvedeny.

Šablony tiskových výstupů: Měla by existovat možnost upravovat podobu tiskových výstupů (zejména Kmenové věty materiálu) přímo v uživatelském rozhraní aplikace. Šablony by definovaly, které vlastnosti budou zobrazeny a jak budou na stránce rozloženy.

Audit: Aplikace by měla uchovávat záznamy o veškerých akcích uživatelů, aby bylo možné zpětně dohledat, kdo které změny provedl. Tyto záznamy by zároveň mělo být možné využít k sestavení historické podoby entit.

Závěr

Cílem práce bylo navrhnout, implementovat a nasadit informační systém pro podporu pracovních činností zaměstnanců oddělení nákupu společnosti Magna Exteriors & Interiors s.r.o. Snahou bylo uvést systém do provozu co nejdříve, aby bylo možné pozorovat jeho přijetí uživateli a zhodnotit jeho přínosy.

V rámci práce jsem se nejprve důkladně seznámil s procesy probíhajícími ve zmíněné společnosti a především v oddělení nákupu. Snažil jsem se co nejlépe pochopit pracovní činnosti zaměstnanců a porozumět obsahu a významu dokumentů, které vytvářejí. Zaměřil jsem se také na analýzu stávajících informačních prostředků, které se v oddělení používají. Důležité rovněž bylo odhalit všechny pojmy, business entity a jejich vlastnosti, které by následně měly být zakomponovány do aplikace. Potřebné znalosti jsem získal při konzultacích s vybranými zaměstnanci oddělení.

Na základě nabytých znalostí a dalších konzultací jsem shromáždil požadavky na budoucí systém, sestavil model případů užití a vytvořil konceptuální model, který jsem dále upřesnil ve fázi návrhu. Navrhl jsem a následně implementoval webovou aplikaci, jejíž logika je rozdělena do samostatných vrstev s přesně definovanou zodpovědností. Aplikace byla realizována v jazyce C# za použití ASP.NET MVC frameworku a několika dalších knihoven. Jako datové úložiště byl vybrán Microsoft SQL Server 2005. Aplikaci jsem otestoval a v únoru 2013 nasadil na server společnosti, aby mohla být testována přímo uživateli, a s reálnými daty. Od té doby byla aplikace mnohokrát rozšířena o nové funkce.

K aplikaci v současné době přistupuje 18 uživatelů. Ze sledování jejich aktivity je zřejmé, že ji používají pravidelně, několikrát denně. Abych získal lepší představu o názorech uživatelů a jejich zkušenostech s aplikací, požádal jsem je o vyplnění dotazníku. Výsledky lze označit za uspokojivé, neboť

zaměstnanci aplikaci využívají a považují ji za přínosnou. Předpokládají, že její význam v budoucnu vzroste, až bude naplněna všemi potřebnými daty.

Cíle práce tak mohu označit za dosažené a zadání za splněné, a to i přesto, že aplikace dosud není zcela dokončená a pokračuji v jejím vývoji. Zpracoval jsem všechny počáteční požadavky s výjimkou požadavků na administrační sekci aplikace, která však nebyla prioritní. Používáním aplikace uživatelé dosud odhalují chybějící funkce, které by dále usnadnily a zefektivnily jejich práci, a požadují jejich doplnění.

Technologie zvolené pro implementaci aplikace se ukázaly být vhodné a v průběhu práce jsem nenarazil na žádné vážnější problémy. Framework ASP.NET MVC poskytuje vývojáři prakticky neomezenou svobodu a umožňuje tak naplnit téměř všechny požadavky uživatelů na podobu a prvky uživatelského rozhraní. Jako jediný nedostatek vnímám použití uložených procedur pro komunikaci s databází, což byl však požadavek IT oddělení společnosti. Jejich začlenění vyžaduje napsání a udržování mnohem většího množství kódu než v případě využití nějakého ORM frameworku.

V průběhu celé práce jsem se seznámil s novými frameworky, technologiemi a metodami. Poprvé jsem měl také možnost vytvářet rozsáhlou aplikaci od jejích základů až po zprovoznění a údržbu pro skutečného zákazníka. Měl jsem možnost využít, ověřit si a především rozšířit znalosti získané během několikaletého školního i osobního studia. Musel jsem rovněž čelit problémům, se kterými se v budoucnu v této profesi budu nejspíše pravidelně setkávat a budu si alespoň dobře pamatovat, že práci nikdy nelze označit za hotovou a požadavky zákazníka za konečné.

Literatura

[1] ALLOCATION NETWORK: *Supplier Relationship Management with ASTRAS*. [online], 2013, [cit. 2013-03-14]. Dostupné z: `http://erfx-support.com/en/srm.html`

[2] ARLOW, J.; NEUSTADT, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2007, ISBN 978-80-251-1503-9.

[3] BIBEAULT, B.; Yehuda, K.: *jQuery in Action*. Stamford: Manning, druhé vydání, 2010, ISBN 978-1-935182-32-0.

[4] BUŠ, L.: *MI-DPO: Přednáška 8 – Observer, MVC*. 2011.

[5] COLES, M.: *Pro T-SQL 2008 Programmer's Guide*. New York: Apress, 2008, ISBN 978-1-4302-1002-3.

[6] COPELAND, L.: *A Practitioner's Guide to Software Test Design*. Boston: Artech House Publishers, 2004, ISBN 1-58053-791-X.

[7] CQS: *ISO/TS 16949:2009 - Automobilový průmysl*. [online], 2010, [cit. 2013-03-08]. Dostupné z: `http://www.cqs.cz/Normy/ISO-TS-169492009-Automobilovy-prumysl.html`

[8] CQS: *ČSN EN ISO 14001:2005 - Environmentální management*. [online], 2013, [cit. 2013-03-08]. Dostupné z: `http://www.cqs.cz/Normy/CSN-EN-ISO-140012005-Environmentalni-management.html`

[9] CQS: *ČSN OHSAS 18001:2008 - Management bezpečnosti a ochrany zdraví při práci*. [online], 2013, [cit. 2013-03-08]. Dostupné z: `http://www.cqs.cz/Normy/CSN-OHSAS-180012008-Management-bezpecnosti-a-ochrany-zdravi-pri-praci.html`

[10] ECMA INTERNATIONAL: *Standard ECMA-334 – C# Language Specification*. červen 2006. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

[11] FADEYEV, Dmitry: *10 Useful Web Application Interface Techniques*. [online], 2009, [cit. 2013-04-23]. Dostupné z: http://uxdesign.smashingmagazine.com/2009/01/12/10-useful-web-application-interface-techniques/

[12] FADEYEV, Dmitry: *12 Useful Techniques For Good User Interface Design*. [online], 2009, [cit. 2013-04-23]. Dostupné z: http://uxdesign.smashingmagazine.com/2009/01/19/12-useful-techniques-for-good-user-interface-design-in-web-applications/

[13] FADEYEV, Dmitry: *7 Interface Design Techniques to Simplify and De-clutter Your Interfaces*. [online], 2009, [cit. 2013-04-23]. Dostupné z: http://www.webdesignerdepot.com/2009/02/7-interface-design-techniques-to-simplify-and-de-clutter-your-interfaces/

[14] FOWLER, M.: *Domain Specific Languages*. Boston: Addison-Wesley Professional, 2010, ISBN 978-0-13-210754-9.

[15] FOWLER, Martin: *FluentInterface*. [online], 2005, [cit. 2013-04-18]. Dostupné z: http://martinfowler.com/bliki/FluentInterface.html

[16] FREEMAN, A.; SANDERSON, S.: *Pro ASP.NET MVC 3 Framework*. New York: Apress, třetí vydání, 2011, ISBN 978-1-4302-3404-3.

[17] GAUFFIN, Jonas: *Localization in ASP.NET MVC with Griffin.MvcContrib*. [online], 2012, [cit. 2013-04-25]. Dostupné z: http://www.codeproject.com/Articles/352583/Localization-in-ASP-NET-MVC-with-Griffin-MvcContri

[18] HANSSON, David Heinemeier: *Ruby on Rails*. [online], 2013, [cit. 2013-04-18]. Dostupné z: http://rubyonrails.org/

[19] HARTL, M.: *Ruby on Rails Tutorial*. Boston: Addison-Wesley, druhé vydání, 2013, ISBN 978-0-321-83205-4.

[20] IEEE STANDARDS BOARD: *IEEE Standard Glossary of Software Engineering Terminology*. 1990.

[21] IMDSYSTEM.cz: *Co je IMDS?* [online], 2009, [cit. 2013-04-18]. Dostupné z: `http://www.imdsystem.cz/imds.php`

[22] ISO.CZ: *ISO 14001:2004*. [online], 2013, [cit. 2013-03-08]. Dostupné z: `http://www.iso.cz/?page_id=40`

[23] ISO.CZ: *ISO 9001*. [online], 2013, [cit. 2013-03-08]. Dostupné z: `http://www.iso.cz/?page_id=38`

[24] Itzik, B.-G.: *Microsoft SQL Server 2012 T-SQL Fundamentals*. Sebastopol: O'Reilly Media, 2012, ISBN 978-0-735-65814-1.

[25] JAWA MANAGEMENT SOFTWARE GMBH: *inforum Internet Information Platform*. [online], 2013, [cit. 2013-03-14]. Dostupné z: `http://www.inforum.at/`

[26] JAWA MANAGEMENT SOFTWARE GMBH: *Magna Steyr Fahrzeugtechnik AG & Co KG*. [online], 2013, [cit. 2013-03-14]. Dostupné z: `http://www.inforum.at/en/index.php/Projects/c-10050-Automobile%3A+MAGNA.html`

[27] LARDON, Gino: *Navigation design for complex web applications*. [online], 2012, [cit. 2013-04-22]. Dostupné z: `http://www.ianua.be/blog/navigation-design-for-complex-web-applications`

[28] LARMAN, C.: *Applying UML and Patterns*. Upper Saddle River: Prentice Hall, druhé vydání, 2002, ISBN 0-13-092569-1.

[29] MACDONALD, M.: *Pro Silverlight 5 in C#*. New York: Apress, čtvrté vydání, 2012, ISBN 978-1-4302-3479-1.

[30] MACDONALD, M.; FREEMAN, A.; SZPUSZTA, M.: *Pro ASP.NET 4 in C# 2010*. New York: Apress, 2010, ISBN 978-1-4302-2530-0.

[31] MAGNA EXTERIORS & INTERIORS BOHEMIA S.R.O.: *Pracovní předpis P-30-05-01 C – Řízení projektu*. 2009.

[32] MAGNA EXTERIORS & INTERIORS BOHEMIA S.R.O.: *Pracovní předpis P-40-35-01 I – Nakupování – Specifikace a poptávky, výběr dodavatele*. 2009.

[33] MAGNA EXTERIORS & INTERIORS BOHEMIA S.R.O.: *Pracovní předpis P-40-40-01 E – Proces nakupování*. 2009.

[34] MAGNA EXTERIORS & INTERIORS BOHEMIA S.R.O.: *Magna Exteriors & Interiors Bohemia s.r.o.* [online], 2013, [cit. 2013-03-07]. Dostupné z: `http://www.magnabohemia.cz`

[35] MAGNA EXTERIORS & INTERIORS BOHEMIA S.R.O.: *Nákup.* [online], 2013, [cit. 2013-03-12]. Dostupné z: `http://www.magnabohemia.cz/nakup`

[36] MAGNA INTERNATIONAL INC.: *Magna International Inc.* [online], 2013, [cit. 2013-03-07]. Dostupné z: `http://www.magna.com/`

[37] MICROSOFT: *Getting Started with the .NET Framework.* [online], 2013, [cit. 2013-04-14]. Dostupné z: `http://msdn.microsoft.com/library/hh425099.aspx`

[38] MICROSOFT: *Microsoft Project 2010.* [online], 2013, [cit. 2013-03-14]. Dostupné z: `http://www.microsoft.com/cze/office2010/produkty/project.aspx`

[39] MICROSOFT: *.NET Framework 4.* [online], 2013, [cit. 2013-04-12]. Dostupné z: `http://msdn.microsoft.com/en-us/library/w0x726c2(v=vs.100).aspx`

[40] MICROSOFT: *.NET Framework Conceptual Overview.* [online], 2013, [cit. 2013-04-14]. Dostupné z: `http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.100).aspx`

[41] MICROSOFT: *.NET Framework Versions and Dependencies.* [online], 2013, [cit. 2013-04-12]. Dostupné z: `http://msdn.microsoft.com/en-us/library/bb822049.aspx`

[42] MICROSOFT: *RazorEngine.* [online], 2013, [cit. 2013-04-21]. Dostupné z: `http://razorengine.codeplex.com/`

[43] MICROSOFT: *SQL Injection.* [online], 2013, [cit. 2013-04-20]. Dostupné z: `http://msdn.microsoft.com/en-us/library/ms161953(v=sql.105).aspx`

[44] MICROSOFT: *Walkthrough: Organizing an ASP.NET MVC Application using Areas.* [online], 2013, [cit. 2013-04-21]. Dostupné z: `http://msdn.microsoft.com/en-us/library/ee671793(v=vs.100).aspx`

[45] MILLETT, S.: *Professional ASP.NET Design Patterns.* Indianapolis: Wilew Publishing Inc., 2010, ISBN 978-0-470-29278-5.

[46] MOSSÉ, Francis G.: *Modeling Roles*. [online], 2008, [cit. 2013-04-16]. Dostupné z: `http://objectdiscovery.com/solutions/publications/roles/index.html`

[47] NIELSEN, J.: *Usability Engineering*. San Diego: Academic Press, 1993, ISBN 1-12-518406-9.

[48] NIELSEN, Jakob: *10 Usability Heuristics for User Interface Design*. [online], 1995, [cit. 2013-04-27]. Dostupné z: `http://www.nngroup.com/articles/ten-usability-heuristics/`

[49] OBJECT MANAGEMENT GROUP: Unified Modeling Language 2: Superstructure. srpen 2005. Dostupné z: `http://www.omg.org/spec/UML/2.0/Superstructure/PDF/`

[50] PECINOVSKÝ, R.: *Návrhové vzory*. Brno: Computer Press, první vydání, 2007, ISBN 978-80-251-1582-4.

[51] RUBY, S.; THOMAS, D.; HANSSON, D. H.: *Agile Web Development with Rails*. Dallas: The Pragmatic Programmers, čtvrté vydání, 2010, ISBN 978-1-934356-54-8.

[52] RUMBAUGH, J.; JACOBSON, I.; BOOCH, G.: *The Unified Modelling Language Reference Manual*. Boston: Addison-Wesley, druhé vydání, 2005, ISBN 0-321-24562-8.

[53] SAP: *Custom Development Services*. [online], 2013, [cit. 2013-04-02]. Dostupné z: `http://www54.sap.com/services-support/svc/custom-app-development.html`

[54] SAP: *SAP*. [online], 2013, [cit. 2013-03-14]. Dostupné z: `http://www.sap.com/index.epx`

[55] SYCH, O.: *ASP.NET Dynamic Data Unleashed*. Indianapolis: SAMS, 2012, ISBN 978-0-6723-3565-5.

[56] THE JQUERY FOUNDATION: *jQuery*. [online], 2013, [cit. 2013-04-18]. Dostupné z: `http://jquery.com/`

[57] THE JQUERY FOUNDATION: *jQuery user interface*. [online], 2013, [cit. 2013-04-19]. Dostupné z: `http://jqueryui.com/`

[58] TROELSEN, A.: *Pro C# 2010 and the .NET 4 Platform*. New York: Apress, páté vydání, 2010, ISBN 978-1-4302-2550-8.

