Sem vložte zadání Vaší práce.

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

Bakalářská práce

Diagnostika dopravního sálu

Tomáš Hanousek

Vedoucí práce: Ing. Radek Dobiáš, Ph.D., MBA.

11. května 2015

Poděkování

Chtěl bych poděkovat vedoucímu mé práce Ing. Radku Dobiášovi, Ph.D., MBA. za rady, které mi poskytl při psaní této bakalářské práce. Dále bych chtěl poděkovat své rodině za podporu po celou dobu mého studia a kantorům Dopravního sálu Fakulty dopravní.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Hanousek, Tomáš. *Diagnostika dopravního sálu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato bakalářská práce se zabývá projektem Diagnostiky dopravního sálu. V první části je analyzován Dopravní sál Fakulty dopravní a komerční diagnostický program LDS-3, v druhé části se věnuji technikám používaných v projektové fázi. Výsledkem této práce je vytvořená projektová dokumentace a navržená softwarová architektura diagnostického programu. V závěru práce jsou pak navrženy funkční a akceptační testy.

Klíčová slova Dopravní sál Fakulty dopravní, softwarová architektura, projekt, projektová dokumentace

Abstract

This bachelor thesis deals with project called Diagnostic of Traffic Hall. In the first section the Traffic hall of Faculty of Transportation Sciences and commercial diagnostic program LDS-3 are analysed, in the second section I devote to the techniques used in project phase. The result of this work is created project documentation and designed software architecture of diagnostic program. In the conclusion of this work there are designed functional and acceptance tests.

Keywords Laboratory of Transport Technics, software architecture, project, project documentation

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem této bakalářské práce je vytvoření diagnostického programu pro Dopravní sál Fakulty dopravní ČVUT. Diagnostický program má za úkol monitorovat dění na pracovišti Dopravního sálu a umět prezentovat data aktuální i historická. Celá tato bakalářská práce je vedena jako projekt pro zákazníka a jedním z výstupů je i projektová dokumentace.

V první části se věnuji podrobné analýze Dopravního sálu a požadavkům, které zákazník stanovil na program. Je zde rozebrán i současný stav a analýza uživatelů. V rámci analýzy jsou zpracovány i základní wireframy aplikace. Tato část se také zábývá komerčním diagnostickým programem LDS-3, který se ve skutečnosti využívá.

Druhá část je zaměřena na teorii projektového řízení. Zde se věnuji projektové fázi a praktikám, které se v praxi používají. Celá projektová fáze je lehce uvozena i do kontextu, ve kterém se nachází.

Praktická část začíná ve třetí fázi projektovou dokumentací. Zde jsou popsány postupy, které jsem použil a proč jsem je tak volil. Jsou zde stanovena rizika projektu, náklady a harmonogram.

Ve čtvrté fázi se věnuji návrhu architektury diagnostického programu. Je zde popsána funkčnost jednotlivých komponent a rozdělení na dvě jednotky — server a klient.

Poslední část se zaměřuje na testování aplikace z pohledu zákazníka. Je zde uvedena teorie k testování a na závěr jsou stanoveny testovací scénáře pro naimplementovaný diagnostický program.

Kapitola 1

Analýza

První kapitola se zaměřuje na definici požadavků zákazníka a problémové domény.

Analýza je jedna z nejdůležitějších částí celého projektu. Chybně definované požadavky nebo špatné pochopení požadavků zákazníka vyústí v aplikaci s jinou funkčností, než zákazník požadoval. V důsledku toho se projekt pravděpoodbně prodraží, v horším případě bude neúspěšný.

Se zástupci Dopravního sálu Fakulty dopravní (dále DSFD) jsem se v rámci tří schůzek seznámil s provozem Dopravního sálu a výstupech, které mají z tohoto projektu vzejít. Zástupci DSFD mi dále odpověděli na mé otázky, které jsem doplnil o [6]. V rámci těchto schůzek mi byl rovněž představen Dopravní sál jako celek a některá zařízení, která se zde nacházejí.

1.1 Současný stav a problémy

V DSFD je již implementován protokol, který sbírá data od jednotlivých připojených zařízení v Dopravním sále. Tato jsou skrze sběrnici odesílána na server. Protokol není standardní, má svou vlastní strukturu.