[59] TROELSEN, A.: *Pro C# 5.0 and the .NET 4.5 Framework*. New York: Apress, šesté vydání, 2012, ISBN 978-1-4302-4234-5.

[60] VOŘÍŠEK, J.: *Principy a modely řízení podnikové informatiky*. Praha: Oeconomica, 2008, ISBN 9788024514406.

[61] W3C: *HTML5*. [online], 2011, [cit. 2013-04-21]. Dostupné z: `http://www.w3.org/TR/2011/WD-html5-20110525/elements.html#embedding-custom-non-visible-data-with-the-data-attributes`

[62] W3C DOM IG: *Document Object Model (DOM)*. [online], 2005, [cit. 2013-04-18]. Dostupné z: `http://www.w3.org/DOM/`

[63] WELLMAN, D.: *jQuery UI 1.8 The User Interface Library for jQuery*. Birmingham: Packt Publishing, 2011, ISBN 978-1-849516-52-5.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

BDD Behaviour-Driven Development

CLR Common Language Runtime

CLS Common Language Specification

CIL Common Intermediate Language

COM Component Object Model

CSS Cascading Style Sheet

CTS Common Type Systém

DCL Data Control Language

DDL Data Definition Language

DI Dependency Injection

DIČ Daňové identifikační číslo

DLL Dynamic-link Library

DML Data Manipulation Language

DMS Document Management Systém

DOM Document Object Model

DUNS Data Universal Numbering System

ECM Enterprise Content Management

EDI Electronic Data Interchange

ERP Enterprise Resource Planning

FTP File Transfer Protocol

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IASW Individuální aplikační software

IATF International Automotive Task Force

ICT Information and Communication Technologies

IČ Identifikační číslo

IDE Integrated Development Environment

IIS Internet Information Services

IMDS International Material Data Systém

IoC Inversion of Control

ISO International Organization for Standardisation

JIT Just-in-Time

JSON JavaScript Object Notation

LINQ Language Integrated Query

MIME Multipurpose Internet Mail Extensions

MVC Model-View-Controller

OHSAS Occupational Health and Safety Advisory Services

OPN Oddělení projektového nákupu

ORM Object-Relational Mapping

REST Representational State Transfer

RIA Rich Internet Application

SP Stored Procedure

SQA Supplier Quality Assurance

SRM Supplier Relationship Management

SQL Structured Query Language

TASW Typový aplikační software

TCL Transaction Control Language

TDD Test-Driven Development

UDF User-Defined Function

UI User Interface

UML Unified Modelling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

VDP Vedoucí dílčího projektu

VPN Virtual Private Network

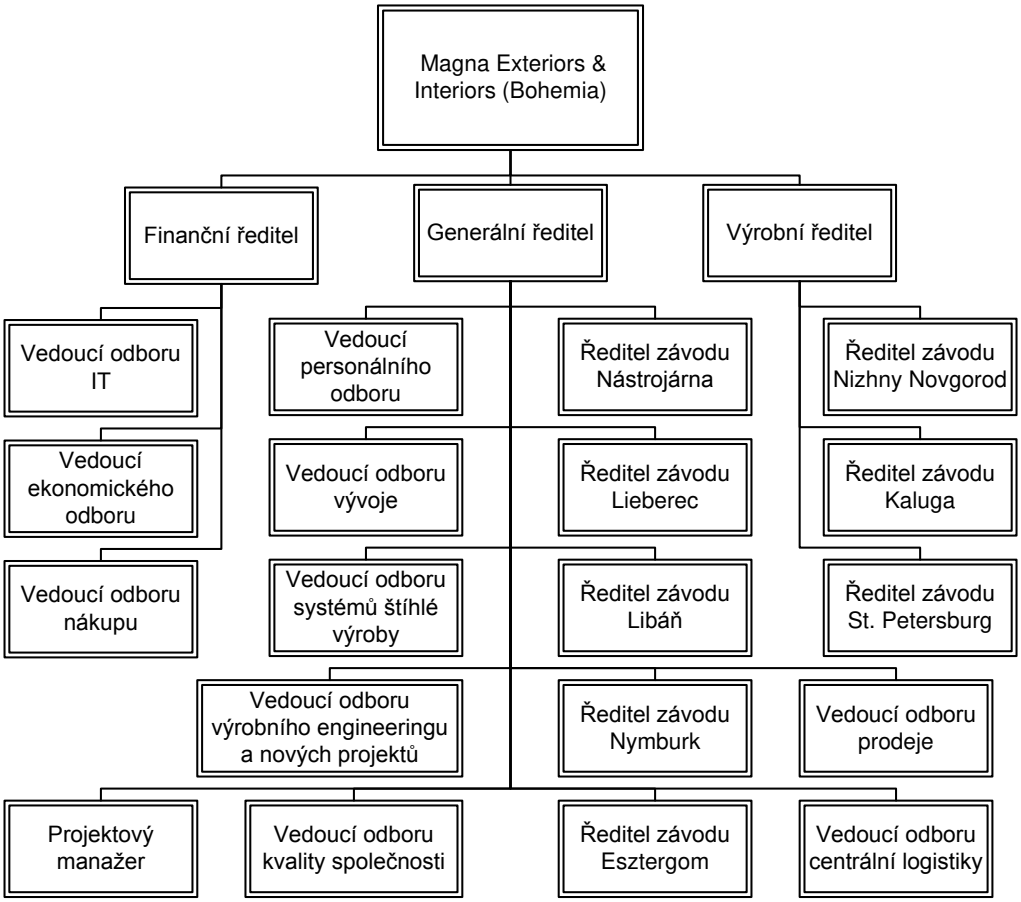
WCF Windows Communication Foundation

WPF Windows Presentation Foundation

XML Extensible Markup Language

Příloha **B**

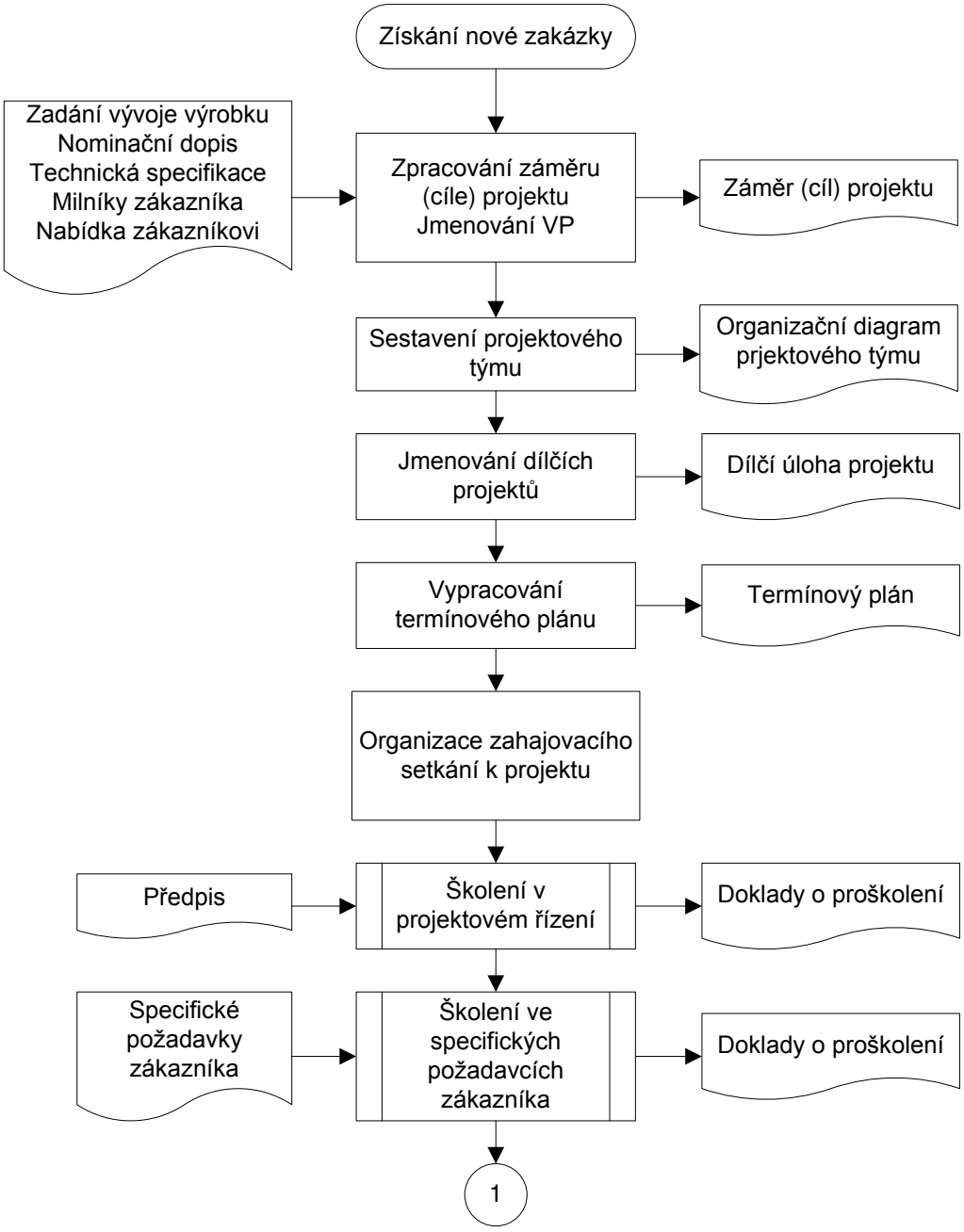
Organizační řád Magna Exteriors & Interiors Bohemia



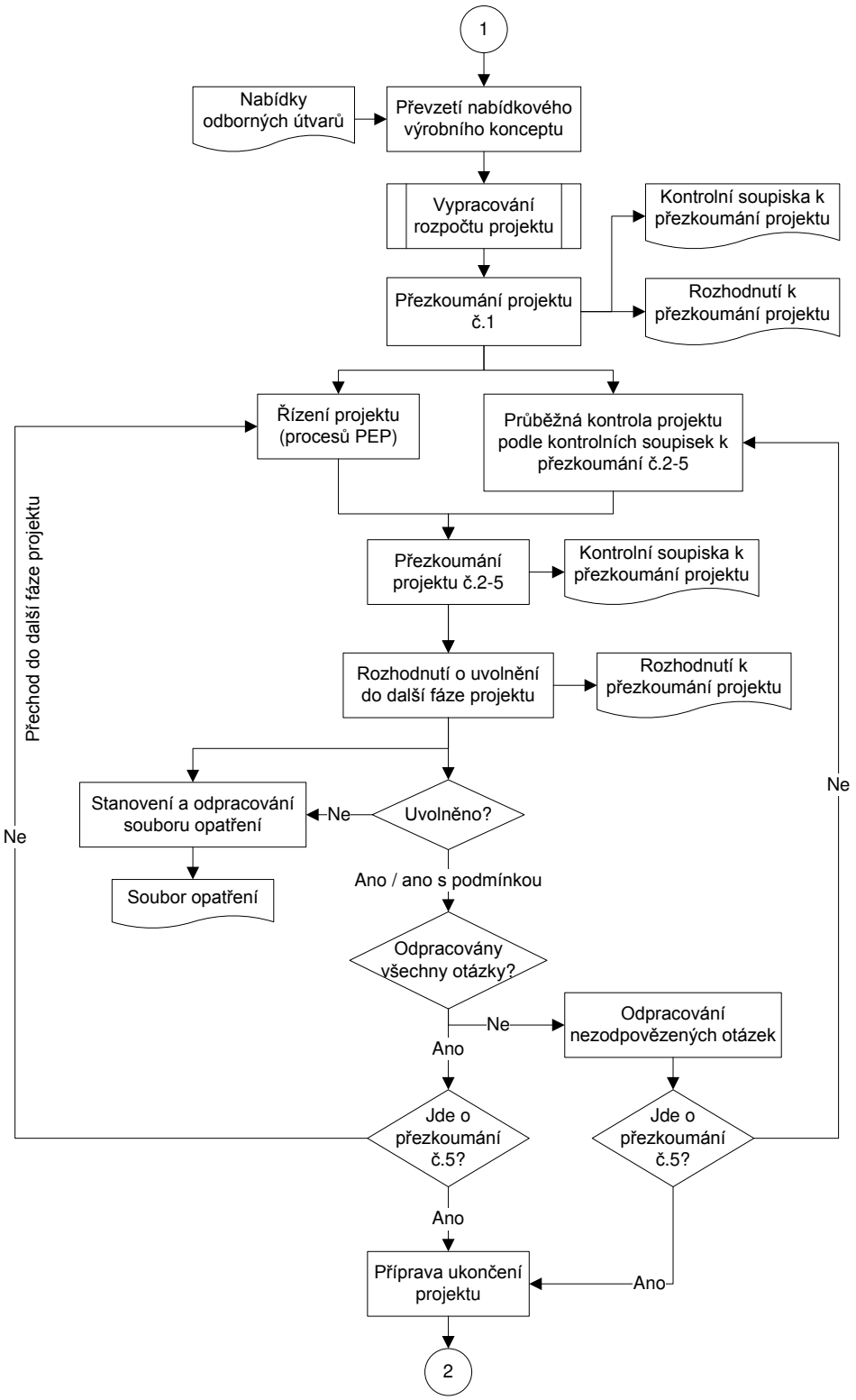
Obrázek B.1: Organizační struktura společnosti Magna Exteriors & Interiors Bohemia s.r.o.

Příloha C

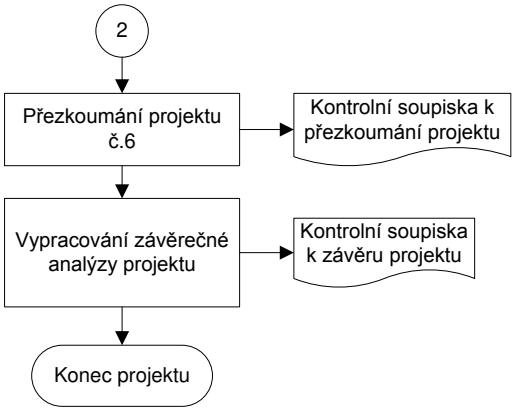
Vývojový diagram procesu Řízení projektu



Obrázek C.1: Vývojový diagram hlavního procesu Řízení projektu – 1. část



Obrázek C.2: Vývojový diagram hlavního procesu Řízení projektu – 2. část



Obrázek C.3: Vývojový diagram hlavního procesu Řízení projektu – 3. část

Příloha D

Kmenová věta materiálu

Příloha č.1 k Interní zprávě Formulář KVM Datum vydání 31.1.2013 - index 1	**KMENOVÁ VĚTA MATERIÁLU**	MAGNA EXTERIORS & INTERIORS

Kmenová věta materiálu Změna/Nový:

suroviny, barvy/laky, nakupované díly, obaly, náhradní díly - ne indirect, HAWA díly

SAP číslo materiálu:		Závod:		Sklad:	
Třida ocenění:				Klíč disp.:	
Projekt:				Nominace dodavatele zákazníkem:	
Název:					
Název 2:					
Zákaznické číslo materiálu:		Číslo výkr.:			
Zákazník:					
MEPI klíč - Komoditní kód:					
Management jakosti:					
				MJ:	
Dodavatel:		Cena :		Měna:	
				Počet MJ v ceně:	
Hmotnost dílu: (g)		Max. zásoba:		Objedn. hladina:	
Profitové centrum:					
D/TLD:		Generační stav:			
Nástroj:		Typ dodávek:			

Poznámka:

Základní (ZU):

Interní (IP):

Kontrolní (TK):

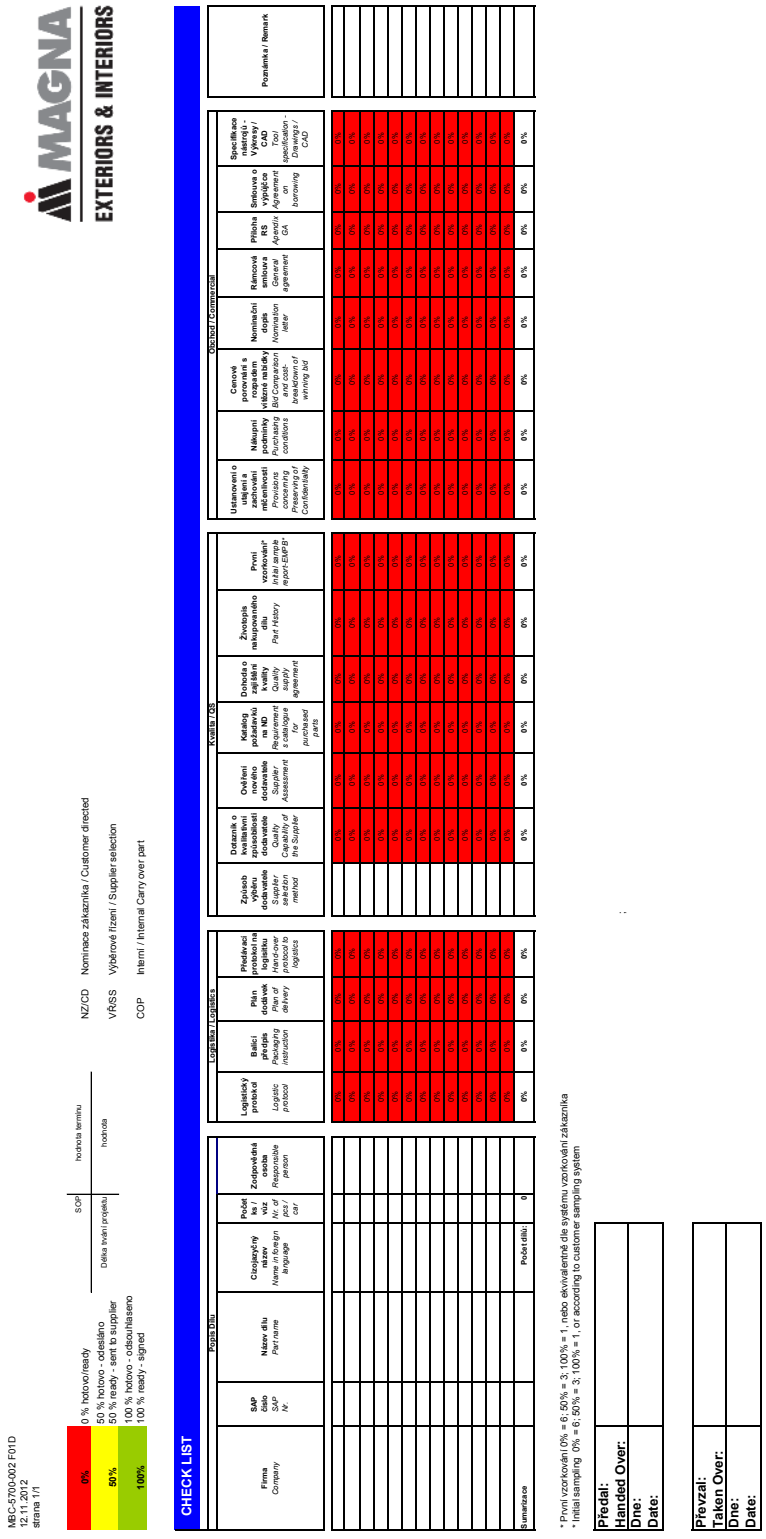
Objednací:

Vystavil:	Dne:
Správu kmen. dat provedl:	Dne:

Obrázek D.1: Formulář Kmenová věta materiálu

Příloha E

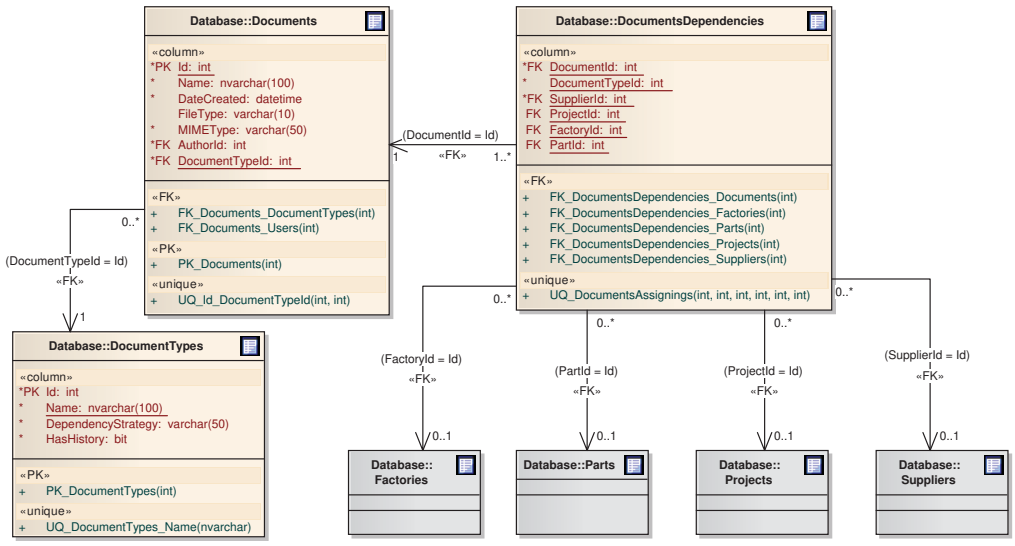
Checklist



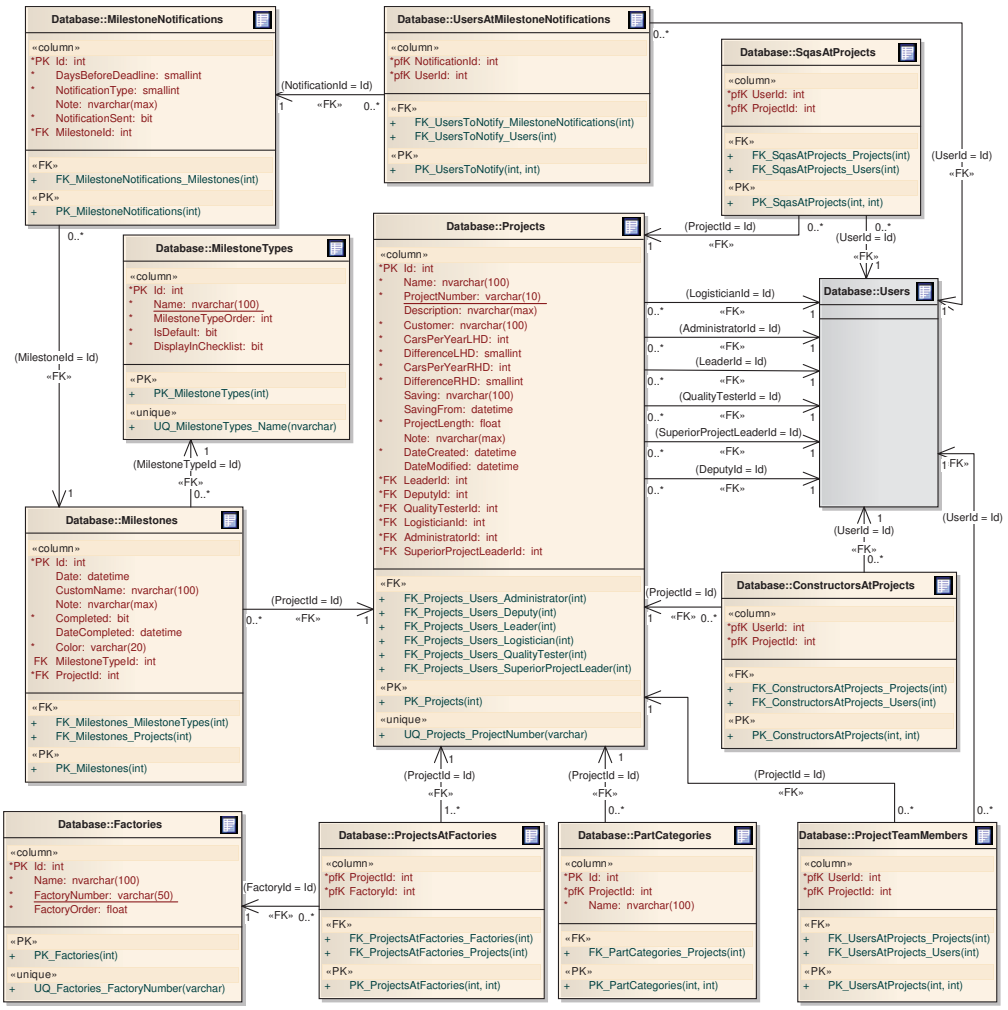
Obrázek E.1: Projektový Checklist

Příloha F

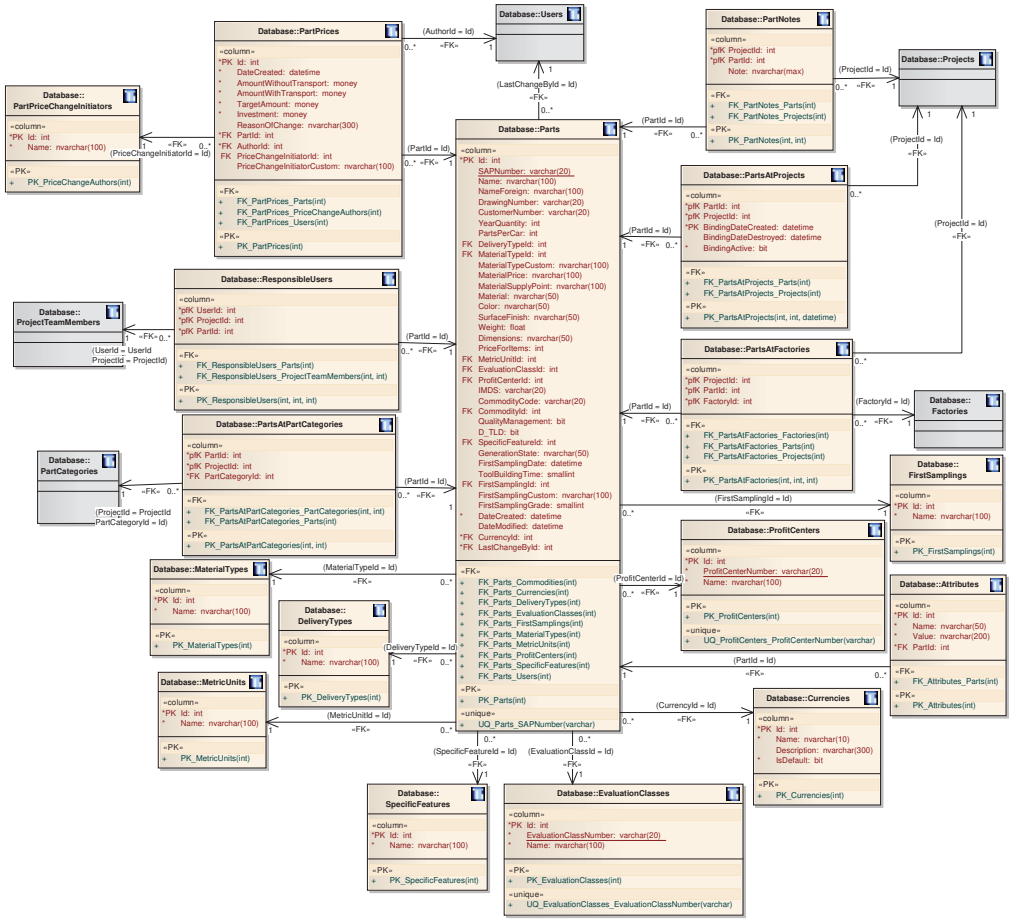
Databázové diagramy



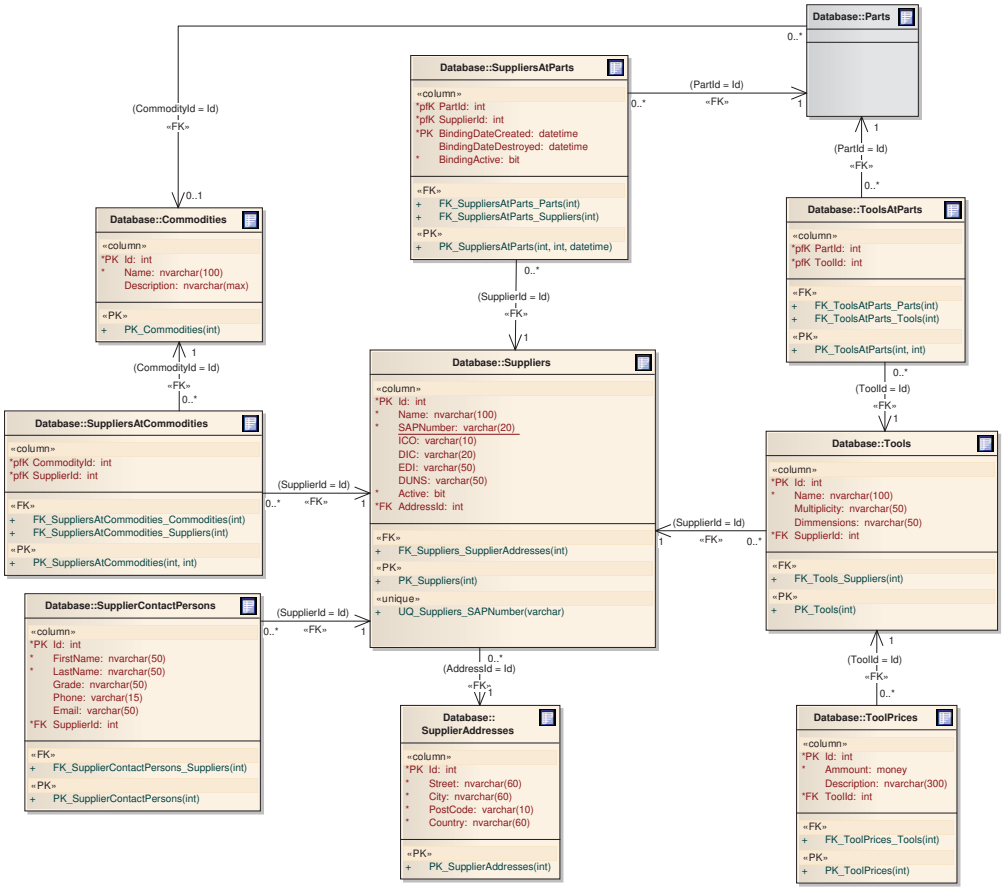
Obrázek F.1: Databázový diagram – dokumenty



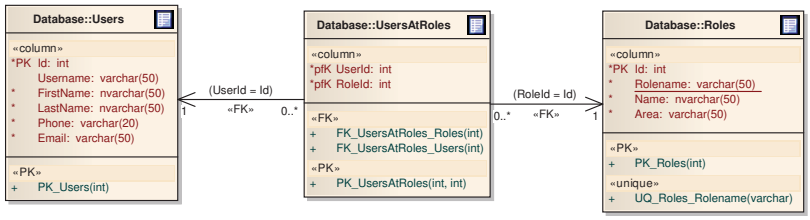
Obrázek F.2: Databázový diagram – projekty