V Dopravním sále je na serveru dostupná knihovna funkcí, které dokáží poskytnout data odesílaná zařízeními. Tato data ovšem nejsou dále zpracovávána a celý diagnostický program v této fázi aktuálně končí. Zmíněná knihovna je napsaná v programovacím jazyce C#.

Hlavním důvodem, proč je potřeba do Dopravního sálu nasadit diagnostický program, je aktuální nemožnost detekovat chyby, které nastaly. Pomocí výše zmíněné sběrnice neproudí data jen směrem od zařízení, ale i směrem k nim. Dají se tak například simulovat poruchy typu prasklé žárovky na návěstidle a podobně. Celý Dopravní sál je simulací reálného provozu, kde si studenti FD ČVUT mohou vyzkoušet řízení provozu na modelové železnici. Určité situace na modelové železnici jsou realizovány programově a pro ověření, že tento scénář funguje správně, je potřeba vytvořit Diagnostický program.

1.2 Data

Tato sekce se zabývá vstupními daty a formou výstupních dat pro klientskou aplikaci.

1.2.1 Vstup

Data jsou ze zařízení v Dopravním sále přístupná skrze již implmenetované funkce a jsou dostupná v binárním kódu. Všechna data mají jasně definovanou strukturu, díky které je možné je přiřadit ke konkrétní kontrolní signalizaci zařízení. Data taktéž nesou informaci, v jakém stavu se kontrolní signalizace nachází.

Na server putují data z obecně více zařízení každých 100 ms a jejich celková velikost je přibližně 1 KB.

1.2.2 Výstup

Výstupní data mají být v grafické podobě — například fotografie, kde jsou jednotlivé kontrolní signalizace zobrazovány různými barvami. Nemá tedy jít o tabulkový výpis stavů.

Pro přenos dat mezi serverem a klientskými stanicemi je v Dopravním sále připraven protokol TCP/IP.

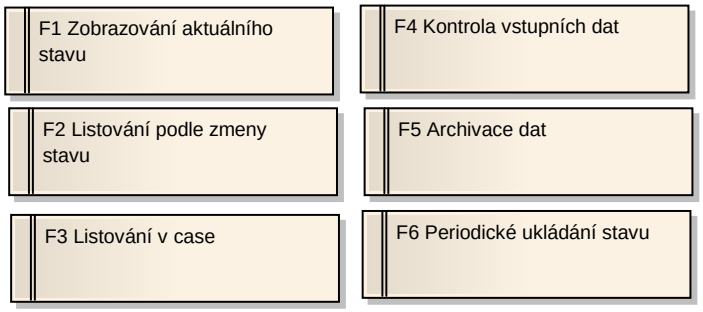
1.3 Analýza požadavků na systém

Po zjištění aktuálního stavu v Dopravním sále, jsem se mohl pustit do zjištění požadavků, které má zákazník na systém. Požadavky jsou rozděleny na funkční a nefunkční a shrnuty na obrázcích 1.1 a 1.2 vytvořené v programu [7].

1.3.1 Funkční požadavky

F1 Zobrazování aktuálního stavu Každé zařízení má v konkrétním čase určité stavy kontrolních signalizací. Zákazník požaduje, aby mohl v reálném čase přes klientskou aplikaci sledovat vývoj všech stavů kontrolních signalizací konkrétního zařízení. Perioda aktualizace stavu jím byla stanovena na jednu sekundu.

F2 Listování podle změny stavů Kontrolní signalizace může v čase měnit svůj stav. Pro lehčí zpětné vyhledávání doby, kdy se stav změnil, bude aplikace umět listovat v historii změn. Součástí bude i nastavení počátečního času, od kterého bude možné sledovat změny stavu kontrolní signalizace. S každou takovou změnou bude v aplikaci přístupný i čas, kdy ke změně došlo.



Obrázek 1.1: Funkční požadavky

F3 Listování v čase Podobně jako u Listování podle změny stavů bude fungovat tento funkční požadavek. Hlavní rozdíl spočívá v tom, že krokování bude závislé od změny jakéhokoliv stavu kontrolní signalizace. Taková změna bude z důvodu lepší přehlednosti zvýrazněna od ostatních.

F4 Kontrola vstupních dat Vstupní data mají určitá pravidla, podle kterých lze jednoznačně zjistit, zda jsou správná. Dají se tak například určit chybná data, která je nežádoucí zapisovat do databáze.