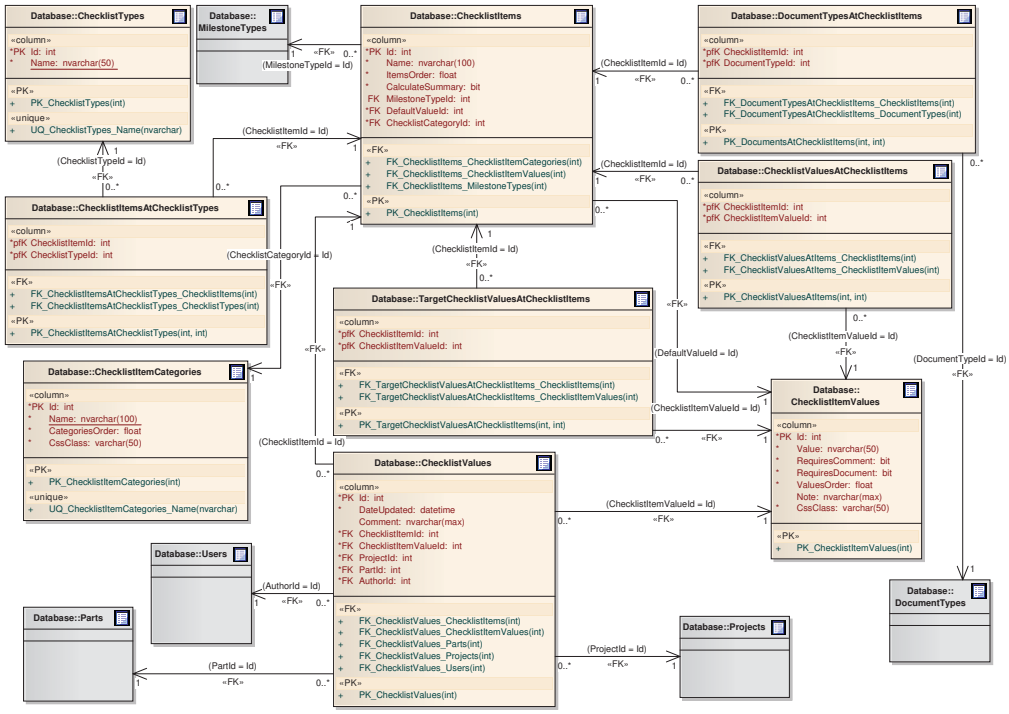
Obrázek F.3: Databázový diagram – díly



Obrázek F.4: Databázový diagram – dodavatelé



Obrázek F.5: Databázový diagram – uživatelé



Obrázek F.6: Databázový diagram – Checklist

Příloha G

Instalační příručka

Následující text slouží správci systému jako návod pro jeho instalaci a zprovoznění na serveru nebo osobním počítači. Popsán je zde také způsob, jak systém připravit k lokálnímu otestování.

G.1 Požadavky systému

Pro úspěšný chod systému jsou vyžadovány následující programy a podpůrné technologie:

- Operační systém: Windows XP SP3 / Vista / 7 / Server 2003 SP 2 / Server 2008 R2.
- .NET Framework 4.0.
- Databázový systém: Microsoft SQL Server 2005 nebo novější.
- Webový server: Microsoft Internet Information Services 7.5 nebo novější.

Vývoj systému probíhal v IDE Microsoft Visual Studio 2012. Pro konfiguraci a úpravu databáze byla použita aplikace Microsoft SQL Server Management Studio. Pro úpravy kódu a lokální testování doporučuji používat uvedené aplikace, aby byla zajištěna kompatibilita. Vývojář by měl rovněž nainstalovat framework ASP.NET MVC 3 dostupný z `http://www.asp.net/mvc/mvc3`. Instalátor zajistí automatickou integraci se zmíněným IDE.

Aplikaci doporučuji spouštět v internetových prohlížečích Firefox nebo Google Chrome nejnovějších verzí. Možné je použít také prohlížeč Microsoft Internet Explorer verze 9 nebo novější. Je však nutné počítat se zhoršenou

kvalitou některých ovládacích prvků (jedná se především o barevné přechody některých tlačítek). Aplikace vyžaduje zapnutý JavaScript a povolené soubory cookie.

G.2 Instalace systému pro používání

Dále je popsán návod jak postupovat v případě zavádění systému na server nebo lokální počítač:

1. Příprava databáze – nejprve je nutné připravit databázi systému. Návod popisuje jako postupovat při použití aplikace Microsoft SQL Server Management Studio.

   1.1. Spusťte aplikaci Microsoft SQL Server Management Studio a přihlaste k serveru, kde má být umístěná databáze.

   1.2. Vytvořte databázi:

```
1 CREATE DATABASE [nazev_databaze]
```

   1.3. Vytvořte login:

```
1 CREATE LOGIN [uzivatelske_jmeno] WITH
2 PASSWORD='heslo',
3 CHECK_EXPIRATION=OFF
```

   1.4. Vytvořte uživatele ve dříve vytvořené databázi a přidělte mu potřebná práva:

```
1 USE [nazev_databaze]
2 CREATE USER [uzivatelske_jmeno] FROM LOGIN [uzivatelske_jmeno]
3 GRANT EXECUTE TO [uzivatelske_jmeno]
```

   1.5. Vytvořte tabulky – otevřete soubor db_init_tables.sql uložený na přiloženém CD ve složce app/db a spusťte všechny příkazy.

   1.6. Vytvořte procedury – otevřete soubor db_init_procedures.sql uložený na přiloženém CD ve složce app/db a spusťte všechny příkazy.

   1.7. Naplňte tabulky výchozími daty – otevřete soubor db_init_data.sql uložený na přiloženém CD ve složce app/db a spusťte všechny příkazy.

   1.8. Vytvořte uživatele aplikace – použijte následující příkaz, kde „uzivatelske_jmeno“ nahraďte uživatelským jménem, který používáte pro přihlašování do systému Windows, na kterém aplikace

poběží. Jméno vyplňte v tradičním formátu „domena/uzivatelske_jmeno“. Nahraďte také „email“, „jmeno“, „prijmeni“ a „telefon“ požadovanými údaji.

```
1 USE [nazev_databaze]
2 INSERT INTO [Users] ([Username], [Email], [FirstName], [LastName
      ], [Phone]) VALUES
3   (N'uzivatelske_jmeno', N'email', N'jmeno', N'prijmeni', N'
        telefon')
4 INSERT INTO [UsersAtRoles] ([UserId], [RoleId]) VALUES (1, 1)
5 INSERT INTO [UsersAtRoles] ([UserId], [RoleId]) VALUES (1, 9)
```

1.9. Upravte proceduru pro zasílání upozornění na blížící se termín – procedura MilestoneNotification_SendNotifications je přizpůsobena použití na serveru společnosti Magna E&I a je tedy nutné upravit způsob zasílání vygenerovaných emailů. Rovněž je nutné nastavit pravidelné automatické spouštění této procedury na každý den po půlnoci.

2. Příprava webového serveru – dalším krokem je vytvoření webové aplikace na serveru Internet Information Services 7.5 a její nastavení.

   2.1. Na disku vytvořte adresář webové aplikace (například v adresáři C:
   inetpub
   wwwroot) a zkopírujte do něj všechny soubory umístěné na přiloženém CD ve složce app/deploy.

   2.2. Spusťte aplikaci Internet Information Services Manager (Správce internetových informačních služeb).

   2.3. Vytvořte nový Application pool (Fond aplikací) – libovolně jej pojmenujte, zvolte .NET Framework verze 4 a jako Managed pipeline mode (spravovaný režim kanálů) ponechte Integrated (Integrovaný).

   2.4. Vytvořte webovou aplikaci – pod libovolným webem (můžete ponechat Default Web Site) vytvořte novou aplikaci. Vyplňte vhodný Alias, vyberte v předchozím kroku vytvořený Application pool a zadejte cestu vedoucí k adresáři vytvořenému v kroku 2.1.

   2.5. Nastavte webovou aplikaci – jediné nastavení, které je potřeba upravit je způsob ověřování uživatelů. Pod možností Authentication (Ověřování) povolte Windows Authentication (Ověřování systému Windows).

3. Konfigurace aplikace – posledním krokem je úprava souboru Web.config, který jste v předchozím kroku umístili do vytvořeného adresáře.

3.1. Vytvořte adresář, kam mají být ukládány dokumenty nahrané do aplikace.

3.2. V libovolném textovém editoru otevřete soubor Web.config.

3.3. Upravte connection string pro připojení k databázi – v souboru Web.config vyhledejte sekci `<connectionStrings>` a v ní jediný connection string nazvaný „MagnaEntities“. V atributu connectionString nahraďte následující úseky: „databazovy_server“ za úplný název serveru, kde jste v kroku 1.2 vytvořili databázi, „nazev_databaze“ za název vytvořené databáze, „uzivatelske_jmeno“ za jméno uživatele vytvořeného v kroku 1.4 a „heslo“ za jeho heslo zadané v kroku 1.3.

3.4. Zadejte cestu k adresáři pro ukládání dokumentů – v souboru Web.config vyhledejte sekci `<documentsConfig>` a v ní element `<root>`. Do jeho atributu path zadejte úplnou cestu k adresáři vytvořenému v kroku 3.1.

Nyní by aplikace měla být správně nastavena a připravena k použití.

G.3 Příprava systému pro lokální testování a úpravy

Zdrojové soubory aplikace jsou umístěny na přiloženém CD ve složce src/impl. Nachází se zde soubor Magna.ProjectManagement.sln představující projekt pro IDE Microsoft Visual Studio 2012. Pro zprovoznění projektu postupujte následovně:

1. Připravte databázi podle kroku 1 předchozího návodu.
2. Spusťte program Microsoft Visual Studio 2012 a otevřete soubor Magna.ProjectManagement.sln.
3. Upravte soubory:
   Web.config v projektu Magna.ProjectManagement.UI.WebMVC,
   app.config v projektu Magna.ProjectManagement.UI.Console
   a app.config v projektu Magna.ProjectManagement.Tests
   podle kroků 3.3 a 3.4 předchozího návodu.

Nyní by aplikace měla být připravena ke spuštění. Pokud chcete spouštět jednotkové testy a testy připravené ve frameworku Selenium, pokračujte následujícími kroky.

4. Vytvořte požadované procedury – v aplikaci Microsoft SQL Server Management Studio otevřete soubor db_init_procedures_tests.sql a vytvořte v něm zapsané procedury pod databází, kterou aplikace využívá. V proceduře DbInit je třeba upravit údaje vytvářeného uživatele (postupujte stejně jako v kroku 1.8 předchozího návodu).

5. Upravte konfiguraci testů ve frameworku Selenium – upravte hodnoty vlastností _testFilePath, _username a _url ve třídě Magna.ProjectManagement.Tests.Infrastructure.SeleniumTestsBase (soubor Infrastructure/SeleniumTestsBase.cs v projektu Magna.ProjectManagement.Tests).

Příloha H

Uživatelská příručka

Následující text slouží jako manuál pro uživatele aplikace. Představuje mu uživatelské rozhraní, některé ovládací prvky a radí, jak vykonávat některé akce. V textu je používáno slovo „entita“, se kterým se čtenář dosud nemusel setkat. Označuje nějaký obecný objekt reálného světa, se kterým se v aplikaci pracuje – konkrétně se může jednat například o projekt, termín, díl, dodavatele, dokument a podobně.

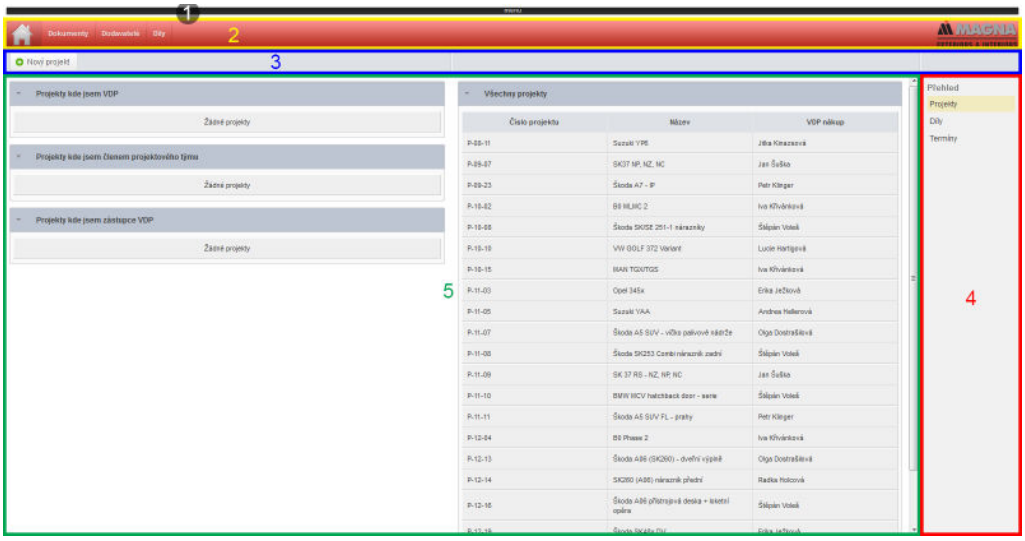
H.1 Přihlášení do aplikace

Aplikace je dostupná na adrese `http://mczmsint02.magna.global/Nakup/`. Po zadání adresy do internetového prohlížeče je uživatel vyzván k zadání Uživatelského jména a hesla. Jako jméno zadá své osobní číslo včetně počátečního písmene „P“ (případně může zadat uživatelské jméno včetně domény LIBEREC ve tvaru „LIBEREC/osobni_cislo“). Heslo použije stejné, kterým se přihlašuje do systému Windows.

Po přihlášení je uživateli zobrazena úvodní strana, kde vidí odkazy na sekce, do kterých má přístup. Tato příručka slouží uživatelům sekce Nákupčí a v dalším textu se předpokládá, že se uživatel pohybuje v této sekci. Sekce je dostupná na adrese `http://mczmsint02.magna.global/Nakup/ProjectManagement`.

H.2 Rozložení uživatelského rozhraní

Uživatelské rozhraní jednotlivých stránek se obecně může skládat z následujících oblastí (viz obrázek H.1):



Obrázek H.1: Rozložení uživatelského rozhraní

- Skryté menu (1) – v horní části obrazovky se nachází tmavá lišta s nápisem „menu“. Po kliknutí na ní se zobrazí menu s odkazy na dostupné sekce a údaji o přihlášeném uživateli.

- Hlavní menu (2) – červený pruh v horní části obrazovky představuje hlavní menu aplikace, které se mění v závislosti na prohlížené stránce. První tlačítko směřuje na úvodní stránku sekce – přehled projektů.

- Menu akcí (3) – na některých stránkách je pod hlavním menu přítomen šedý pruh s tlačítky akcí, které souvisí s aktuálně zobrazenými informacemi a umožňují například jejich úpravu a podobně.

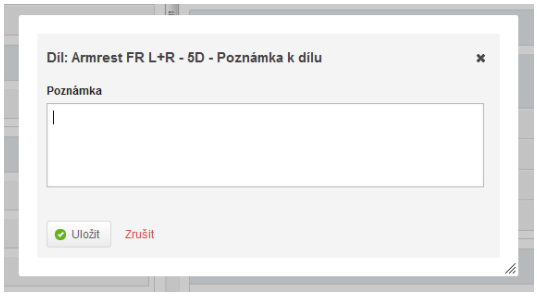
- Vertikální menu (4) – na některých stránkách je k dispozici pravé vertikální menu, které nabízí akce související s aktuálním přehledem. Umožňuje například řazení, přepnutí způsobu zobrazení a podobně.

- Hlavní obsah (5) – hlavní obsah stránky tvoří vždy jeden nebo dva nezávislé sloupce.

H.3 Ovládací prvky aplikace

Při práci s aplikací se uživatel bude opakovaně setkávat s následujícími ovládacími prvky:

Obrázek H.2: Sbalovací box



Obrázek H.3: Dialogové okno

Sbalovací box

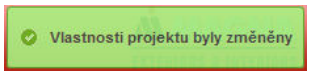
Sbalovací boxy (viz obrázek H.2) ohraničují skupinu souvisejících dat či entit. Jejich obsah lze skrýt a opětovně zobrazit kliknutím na modrošedé záhlaví boxu. V pravé části záhlaví mohou být umístěna tlačítka pro vyvolání akcí přímo souvisejících s obsahem boxu. Kliknutím na tlačítko dojde ve většině případů k otevření nějakého dialogového okna.

Dialogové okno

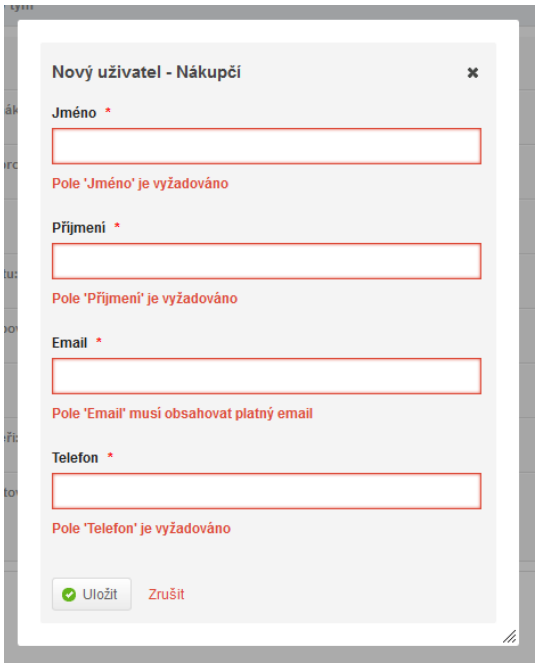
Dialogová okna (viz obrázek H.3) jsou používána napříč celou aplikací k úpravě vlastností entit nebo nahlížení na detail entity. V záhlaví okna je zobrazeno, se kterou konkrétní entitou okno souvisí. V případě okna s formulářem je v zápatí tlačítko „Uložit“ pro potvrzení provedených změn a tlačítko „Zrušit“ pro zrušení provedených změn a uzavření okna. Okno je možné zavřít křížkem v pravém horním rohu.

Notifikace

O úspěšnosti provedených akcí systém informuje notifikací krátce zobrazenou v pravém horním rohu obrazovky v místě loga společnosti (viz obrázek H.4). Zelená barva notifikace značí úspěch, červená barva značí neúspěch.



Obrázek H.4: Notifikace



Obrázek H.5: Zadávání vstupu

Zadávání vstupu

Pro zadávání vstupu slouží formuláře otevírané v dialogových oknech. Povinná pole, která musí být vždy vyplněna, jsou označena červenou hvězdičkou. Pokud uživatel zadá neplatnou hodnotu, je ihned informován červeně psaným hlášením. Kolem pole se špatně vyplněnou hodnotu se také zobrazí červený rámeček. (viz obrázek H.5)

Výběr z mnoha položek

Například pro výběr závodů v projektu (viz obrázek H.6) nebo výběr členů projektového týmu slouží speciální dialogová okna se seznamem položek. Položku uživatel vybere nebo odebere kliknutím na ni. Tmavě žlutě označené položky jsou vybrané, šedé položky jsou nevybrané. Světle žluté pozadí a šedý text označuje vybrané položky, které není možné momentálně odebrat. Důvodem může být například to, že jsou přiřazeny k jiným entitám.

Obrázek H.6: Výběr závodů v projektu

Výběr položek uživatel potvrdí tlačítkem „Uložit“.

H.4 Úvodní obrazovka

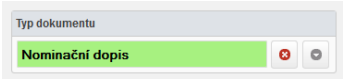
Na úvodní obrazovce aplikace je zobrazen přehled existujících projektů a je možné v pravém menu přepnout na zobrazení dílů a termínů. Zobrazení obsahují následující přehledy:

- Projekty – Levý sloupec: projekty, kde je přihlášený uživatel VDP, projekty, kde je člen projektového týmu a projekty, kde je zástupcem VDP. Pravý sloupec: všechny projekty.
- Díly – Levý sloupec: Díly rozdělené dle projektů, za které je přihlášený uživatel zodpovědný. Pravý sloupec: všechny díly rozdělené dle projektů.
- Termíny – Levý sloupec: Termíny rozdělené dle projektů, kde je přihlášený uživatel VDP, zástupce VDP nebo člen projektového týmu. Pravý sloupec: zpožděné termín rozdělené dle projektů, budoucí termín rozdělené dle projektů a dokončené termíny rozdělené dle projektů.

H.5 Vyhledávání

V aplikaci existují tři samostatné stránky pro vyhledávání dokumentů, dodavatelů a dílů. Princip vyhledávání na stránkách je stejný, liší se pouze

Obrázek H.7: Filtrovací boxy



Obrázek H.8: Filtrovací box s vybranou položkou

složení filtrovacích boxů. Filtrovací boxy (viz obrázek H.7) slouží k upřesnění kritérií, podle kterých budou entity vyhledávány. Kliknutím na šipku v pravé části boxu je možné zobrazit všechny položky, které jsou k dispozici. Kliknutím na položku dojde k jejímu vybrání (viz obrázek H.8). Zrušení výběru je možné provést kliknutím na červený křížek v pravé části boxu. Kliknutím do bílého textového pole v boxu a psaním se pod boxem zobrazují obsahující zadávaný text. Položky je opět možné vybrat kliknutím. Také je možné použít klávesy šipek nahoru a dolů pro pohyb mezi položkami a klávesu enter pro jejich výběr.

Všechny boxy zobrazené na jedné stránce jsou vždy navzájem závislé. To znamená, že při výběru položky v jednom boxu, budou v ostatních boxech na výběr pouze takové položky, které nějakým způsobem souvisí s vybranou položkou. (Například vybereme-li ve vyhledávání dokumentu typ dokumentu „Nominační dopis“, pak v boxu dodavatel budou na výběr pouze dodavatelé, kteří mají přiřazený nějaký nominační dopis. V boxu projekt budou na výběr pouze projekty, ve kterých se vykytují díly, které dodává nějaký dodavatel s dokumentem typu „Nominační dopis“. A v boxu díl budou na výběr pouze díly, které dodávají dodavatelé s dokumentem typu „Nominační dopis“. Vybereme-li nějakého dodavatele, pak v boxu projekt budou na výběr pouze projekty, které obsahují díly, které dodává daný dodavatel.)

Samotné vyhledání se provede kliknutím na tlačítko „Vyhledat“, přičemž se vyhledají pouze entity odpovídající kritériím zadaným ve filtrovacích boxech.

Pokud ponecháme zaškrtnuté pole „Pouze aktivní“, pak se vyhledávají pouze entity s aktivní vazbou. V případě dokumentů se tak například nevyhledávají dokumenty přiřazené dílům, které se momentálně nevyskytují

Obrázek H.9: Výsledek vyhledávání dokumentů

v žádném projektu. V případě dodavatelů se vyhledává pouze mezi aktivními dodavateli, kteří dodávají alespoň jeden díl. A v případě dílů se vyhledává pouze mezi díly vyskytujícími se v alespoň jednom projektu.

H.5.1 Dokumenty

Dokumenty lze filtrovat podle:

- Typu dokumentu – do pole se zadává název typu dokumentu.
- Dodavatele – do pole se zadává název nebo číslo dodavatele.
- Projektu – do pole se zadává název nebo číslo projektu.
- Dílu – do pole se zadává název nebo číslo dílu.

Po vyhledání se zobrazí dokumenty rozdělené dle typů (viz obrázek H.9). Dokument je možné stáhnout kliknutím na tlačítko s ikonou diskety a jeho názvem. Viditelné jsou také názvy entit, na kterých dokument závisí. Názvy fungují zároveň jako odkazy vedoucí na detail entity.

H.5.2 Dodavatelé

Dodavatele lze filtrovat podle:

- Dodavatele – do pole se zadává název nebo číslo dodavatele.
- Projektu – do pole se zadává název nebo číslo projektu.

Po vyhledání se zobrazí seznam dodavatelů s obecnými informacemi (viz obrázek H.10). Kliknutím na název dodavatele lze přejít k obecnému detailu dodavatele, kde lze upravit jeho vlastnosti, adresu, kontaktní osoby a komodity. Jsou zde také vidět všechny díly, které dodává a přiřazené dokumenty, které nezávisí na projektu ani na dílu.

Na stránce Dodavatelé je také možné vytvořit nového dodavatele a importovat seznam dodavatelů ze souboru .CSV. Do souboru se vždy na jeden řádek zadávají vlastnosti dodavatele. První dodavatel se vkládá na druhý řádek. Význam sloupečků v souboru je postupně od prvního následující:

Obrázek H.10: Výsledek vyhledávání dodavatelů

Obrázek H.11: Výsledek vyhledávání dílů

SAP číslo (povinné), název dodavatele (povinné), ulice (povinné), město (povinné), PSČ (povinné), země (povinné), IČO (nepovinné), DIČ (nepovinné) a DUNS (nepovinné).

H.5.3 Díly

Dodavatele lze filtrovat podle:

- Dílu – do pole se zadává název nebo číslo dílu.
- Dodavatele – do pole se zadává název nebo číslo dodavatele.
- Projektu – do pole se zadává název nebo číslo projektu.

Po vyhledání se zobrazí seznam dílů s obecnými informacemi (viz obrázek H.11). Kliknutím na název dílu lze přejít k obecnému detailu dílu, kde jsou zobrazeny jeho vlastnosti, ceny, dodavatel, přiřazená zařízení a projekty, ve kterých se díl vykytuje.

H.6 Založení projektu

K založení projektu slouží tlačítko „Nový projekt“ na úvodní stránce s přehledem projektů. Vytvoření projektu je rozděleno do čtyř kroků. Mezi jednotlivými kroky se lze pohybovat tlačítky „Další“ a „Předchozí“, ale pouze pokud jsou všechna pole v aktuálním kroku správně vyplněna. V posledním kroku se tlačítko „Další“ změní na „Dokončit“ a kliknutím na něj dojde k vytvoření projektu. Následuje popis jednotlivých kroků:

1. Vlastnosti projektu – zde je nutné vyplnit číslo projektu, název projektu, zákazníka, délku projektu a vybrat alespoň jeden závod.

2. Projektový tým – všechny pozice jsou povinné kromě SQA. Pokud není na výběr požadovaná osoba, lze ji přidat. Kliknutím na tlačítko se zeleným plus se zobrazí formulář, kde se vyplní jméno, příjmení, email a telefon.

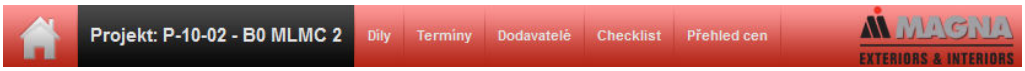
3. Termíny – automaticky jsou vyplněny všechny výchozí termíny. Termín lze odebrat kliknutím na tlačítko „Odstranit“ nebo přidat tlačítkem „Přidat“. Každý typ termínu se v projektu může vyskytovat pouze jednou. Povinnou vlastností termínu je pouze jeho název, přičemž může být vybráno z předdefinovaných typů nebo může být zadán vlastní.

4. Díly – do projektu lze přidat neomezené množství dílů. Díly zde přidávané jsou nové, které v systému dosud neexistují. Existující díly je možné přidat po založení projektu. Povinnou vlastností dílu je alespoň jeden název – český nebo cizojazyčný a alespoň jeden zodpovědný nákupčí. SAP číslo dílu je možné doplnit později.

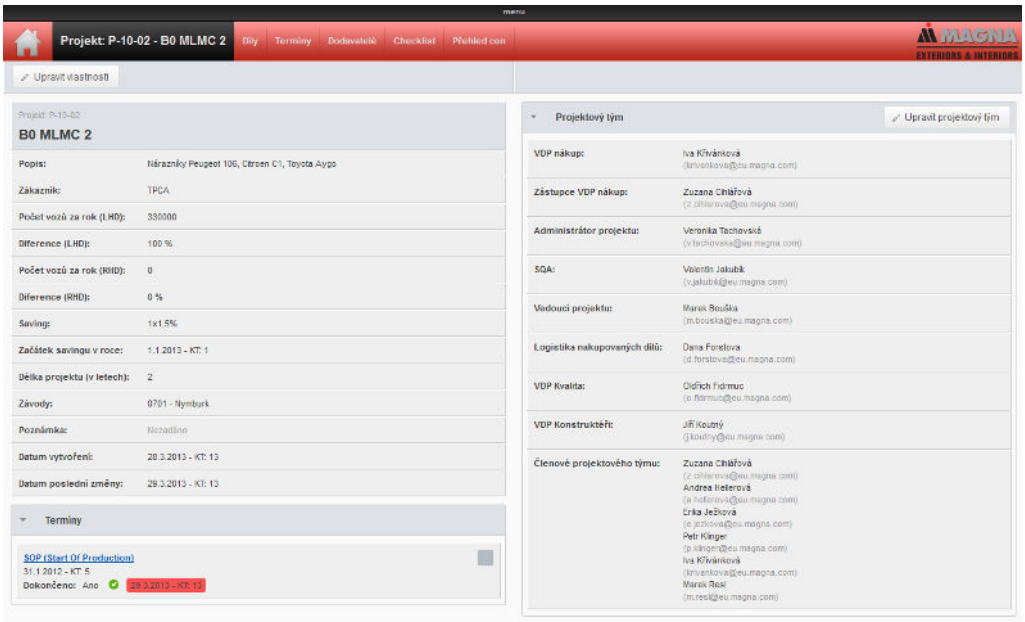
H.7 Detail projektu

Detail projektu se zobrazí kliknutím na požadovaný projekt v přehledu projektů na úvodní stránce. Po zobrazení projektu dojde ke změně hlavního menu (viz obrázek H.12). Všechny položky menu souvisí se zobrazeným projektem. Lze navigovat na detail projektu, přehled dílů, přehled termínů, přehled dodavatelů dodávajících díly v projektu, Checklist a přehled cen.

Na stránce detailu projektu (obrázek H.13) jsou zobrazeny vlastnosti projektu, projektový tým a přehled termínů. VDP a zástupce VDP má možnost upravovat vlastnosti projektu (tlačítko Upravit vlastnosti) i složení projektového týmu (tlačítko Upravit projektový tým).