F5 Archivace dat Pro případ selhání databázového stroje je nutné získaná data zálohovat. Perioda zálohování a umístění exportu z databáze bude parametrizováno pomocí konfiguračního souboru.

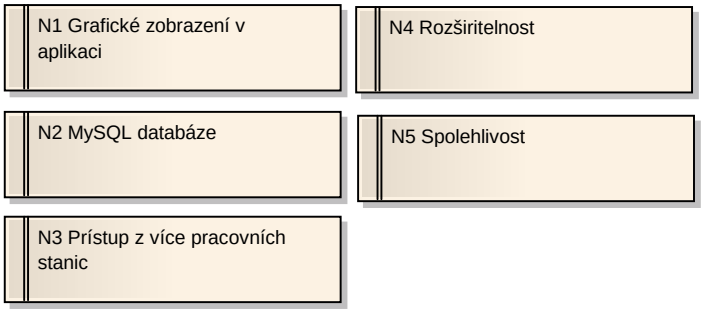
F6 Periodické ukládání stavů Zákazníkovo přání je ukládat stavy všech kontrolních signalizací periodicky, i když se stav nezměnil. Perioda ukládání bude nastavitelná pomoci konfiguračního souboru.

1.3.2 Nefunkční požadavky

N1 Grafické zobrazení v aplikaci Aplikace bude zobrazovat data pomocí grafické reprezentace. Každý modul bude mít svou vlastní fotografii nebo jednoznačně rozeznatelnou kresbu a každá kontrolní signalizace svou vlastní množinu stavů.

N2 MySQL databáze Na serveru je nainstalována MySQL databáze, do které budou ukládány získané informace ze zařízení v Dopravním sále.

N3 Přístup z více pracovních stanic K datům získaným ze zařízení musí být přístup z více stanic najednou. Na uživatelských stanicích je operační systém Windows.



Obrázek 1.2: Nefunkční požadavky

N4 Rozšiřitelnost Do budoucna se předpokládá, že se Dopravní sál může rozšířit o další zařízení. Z tohoto důvodu je kladen důraz na rozšiřitelnost vyvíjeného softwaru o další moduly zobrazování.

N5 Spolehlivost Software má klást důraz na spolehlivost zobrazovaných dat. Důležité je ukládat pouze 100% správná data.

1.4 Navržené funkčnosti

Jediná věc, která na takto definovaném systému chybí, je logování chyb, které je doporučené i v [8]. Tuto funkcionalitu jsem tedy doporučil k implementaci a zákazník ji schválil.

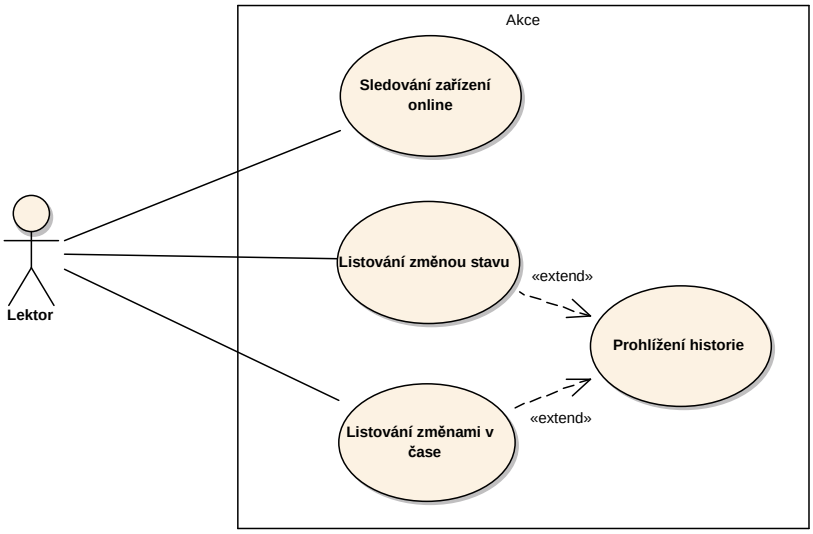
Otázkou zůstává, zda tyto chyby ukládat do souboru nebo do databáze. Podle mého uvážení má databáze nesporné výhody především v:

- atomicitě dat,
- možnosti jednoduchého filtrování dat pomocí SQL dotazů,
- filtrování podle času a
- jednodušším zápisu pro vícevláknové aplikace.