Obrázek H.12: Hlavní menu v detailu projektu



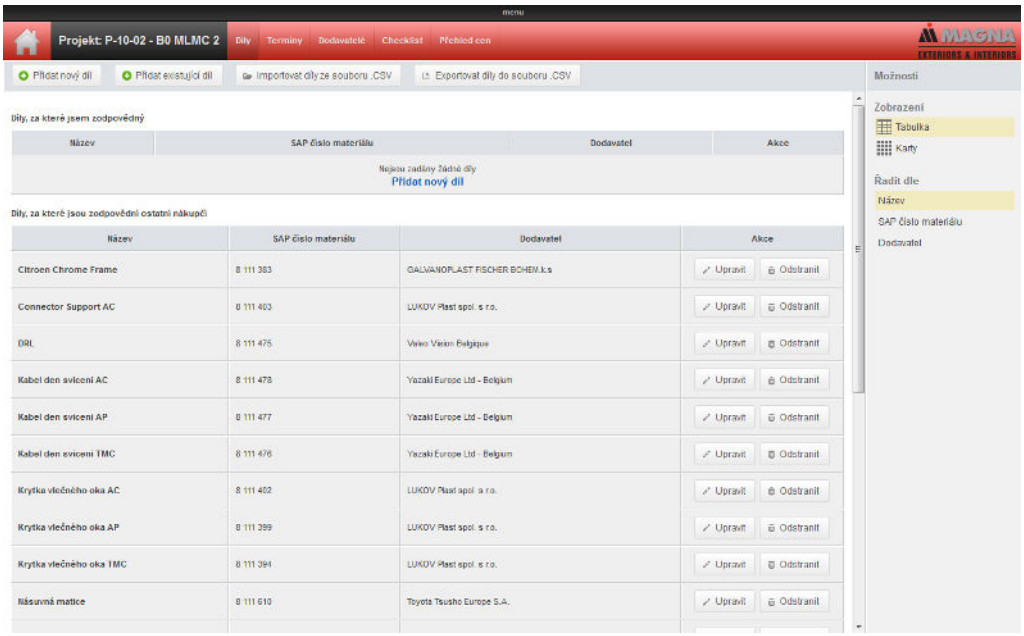
Obrázek H.13: Stránka detailu projektu

H.8 Přehled dílů

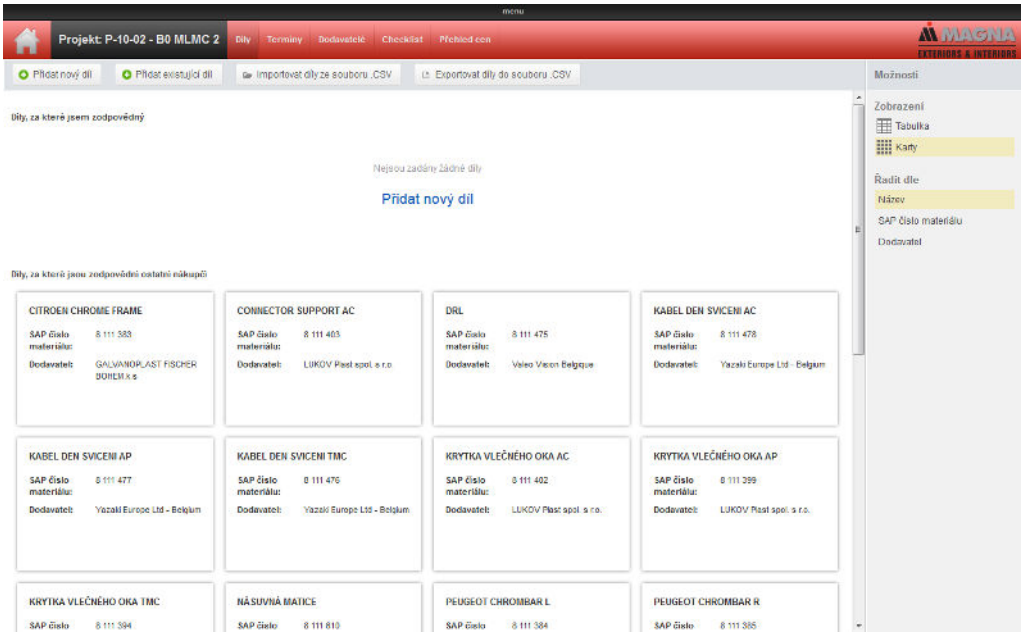
Na stránce Díly jsou zobrazeny všechny díly, které jsou přidané do projektu. Díly jsou rozděleny do dvou skupin. V první skupině jsou zobrazeny díly, za které je zodpovědný přihlášený uživatel. Ve druhé skupině jsou ostatní díly. V pravém menu lze vybírat mezi tabulkovým zobrazením (viz obrázek H.14) a zobrazením karet (viz obrázek H.15) a řadit díly podle názvu, SAP čísla a názvu dodavatele. Kliknutím na díl lze přejít na jeho detail.

Správa dílů

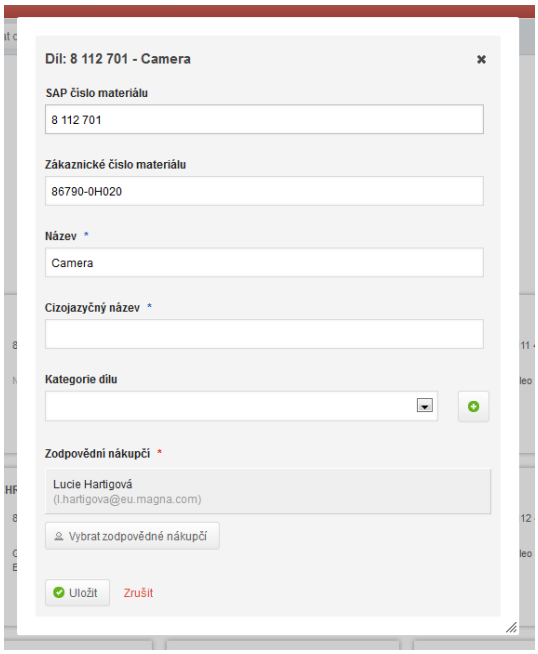
VDP a zástupce VDP má možnost do projektu přidávat nové (tlačítko Přidat nový díl) i existující díly (tlačítko Přidat existující díl). Každý stávající díl může také upravit (tlačítko Upravit) a odstranit (tlačítko Odstranit). Upravuje se SAP číslo materiálu, zákaznické číslo materiálu, název, cizojazyčný název, kategorie dílu a zodpovědní nákupčí (viz obrázek H.16). Odstraněním dílu nedojde k jeho úplnému smazání, díl je pouze odebrán



Obrázek H.14: Stránka přehledu dílů – tabulkové zobrazení



Obrázek H.15: Stránka přehledu dílů – zobrazení karet



Obrázek H.16: Dialogové okno úpravy základních vlastností dílu

z projektu a je možné jej opět nalézt mezi existujícími díly a přidat.

Import a export dílů

VDP a zástupce VDP rovněž mohou využít funkce importování (tlačítko Importovat díly ze souboru .CSV) a exportování (tlačítko Exportovat díly do souboru .CSV) dílů. Do souboru se na každý řádek vkládá jeden díl, přičemž první řádek slouží jako záhlaví a první díl se zapisuje na druhý řádek. Význam jednotlivých sloupečků je postupně od prvního následující: SAP číslo dílu, zákaznické číslo materiálu, číslo výkresu, název, cizojazyčný název, zodpovědný nákupčí (ve tvaru jméno příjmení), kategorie dílu, závod (ve tvaru číslo závodu), množství za rok, hmotnost dílu, dodavatel (ve tvaru SAP číslo dodavatele). SAP číslo dílu je vždy povinné. Číslo závodu a dodavatele musí odpovídat nějakému v systému existujícímu. Zodpovědný nákupčí musí být členem projektového týmu projektu, do kterého díly importujeme. Pokud název kategorie odpovídá nějaké v projektu existující kategorii, je do ní díl zařazen. Pokud kategorie dosud neexistuje, je nejprve vytvořena.

Importováním je možné vytvářet a přidávat nové díly, přidávat existující díly a upravovat stávající díly.

- Přidání nového dílu: Pokud je v souboru zapsán díl se SAP číslem, které dosud není zaneseno do systému, je vytvořen zcela nový díl a je přiřazen k projektu. V tomto případě je povinné vyplnit SAP číslo dílu, alespoň jeden z názvů dílu a zodpovědného nákupčího.

- Přidání existujícího dílu: Pokud je v souboru zapsán díl se SAP číslem, které již v systému existuje, je díl pouze přiřazen k projektu. V tomto případě je povinné vyplnit SAP číslo dílu a zodpovědného nákupčího. Ostatní hodnoty vlastností, které nejsou vyplněny, budou přejaty z aktuálních vlastností dílu (kromě závodu a kategorie, které jsou specifické pro projekt). Pokud je některá vlastnost vyplněna, pak tato hodnota nahradí stávající hodnotu vlastnosti.

- Upravení stávajících dílů: Pokud je v souboru zapsán díl se SAP číslem, který je již součástí projektu, pak pouze dojde k upravení vlastností tohoto dílu. V tomto případě je povinné vyplnit pouze SAP číslo dílu. Opět jako v předchozím bodě platí, vlastnosti, jejichž hodnoty nejsou vyplněné, zůstanou zachovány. Vlastnosti, jejichž hodnoty jsou vyplněné, budou upraveny.

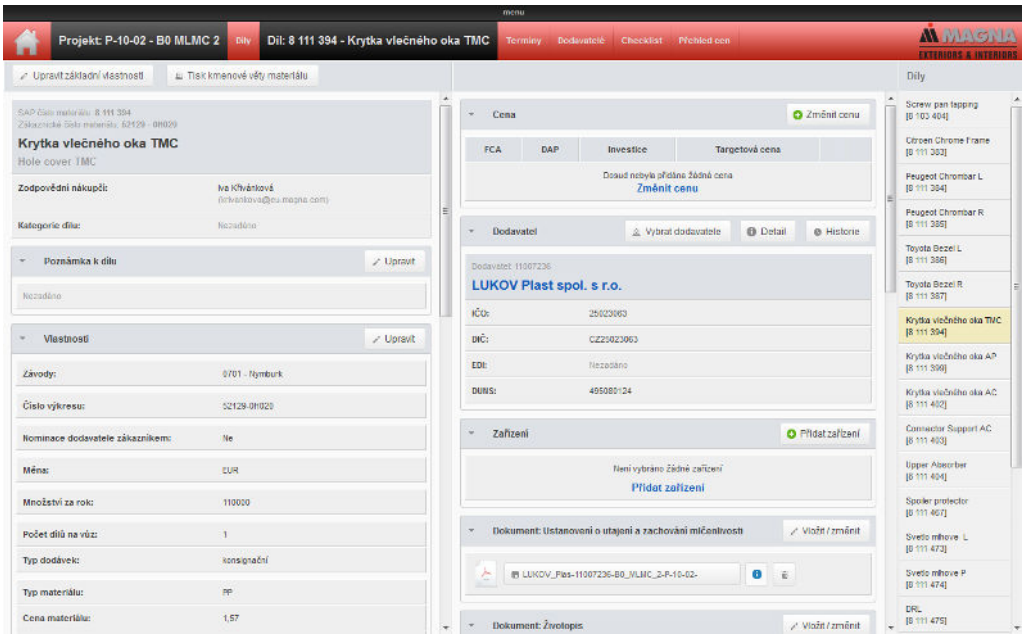
Funkce exportování dílů vytvoří soubor .CSV ve výše popsaném formátu a naplní jej všemi díly, které jsou součástí projektu. Soubor je nabídnut ke stažení.

H.9 Detail dílu

Na stránce detailu dílu v projektu (viz obrázek H.17) jsou zobrazeny všechny informace o dílu. Přehled všech dílů v projektu umístěný v pravé části stránky lze použít pro rychlou navigaci mezi detaily dílů. VDP a zástupce VDP mohou upravovat základní vlastnosti dílu (tlačítko Upravit základní vlastnosti). Upravují se stejné vlastnosti jako na stránce přehledu dílů.

Kmenová věta materiálu

Kterýkoliv uživatel systému má možnost zobrazit si kmenovou větu materiálu přizpůsobenou pro tisk (tlačítko Tisk kmenové věty materiálu), jejíž podoba odpovídá aktuálnímu předpisu (viz obrázek H.18). Většina vlastností dílu je automaticky vyplněna. Některé vlastnosti lze upravit ručně kliknutím na ně, jako je to ukázáno na obrázku H.19. Změny se potvrdí tlačítkem se zeleným symbolem zaškrtnutí nebo zruším tlačítkem s červeným křížkem. Lze upravit následující vlastnosti: Změna/Nový, Sklad, Klíč



Obrázek H.17: Stránka detailu dílu

disp., Max. zásoba, Obj. hladina, Poznámka, Základní (ZU), Interní (IP), Kontrolní (TK) a Objednací.

H.9.1 Oddíly stránky detail dílu

Stránka detailu dílu se skládá z následujících oddílů:

Základní informace o dílu

Tyto upravuje VDP nebo zástupce VDP (tlačítko Upravit základní vlastnosti).

Poznámka k dílu

Upravuje VDP, zástupce VDP a zodpovědný nákupčí. Poznámka je specifická pro díl a projekt, ve kterém se díl vykytuje.

Vlastnosti

Upravuje VDP, zástupce VDP a zodpovědný nákupčí. Všechny vlastnosti kromě závodů jsou sdíleny mezi stejnými díly v různých projektech. Vybrané závody se váží ke specifickému projektu.

Příloha č.1 k Interní zprávě
Formulář KVM
Datum vydání 31.1.2013 - index 1

KMENOVÁ VĚTA MATERIÁLU

MAGNA EXTERIORS & INTERIORS

Kmenová věta materiálu
suroviny, barvy/laky, nakupované díly, obaly, náhradní díly - ne indirect, HAWA díly

Změna/Nový:

SAP číslo materiálu:	8 111 394	Závod:	0701 - Nymburk	Sklad:	
Třída ocenění:				Klíč disp:	
Projekt:	B0 MLMC 2 - P-10-02			Nominace:	Ne
Název:	Krytka vlečného oka TMC				
Cizojazyčný název:	Hole cover TMC				
Zákaznické číslo materiálu:	52129 - 0H020			Číslo výkr.:	52129-0H020
Zákazník:	TPCA				
MEPI klíč - Komoditní kód:					
Management jakosti:	ANO				
				MJ:	ks
Dodavatel:	11007236 - LUKOV Plast spol. s r.o.	Cena:		Měna:	EUR
				Počet MJ v ceně:	1
Hmotnost dílu (g):	25	Max. zásoba:		Obj. hladina:	
Profitové centrum:	CZAMPC02 - Nymburk TPCA				
D/TLD:	NE			Generační stav:	
Zařízení:	NE			Typ dodávek:	konsignační

Poznámka:

Základní (ZU):

Interní (IP):

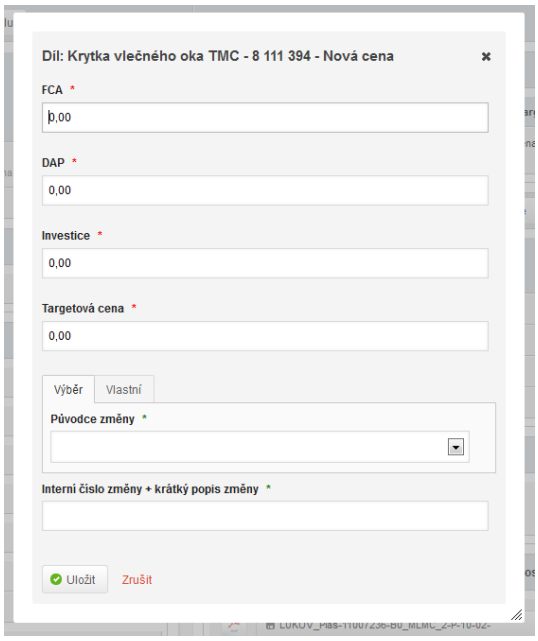
Kontrolní (TK):

Objednací:

Vystavil:	Petr Slavotínek	Dne:	04.05.2013
Správu kmen. dat provedl:		Dne:	

Obrázek H.18: Stránka tisku kmenové věty materiálu

Obrázek H.19: Úprava hodnoty na stránce tisku kmenové věty materiálu



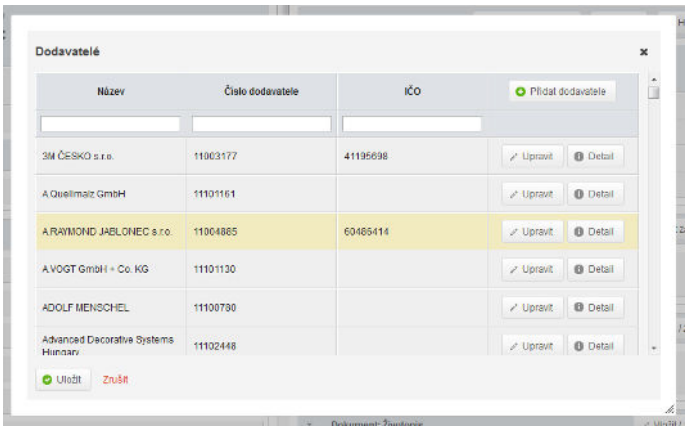
Obrázek H.20: Dialogové okno zadávání ceny

Cena

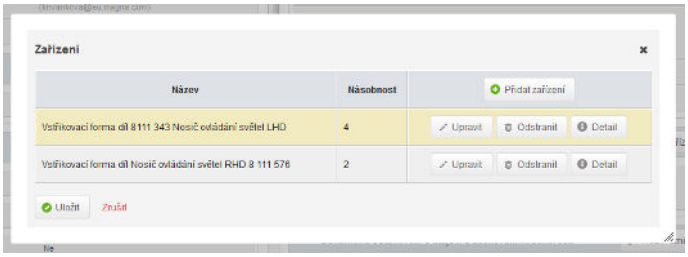
Nové ceny přidává VDP, zástupce VDP a zodpovědný nákupčí. Existující cenu může smazat pouze VDP. Tabulka zachycuje postupný vývoj cen, přičemž aktuální platná cena je vždy umístěna nejvýše. Při zadávání první ceny (viz obrázek H.20) není nutné vyplnit původce změny ani interní číslo změny a krátký popis. Tyto položky jsou povinné až při následovných změnách ceny.

Dodavatel

Dodavatele dílu může vybrat VDP, zástupce VDP a zodpovědný nákupčí. Tlačítkem Vybrat dodavatele je vyvolán seznam dodavatelů (viz obrázek H.21). Výběr dodavatele se provede kliknutím na požadovaného dodavatele (barva pozadí řádku se změní ze šedé na žlutou) a potvrzením tlačítkem Uložit. Přiřazeného dodavatele je možné odebrat opětovným kliknutím na něj (změna barvy zpět na šedou) a uložením. Dodavatele je možné vyhledávat podle názvu, SAP čísla a IČ psaním do příslušných textových polí v záhlaví tabulky. Je zde možné rovněž přidat nového dodavatele (tlačítko Přidat dodavatele), upravit stávajícího dodavatele (tlačítko Upravit) nebo zobrazit jeho detail (tlačítko Detail).



Obrázek H.21: Dialogové okno výběru dodavatele



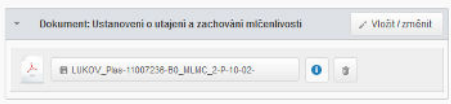
Obrázek H.22: Dialogové okno výběru zařízení

Zařízení

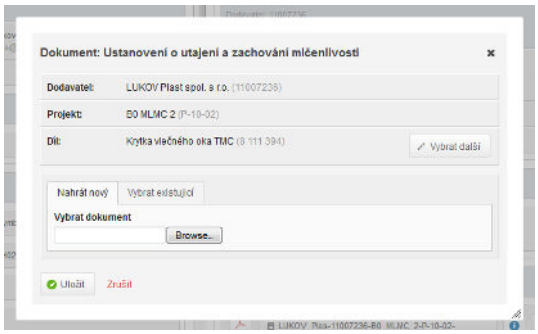
Zařízení může k dílu přiřazovat VDP, zástupce VDP a zodpovědný nákupčí, ale až poté, co je vybrán dodavatel dílu. Tlačítkem Přidat zařízení je vyvolán seznam zařízení od dodavatele dílu (viz obrázek H.22). V seznamu nejsou zobrazena zařízení již přiřazená k dílu. Výběr zařízení se provede kliknutím na požadované zařízení (barva pozadí řádku se změní ze šedé na žlutou) a potvrzením tlačítkem Uložit. Odebrání zařízení se provede ze stránky detailu dílu kliknutím na Tlačítko Odebrat u příslušného zařízení. V dialogovém okně pro výběr zařízení je rovněž možné vytvářet nová zařízení (tlačítko Přidat zařízení), upravovat stávající zařízení (tlačítko Upravit), odstraňovat je (tlačítko odstranit), čímž dojde k jejich úplnému vymazání ze systému a odebrání od všech dílů, a zobrazit jejich detail (tlačítko Detail).

Dokumenty

(platí rovněž pro dokumenty zobrazené u dodavatele)



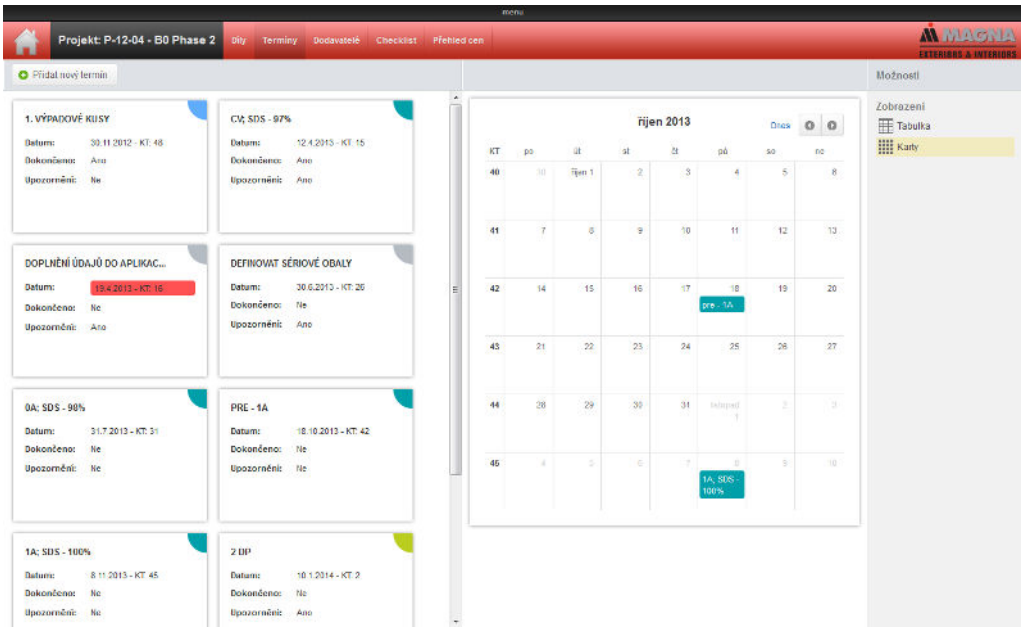
Obrázek H.23: Box nabízející stažení dokumentu



Obrázek H.24: Dialogové okno nahrávání dokumentu

Dokumenty vázající se k dílu jsou rozděleny do několika samostatných boxů dle typů dokumentů (viz obrázek H.23). Vkládat nové dokumenty a mazat stávající může VDP, zástupce VDP a zodpovědný nákupčí. Stažení dokumentu se provede kliknutím na tlačítko s ikonou diskety a názvem dokumentu. K odstranění dokumentu slouží tlačítko s ikonou odpadkového koše. Odstraněním dokumentu je pouze smazána vazba dokumentu na příslušné entity. V případě, že dojde ke smazání poslední vazby daného dokumentu, je dokument smazán rovněž z adresářové struktury projektové dokumentace. Pokud existuje více dokumentů daného typu, je zobrazen pouze poslední nahraný. Ostatní dokumenty lze zobrazit kliknutím na tlačítko Zobrazit další dokumenty.

Kliknutím na tlačítko Vložit / změnit je vyvoláno dialogové okno pro nahrání nového dokumentu (viz obrázek H.24). Pokud je dokument nahráván k dílu, pak je možné vybrat více dílu, ke kterým bude přiřazen. K tomu slouží tlačítko Vybrat další. Kliknutím na něj se zobrazí seznam dílů, které se vyskytují ve stejném projektu, a dodává je stejný dodavatel. Výběr dílů se provede jejich označením a potvrzením tlačítkem Uložit. Dále je možné buď nahrát nový dokument (záložka Nahrát nový) nebo vybrat existující (záložka vybrat existující). Pod záložkou Vybrat existující jsou zobrazeny dokumenty stejného typu přiřazené dílům ve stejném projektu od stejného dodavatele.



Obrázek H.25: Stránka přehledu termínů s kalendářem

H.10 Přehled termínů

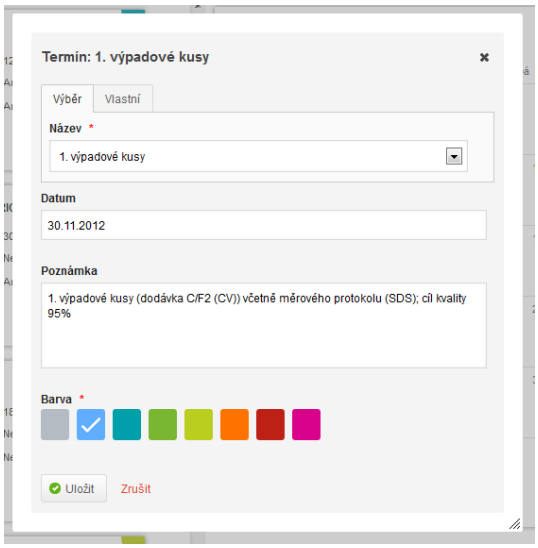
Na stránce Termíny (viz obrázek H.25) jsou zobrazeny všechny termíny přiřazené k projektu. Podobně jako u přehledu dílů lze přepínat mezi tabulkovým zobrazením a zobrazením karet. Na stránce je také zobrazen kalendář, ve kterém jsou termíny vyznačeny. V kalendáři lze přepínat mezi jednotlivými měsíci pomocí tlačítek s šipkami umístěnými v pravém horním rohu.

Správa termínů

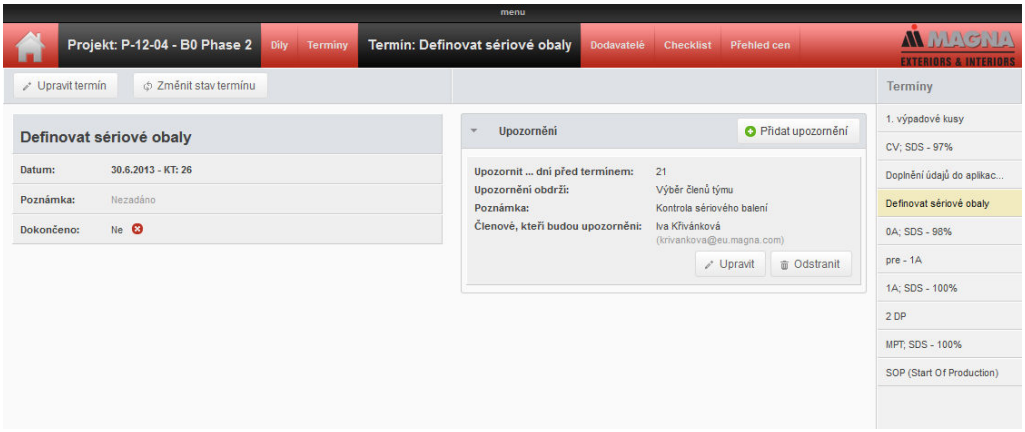
VDP a zástupce VDP mohou přidávat nové termín (tlačítko Přidat nový termín) a upravovat (tlačítko Upravit) a mazat (tlačítko Odstranit) stávající termíny. Při přidávání nebo úpravě termínu je povinné vyplnit pouze název, přičemž může být vybrán buď z předdefinovaných typů, nebo může být zadán vlastní (viz obrázek H.26). Datum i poznámku je možné doplnit později. Kliknutím na termín lze přejít na stránku detailu termínu.

H.11 Detail termínu

Na stránce detailu termínu (viz obrázek H.27) jsou zobrazeny všechny informace související s daným termínem včetně upozornění. Přehled všech



Obrázek H.26: Dialogové okno úpravy termínu

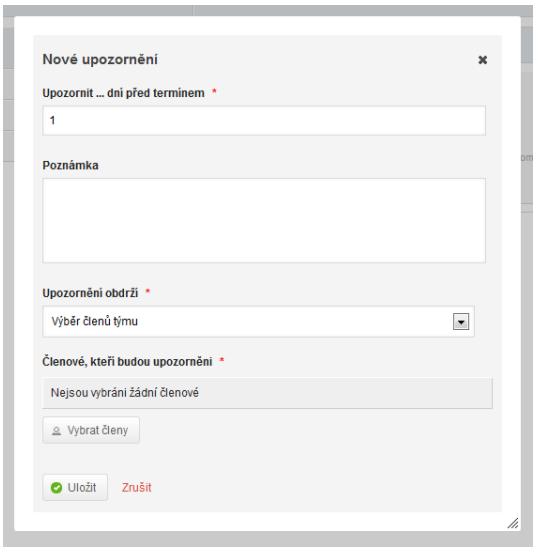


Obrázek H.27: Stránka detailu termínů

termínů v projektu umístěný v pravé části stránky lze použít pro rychlou navigaci mezi detaily termínů. VDP a zástupce VDP mohou upravovat vlastnosti termínu (tlačítko Upravit termín). Upravují se stejné vlastnosti jako na stránce přehledu termínů. Oba mohou také změnit stav termínu z Dokončeno na Nedokončeno a naopak (tlačítko Změnit stav termínu.

Upozornění

VDP a zástupce VDP mohou přidávat upozornění na blížící se termín (tlačítko Přidat upozornění). U upozornění je nutné vyplnit, kolik dní před



Obrázek H.28: Dialogové okno přidání nového upozornění

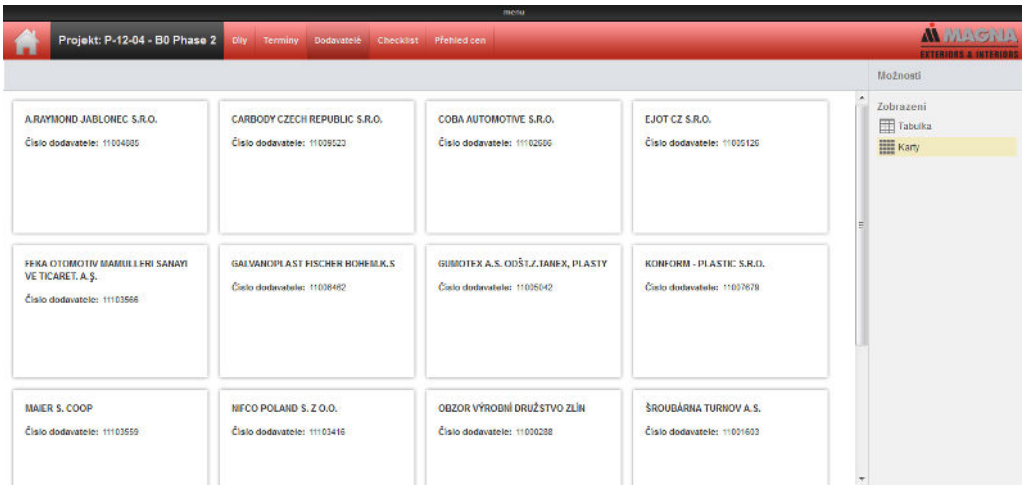
proběhnutím termínu má být upozornění zasláno (viz obrázek H.28). Dále je nutné vybrat, komu bude upozornění zasláno. Na výběr jsou tři možnosti: všichni, vedoucí a zástupce a výběr členů týmu, kdy si uživatel sám vybere uživatele. Existující upozornění je možné upravovat (tlačítko Upravit) a mazat (tlačítko Odstranit).