Podobně se dají napsat skripty i pro soubor neboť by podobnou funkcionalitu taktéž zvládly. Vzhledem k tomu, že se však s daty bude dále aktivně pracovat, je zápis chyb do databáze schůdnější cestou.

I jak se píše v [9]: „Ukládání logů do databáze není hrozný nápad, ale ukládat je do stejné databáze jako produkční data ano."

Po porovnání jsem se rozhodl ukládat chybová hlášení do databáze zejména kvůli snadnějšímu vyhledávání potřebných dat. Další nespornou výhodou je



Obrázek 1.3: Případy užití

snadnější zápis dat — v případě ukládání chyb do souboru je vzhledem k vícevláknovému řešení třeba zajistit synchronizace, aby nedocházelo k přepisování nebo ztrátě dat.

1.5 Uživatelé systému a zabezpečení

Uživateli Diagnostického programu budou výhradně lektoři v Dopravním sále, kteří budou mít možnosti zobrazené na případech užití na obrázku 1.3. Jelikož jedním z požadavků je i přístup k diagnostickému programu z více uživatelských stanic, je potřeba přenášet data po síti.

Přístup k aplikaci bude umožněn na kokrétní pracovní stanici, kde bude po přihlášení na administrátorský účet předinstalovaná klientská aplikace. K administrátorskému účtu mají přístupové údaje právě jen lektoři v Dopravním sále.

Přenášená data ze serveru po síti nejsou citlivá. Jsou přenášena výhradně v rámci intranetu Dopravního sálu a navrhovaný systém v Dopravním sále nic nenastavuje. Server má zákazník pod vlastní kontrolou. Proti případnému fyzickému poškození třetími stranami nebo přírodními živly ho chrání sám.

1.6 Wireframy

Wireframy jsou jedním z prostředků, jak dle požadavků zákazníka definovat vzhled a funkčnost vyvíjené aplikace a na druhé straně programátorům vizualizovat myšlenky a usnadnit pochopení návrhu.

Po vydefinování všech požadavků a zjištění maxima informací od zákazníka, jsem navrhnul v rámci analýzy i wireframy. Jedná se o tři obrazovky, jejichž konkrétní funkčnost je vždy popsána.

Pro vytvoření wireframů byl použit program [10].

1.6.1 Online vizualizace

První wireframe, vyobrazený na obrázku B.1, ukazuje návrh zobrazení při online módu, tedy při zobrazování aktuálního stavu konkrétního modulu v Dopravním sále.

Následující seznam popisuje jednotlivé body očíslované na wireframu:

1. Tímto tlačítkem bude moci uživatel přepínat mezi tzv. online módem a offline módem. Zobrazení na tlačítku bude signalizovat v jakém konkrétním stavu se uživatelská aplikace nachází.

2. Zde se bude nacházet dropdown menu. To částečně pokryje požadavek na rozšiřitelnost software, neboť později nebude potřeba upravovat zobrazovací vrstvu aplikace (jako například při výběru pomocí radio buttonů). Pro co nejjednodušší vyhledání požadoveného modulu budou moduly seřazené podle abecedy.

3. Na tomto místě se bude zobrazovat čas, ke kterému jsou uváděná data aktuální. Čas je zde uveden zejména kvůli možnosti selhání sítě nebo dalším problémům. Vzhledem k požadavkům klienta bude stačit přesnost na sekundy bez uvedení datumu, neboť je zřejmé, že se jedná o čas stejného dne.

4. V tomto místě bude zobrazen modul a jednotlivé vizualizované stavy.

1.6.2 Offline vizualizace — signál

Další wireframe B.2 ukazuje návrh uživatelského prostředí při offline vizualizaci. Bude možné zpětně procházet daty z historie vybrané kontrolní signalizace.

V následujících bodech popisuji jednotlivé funkčnosti:

1. Podobně jako u online vizualizace zde bude tlačítko definující v jakém módu se aplikace nachází. V tomto případě bude na tlačítku vyzobrazeno Offline. Jelikož offline mód nabízí dvě možnosti zobrazování, respektive vyhledávání, bude zde i další tlačítko, které tento výběr umožňuje a zobrazuje, tedy jaká možnost je momentálně vybrána.

2. Toto menu má stejnou funkčnost jako v případě online módu.

3. Zde bude možnost určit konkrétní kontrolní signalizaci na dříve vybraném modulu. Pod možností výběru kontrolní signalizace se nachází posuvník, díky kterému se lze v historii vpřed i vzad.

4. V tomto místě si lze vybrat konkrétní datum, od něhož se bude možné vydávat v historii vpřed i vzad. Součástí bude i definice data. Tato skutečnost umožní, aby se v historii dalo pohybovat dále než jeden den zpět.

5. Stejně jako při online módu se v této části bude zobrazovat modul se všemi kontroloními signalizacemi.

1.6.3 Offline vizualizace — modul

Poslední obrazovka B.3 zobrazuje uživatelské rozhraní pří sledování historie modulu. V rámci této obrazovky bude možné sledovat jakoukoliv změnu stavu na konkrétním modulu.

1. Jak už jsem zmiňoval výše, offline mód má dvě možnosti zobrazování. První je výše zmíněná, tou druhou je jakékoliv zobrazování změny v čase. Výběr tohoto módu bude opět symbolizovat tlačítko s příslušným názvem.

3. Podobně jako u předchozího módu, i zde bude možné vybrat datum a čas, ke kterému se má systém spouštět. Pomocí posuvníků bude možné pohybovat se vpřed a vzad k nejbližším změnám v daném čase.

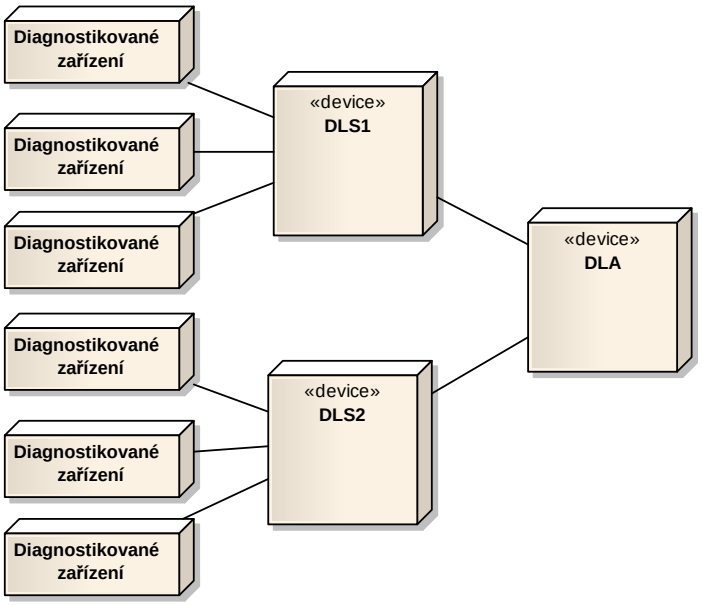
Další nepopsané body mají v principu podobnou nebo stejnou funkčnost jako výše uvedené.

1.7 Existující řešení

Jelikož se jedná o velmi specifický software, od vedoucího práce jsem obdržel Návod pro obsluhu a údržbu lokálního diagnostického systému LDS-3 [11], ze kterého tato část mé bakalářské práce vychází. Jedná se o komplexní systém vytvořený AŽD Praha, s.r.o. určený pro diagnostiku železnice dodávaný spolu hardwarovým vybavením.

1.7.1 Struktura LDS-3

Celý komplex lze rozdělit do dvou základních částí — DLS a DLA. Nedílnou součástí mimo systém samotný jsou diagnostikované jednotky.



Obrázek 1.4: Propojení DLA, DLS a diagnostikovaných zařízení

1.7.1.1 DLS

DLS je průmyslový počítač, který získává z připojených diagnostikovaných jednotek informace, které dále zpracovává a uchovává v datových souborech. Na tomto počítači běží operační systém Linux Fedora.

Systém má rozsáhlou adresářovou strukturu s programy pro velkou škálu funkcí. Nejzajímavější z pohledu projektu diagnostiky pro Dopravní sál jsou programy pro analýzu a archivaci, které ovšem nejsou v návodu podrobněji popsány.

Podle uvedených dat by si systém s kapacitou disku 250 GB měl vystačit na 2–5 let v závislosti na provozu na sledovaném úseku. Systém je také vybaven programem pro vymazání nejstaších dat, která udělají dostatek prostoru na disku pro data novější.

1.7.1.2 DLA

DLA je systém lokálního přístupu k datům. Na rozdíl od DLS je na takovém počítači nainstalován systém Microsoft Windows XP nebo Windows 7. DLA je schopen připojovat se k více DLS, jak je znázorněno na obrázku 1.4.

Tento systém se může dále rozložit na celkem pět dalších podprogramů, kde každý z nich dokáže podat informace ze železnice jiným způsobem nebo pro jiné diagnstikované jednotky. Jak je v tomto návodu uvedeno, plné funkčnosti se docílí spuštěním všech těchto podprogramů.

V nabídce režimů, které DLA podporuje, jsou následující možnosti:

- online — zobrazování automaticky aktualizovaných dat z diagnostikovaných zařízení,
- procházení archivu — DLA stahuje data z DLS vzhledem k datu a času stanoveného uživatelem,
- procházení lokálního archivu — uživatel prochází data uložená v lokální paměti DLA. Balíček s daty, definován datem a časem od — do, je potřeba předem stáhnout z DLS. K takovému balíčku pak lze přiřadit textový popisek.

V jakém režimu se DLA aktuálně nachází, jednoznačně určuje stavový řádek.

DLA je oproti DLS funkčně mnohem lépe vybaven a poskytuje mnoho funkcí, například:

- režim výběru DLS — v případě, že jsou systémy DLA a DLS propojeny, může se DLA připojit k vybranému DLS,
- podpora tisku a uložení do souboru aktuálního zobrazení,
- skoky na konec a začátek staženého lokálního archivu,
- pevně dané klávesové zkratky,
- automatické přehrávání,
- zobrazování grafů s časovými průběhy napětí,
- měření výkonu přestavníků,
- nastavení GSM modulu — stanovení příjemců a odesílání varovných SMS zpráv.

1.7.2 Bezpečnost

Obě části nabízí různé úrovně zabezpečení. V částech, které se problematice zabezpečení věnují, se autoři zřejmě dopustili nekonzistence názvosloví. V případě DLS je uvedeno, že se jedná o uživatelské účty, u DLA jsou uvedeny úrovně oprávnění. Z náhledu obrazovky u DLA ovšem plyne, že jsou zde zavedeny uživatelské účty a nikoliv oprávnění. Protože není v mých možnostech si systém vyzkoušet, budu vycházet z toho, že oba systémy využívají ke svému zabezečení uživatelské účty, kterým je přiřazeno určité oprávnění.

U popisu zabezpečení není rozebrána problematika šifrování dat putujících po síti. Vzhledem k tomu, že tato příručka je určena uživatelům systému k pochopení toho, jak vůbec se systémem zacházet, není šifrování dat problematika, která je k tomuto potřebná.

1.7.2.1 DLS

V rámci DLS jsou vytvořeni uživatelé s následujícími oprávněními:

- root — veškerá přístupová práva,
- azd, servis — možnost modifikace některých souborů ve složce „azd/",
- kolej — prohlížení archivních dat, bez hesla,
- kolej1-kolej4 — prohlížení aktuálních dat pro příslušnou kolej, bez hesla.

1.7.2.2 DLA

DLA má celkem tři uživatele:

- údržba — prohlížení online a archivních dat, bez hesla,
- správce — nastavování systému pro místní správce,
- administrátor — stejná oprávnění jako správce, určeno pro zaměstnance AŽD Praha, s.r.o., DSE.

1.7.2.3 Upgrade a modifikace

Do softwarového a hardwarového vybavení DLS i DLA jsou oprávněni zasahovat pouze zaměstnanci servisu AŽD Praha, s.r.o., DSE. Po dobu záruky je k instalaci potřeba schválení odpovědným zaměstnancem AŽD Praha, s.r.o., DSE.

1.7.3 Průnik s požadavky zákazníka

V systému LDS-3 je evidentně vidět množina funkcí, které zákazník požaduje. Je ovšem zřejmé, že se jedná o komplexní systém, kde by většina jeho funkcí nenašla v DSFD uplatnění.

Ve své podstatě má zákazník připraven hardware DLS, což je server v sále, na který bude nahrána aplikace pro sběr a odesílání dat klientovi a narozdíl od skutečného DLS, tato poběží na serveru s OS Microsoft Windows.

Pomocí již vytvořené knihovny funkcí, která poskytuje data ze zařízení v dopravním sále, budou k tomuto serveru připojeny tzv. diagnostikované jednotky. Kompatibilitu bude zajišťovat server, který dokáže přijatá data rozeznat a vhodně ukládat do databáze.

Klientské stanice v dopravním sále vybavené OS Microsoft Windows 7, symbolizují DLA. Zde ale tento systém nebude mít stejnou funkčnost, jako skutečné DLA, ale bude se prakticky zabývat jen režimy zobrazení dat — tedy online a procházení archivních dat na serveru. Další funkčnosti, kterými je DLA vybaven, by nenašly v DSFD využití (například odesílání stavů SMS zprávou), neboť diagnostikované jednotky neposkytují potřebná data.

Zabezpečení přístupu k aplikaci nebude rozděleno jako v případě LDS-3, ale bude omezeno jen na administrátorský účet, ke kterému znají přístupová práva jen lektoři v dopravním sále. Pokud bych měl přirovnat úroveň zabezpečení LDS-3, pak se v případě DLA jedná o úroveň „údržba", jelikož vzdáleně s daty nebude možné manipulovat nebo na serveru skrze tuto aplikaci provádět nějaké modifikace nastavení.

Kapitola 2

Projekt

Projekt je koordinované usilí ohraničené zdroji, náklady a časem, které je zaměřeno na vytvoření unikátní služby nebo produktu [12].

Každý projekt by měl být SMART [13], což je zkratka, která v sobě skrývá slova

- specific (specifický) — znát konkrétní cíl projektu,
- measurable (měřitelný) — díky tomuto kritériu lze sledovat progres a naplnění cílů projektu,
- acceptable (přijatelný) — všechny zúčastněné strany musí souhlasit se stanovenými cíly,
- realistic (realizovatelný) — stanovení cílů, které jsou za aktuálních podmínek splnitelné,
- time bounded (časově ohraničený) — stanovení časového harmonogramu, především začátku a konce.

Projekt se snaží naplnit tzv. trojimperativ [13]. Jde o vztah mezi časem, cílem a finančními náklady vynaloženými na projekt. Ideální projekt je dokončen v nejkratším možném čase s maximálním cílem za minimální náklady. V praxi lze ale ideálního projektu jen velmi těžko dosáhnout. Proto je potřeba vytořit kontrolní strategii [12], která může být zaměřena na jeden konkrétní atribut trojimperativu. Další možností je vytvořit vyváženou strategii, což znamená vytoření kompromisu mezi jednotlivými atributy.

S projektem je také spjat pojem projektové řízení. Jedná se o soubor rad, jak řídit projekt. Aby se projektové řízení dalo použít obecně, nejedná se o přesné definice, ale spíše o „filozofii přístupu k řešení dané problematiky" [5].

Ne každou činnost je vhodné vést jako projekt a využívat projektového řízení. Například pro krizové situace, opakující se činnosti nebo jednoduché

akce není vhodné použít projektové řízení, ale zvolit nějaký jiný nástroj pro vedení příslušné činnosti [5].

2.1 Životní cyklus projektu

Životní cyklus projektu lze rozdělit do tří částí [5]. První část, tzv. předprojektová fáze, poskytuje odpověď na otázku, zda projekt realizovat. Součástí jsou zpravidla dva dokumenty. Prvním je studie příležitosti, která dává doporučení k realizaci. Součástí jsou mimo jiné analýzy trhu, předpokladů a rizik. Druhým dokumentem je studie proveditelnosti. Zpravidla se provádí po studii příležitosti, která doporučila projekt realizovat. Cílem studie je odhadnout náklady, časový rozsah projektu a „ukázat nejvhodnější cestu k realizaci" [5].

Druhou fází je fáze projektová. Ta je podrobně popsána v následující sekci 2.2.

Poslední, třetí fází, je fáze poprojektová. Této fáze se většinou neúčastní tým, který po celou dobu pracoval na projektu. Je to především z důvodu zachování objektivity hodnocení. V této fázi se analyzuje průběh projektu a chyby, ke kterým došlo a které by se v dalších projektech neměly opakovat.

2.2 Projektová fáze

Praktiky projektového řízení nestanoví, jak přesně postupovat, ale určují myšlenkový směr, jakým se ubírat. Praktik je mnoho, proto je potřeba zvolit takovou, která bude dostatečná, ale zároveň nebude příliš mohutná [13].

„Cílem projektového řízení je úspěšně zrealizovaný projekt." [13] Nutně to ovšem neznamená, že neúspěšné projektové řízení vyústí v neúspěšný projekt a naopak, že úspěšné projektové řízení dokáže úspěšně zrealizovat jakýkoliv projekt.

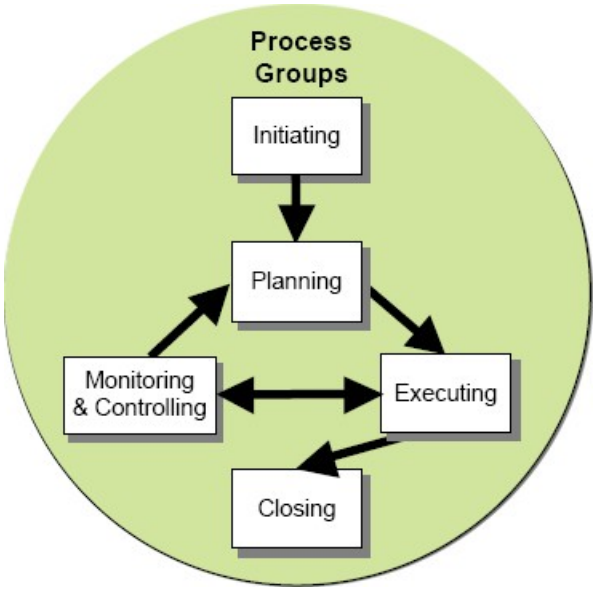
Projektovou fázi lze rozdělit na čtyři na sebe navazující oblasti — zahájení, plánování, řízení a ukončení. Vzájemnou provázanost přehledně znázorňuje obrázek 2.1.

2.2.1 Zahájení

V zahajovací části je zahájen celý projekt, vytvořena zakládací listina projektu. V této části je také složen tým lidí, který se na projektu bude podílet, tedy tzv. projektový tým. Probíhá i analýza zadání a cílů a nejpozději v této fázi je vytvářen logický rámec [5].

2.2.2 Plánování

Plánování můžeme rozdělit na dvě skupiny [14]. První skupinou jsou „základní plány projektu," do kterého patří plánování rozsahu, zdrojů a nákladů. V zásadě tvoří základ trojimperativu. Druhou skupinou jsou „doplňkové plány,"



Obrázek 2.1: Životní cyklus projektové fáze, převzato z [1]

které se zabývají dalšími částmi projektového plánování — riziky, komunikací a dalšími. Základní plány projektu i doplňkové plány jsou na stejné úrovni významnosti.

V první řadě je potřeba stanovit cíl a strategii jeho dosažení. Po tomto kroku je již znám cíl projektu, který můžeme dále dekomponovat na menší části (například podle WBS na pracovní balíky) a jednotlivé úkony.

Jednotlivým pracovním balíkům je pak potřeba přiřadit délku jejich trvání. Jakmile má každý pracovní balík přiřazenou dobu trvání, je dalším krokem stanovení návaznosti jednotlivých pracovních balíků. Návaznost i délku lze dobře ilustrovat pomocí Ganttova grafu.

Po stanovení časového průběhu projektu můžeme k jednotlivým pracovním balíkům přiřadit pracovní zdroje, které se na řešení příslušného úkolu budou podílet. Jednotlivé přidělené odpovědnosti se zapisují do matice odpovědností.

Nyní již známe části projektu, ke kterým jsou přiřazeny zdroje a ty lze nacenit. Můžeme také stanovit náklady na projekt jako celek.

Zbývá stanovit kontrolu projektu. Plánuje se především frekvence kontrol a to, jakým způsobem bude kontrola prováděna. V samotném závěru jsou vyřešena identifikovaná rizika. Řízení rizik poskytuje mnoho metod evidence rizik. Většina z nich má ovšem společné atributy jako název, pravděpodobnost nástání a dopady