H.12 Přehled dodavatelů

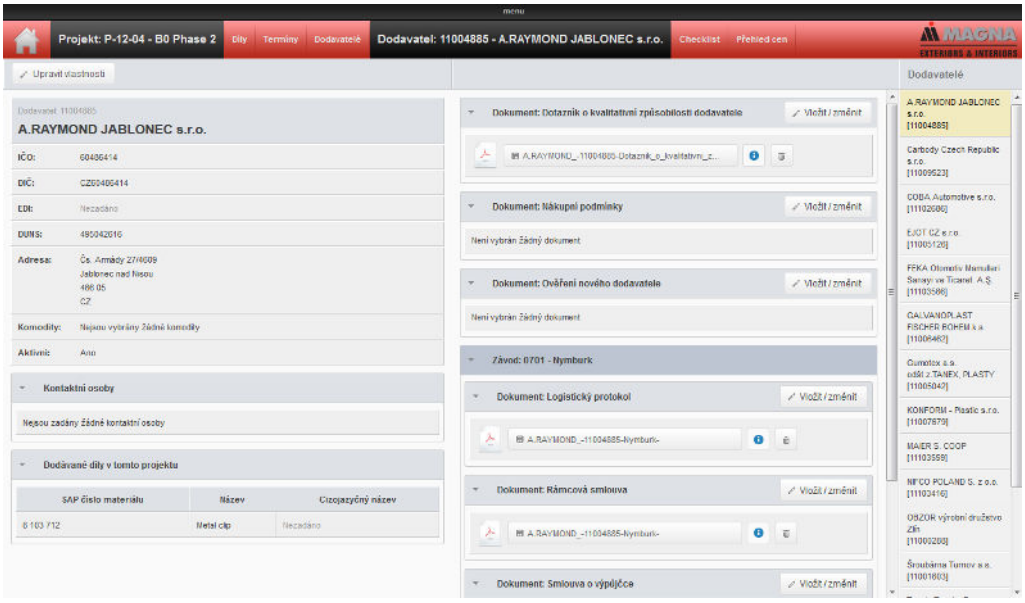
Na stránce Dodavatelé (viz obrázek H.29) jsou zobrazeny všichni dodavatelé dodávající nějaký díl v projektu. Opět lze přepínat mezi tabulkovým zobrazením a zobrazením karet. Kliknutím na dodavatele se zobrazí jeho detail.

H.13 Detail dodavatele

Na stránce detailu dodavatele (viz obrázek H.30) jsou zobrazeny všechny informace související s daným dodavatelem. Přehled všech dodavatelů dodávajících nějaký díl v projektu umístěný v pravé části stránky lze použít pro rychlou navigaci mezi detaily dodavatelů. Kterýkoliv uživatel může upravit vlastnosti dodavatele (tlačítko Upravit vlastnosti). Upravují se údaje o dodavateli, adresa, komodity a kontaktní osoby.



Obrázek H.29: Stránka přehledu dodavatelů



Obrázek H.30: Stránka detailu dodavatele

H.13.1 Oddíly stránky detail dodavatele

Stránka detailu dodavatele se skládá z následujících oddílů:

Základní informace o dodavateli

Jsou zde zobrazeny základní údaje o dodavateli, adresa a komodity.

Kontaktní osoby

Jedná se o seznam kontaktních osob dodavatele. Upravují se spolu se základními vlastnostmi.

Dodávané díly v tomto projektu

Představuje seznam dílů, které dodavatel dodává v daném projektu. Kliknutím na díl lze přejít na jeho detail.

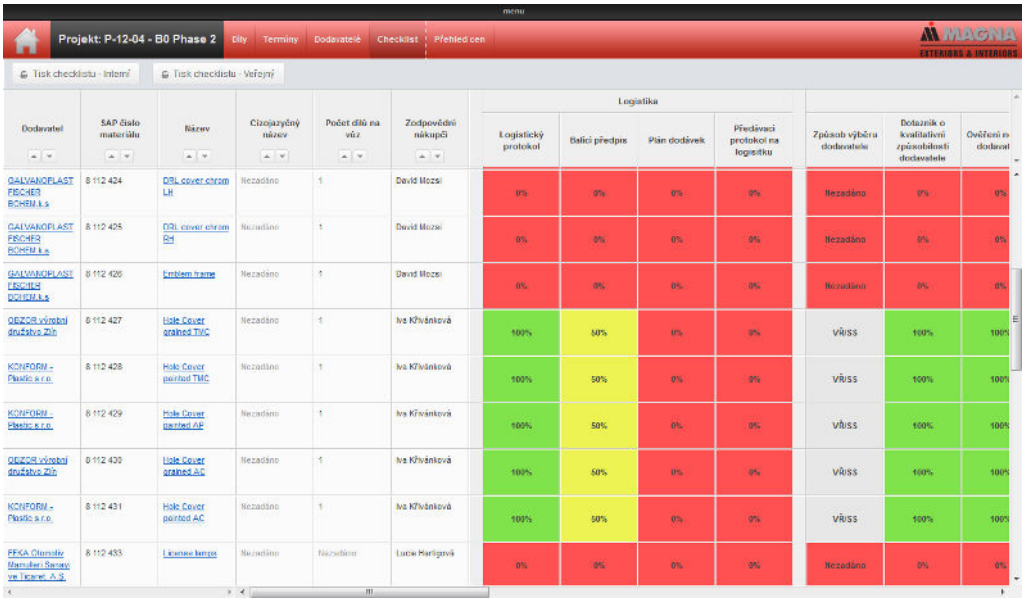
Dokumenty

Dokumenty jsou podobně jako u dílu rozděleny do samostatných boxů podle typu. Některé typy dokumentů jsou zároveň seskupeny podle příslušnosti k jednotlivým závodům. Jak dokumenty spravovat je popsáno v oddílu Detail dílu.

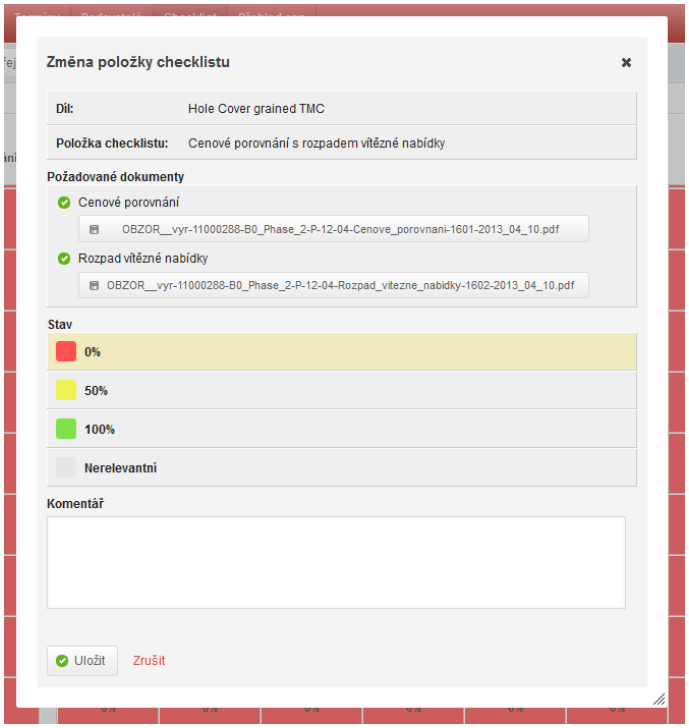
H.14 Checklist

Na stránce Checklist (viz obrázek H.31) je zobrazen projektový Checklist. Lze zobrazit Checklist v podobě přizpůsobené pro tisk, a to buď pro osobní použití bez záhlaví a zápatí (tlačítko Tisk checklistu – Interní), nebo pro účely vystavení Checklistu na projektovém serveru, jehož vzhled odpovídá interním nařízením (tlačítko Tisk checklistu - Veřejný).

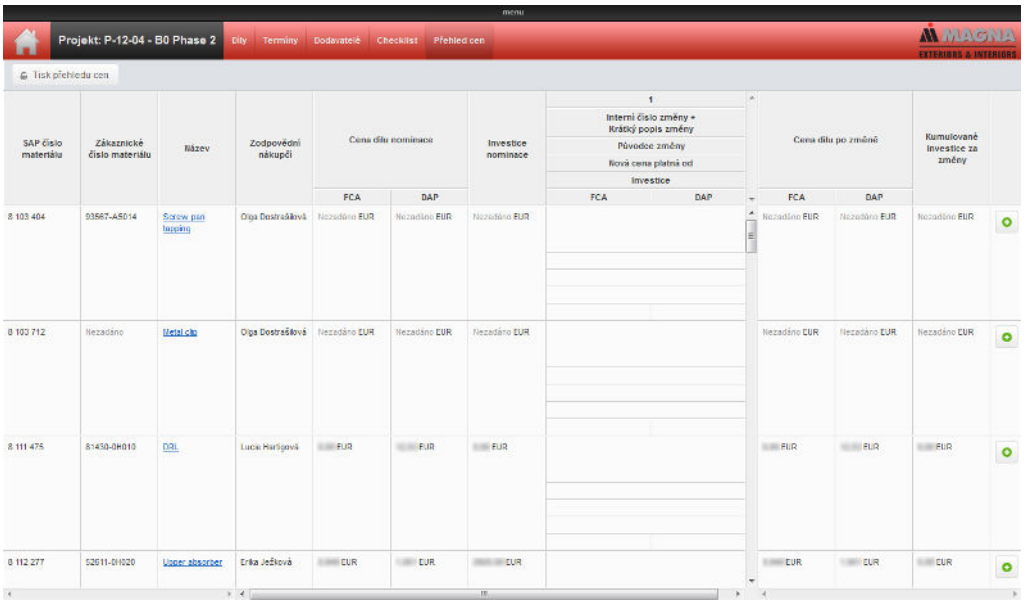
Hodnotu položky Checklistu (jednu buňku v tabulce) může upravit VDP, zástupce VDP a nákupčí zodpovědný za díl, ke kterému se položka vztahuje. Aby bylo možné položku upravit, musí být k danému dílu přiřazen dodavatel. Kliknutím na položku se vyvolá dialogové okno pro změnu hodnoty položky (viz obrázek H.32). Pokud je položka závislá na nějakém dokumentu, jsou zde tyto dokumenty nabídnuty ke stažení. Případně je ukázáno, že dokumenty dosud nebyly vloženy. Pro tyto položky zároveň platí, že hodnota 100 % může být zadána až poté, co byly vloženy všechny požadované dokumenty. Hodnota Nerelevantní vyžaduje vyplnění komentáře, neboli zadání důvodu, proč položka není relevantní.



Obrázek H.31: Stránka Checklistu



Obrázek H.32: Dialogové okno změny hodnoty položky Checklistu



Obrázek H.33: Stránka přehledu cen

H.15 Přehled cen

Na stránce Přehled cen (viz obrázek H.33) je zobrazen seznam všech dílů a vývoj jejich cen. Ve sloupcích Cena dílu nominace a Investice nominace je zobrazena první vložená cena k dílu. V dalších sloupcích jsou následně zobrazeny změny cen. V posledním sloupci Cena dílu po změně je vidět aktuální cena dílu a ve sloupci Kumulované investice za změny je součet investic ze všech změn. Kliknutím na tlačítko Tisk přehledu cen se zobrazí přehled cen v podobě přizpůsobené pro tisk.

Změnu ceny dílu (neboli zadání nové ceny) je možné provést kliknutím na tlačítko se zeleným plus, což vyvolá dialogové okno pro zadání ceny. Právo na změnu ceny má VDP, zástupce VDP a zodpovědný nákupčí.

H.16 Dokumentace

V tabulce H.1 je uveden seznam všech dokumentů a na jaké stránce lze dokumenty prohlížet a spravovat.

Tabulka H.1: Umístění dokumentů

Název typu dokumentu	**Obecný detail dodavatele** (všechny závody)	**Detail dodavatele v projektu** (jen závody přiřazené v projektu)	**Detail dílu**
Dotazník o kvalitativní způsobilosti dodavatele	×	×	
Nákupní podmínky	×	×	
Ověření nového dodavatele	×	×	
Logistický protokol	×	×	
Rámcová smlouva	×	×	
Příloha rámcové smlouvy		×	
Smlouva o výpůjčce nástrojů		×	
Ustanovení o utajení a zachování mlčenlivosti			×
Životopis nakupovaného dílu			×
Cenové porovnání			×
Rozpad vítězné nabídky			×
Katalog požadavků na nakupovaný díl			×
Dohoda o zajištění kvality			×
Plán zkoušek			×
Nominační dopis			×
Balící předpis			×
První vzorkování			×
Předávací protokol na logistiku			×

Příloha I

Dotazník

1. **Používáte aplikaci?** Pokud ne, uveďte důvod a na další otázky neodpovídejte.

Ano	Ne, ale brzy začnu	Ne a ani nehodlám	Ne a dosud jsem se nerozhodl/a jestli začnu

Důvod: .....

.....

2. **Jak často aplikaci používáte?**

Neustále	Denně	Několikrát do týdne	Jednou týdně	Jednou za měsíc

3. **Ulehčuje Vám aplikace práci?** Pokud ne, uveďte důvod.

Určitě ano	Spíše ano	Nevím, zatím nedokážu posoudit	Spíše ne	Určitě ne	Určitě ne, práce s ní je zdlouhavá

Důvod: .....

.....

4. **Využíváte aplikaci jako poskytovatele informací namísto dosavadních prostředků?** (papírové dokumenty, elektronické dokumenty na projektovém serveru, vlastní dokumenty a poznámky) Pokud ne, uveďte důvod.

Ano, neustále	Ano, občas	Ne, nezkoušel/a jsem to	Ne, nemůžu tam nic najít	Ne, nejsou tam informace, které potřebuji

Ne, jiný důvod: .....

.....

5. **Jak hodnotíte uživatelské prostředí aplikace?**

Líbí se mi a je přehledné	Líbí se mi, ale je nepřehledné	Nelíbí se mi, ale je praktické	Nelíbí se mi a je nepřehledné	Je strašné, nedá se s ním pracovat

6. **Jak se v aplikaci orientujete?**

Rychle a bez zaváhání	Pomaleji, občas něco hledám	Pomalu, pořád něco hledám	Pomalu, musím hodně klikat, než se někam dostanu	Pomalu, funkce a odkazy jsou nesmyslně umístěné

7. **Považujete novou aplikaci za přínosnou?**

Určitě ano	Spíše ano, ale představoval/a jsem si něco jiného	Zatím nevím, uvidí se časem	Ne, je zbytečná	Ne, nepracuje se mi s ní dobře

Vaše připomínky: .....

.....

Příloha J

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `app` ..... soubory potřebné pro spuštění aplikace
  - `db` ..... SQL skripty pro vytvoření a naplnění databáze
  - `deploy` ..... adresář se zkompilovanou podobou aplikace
- `src`
  - `diagrams` ..... diagramy pro aplikaci Enterprise Architect
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF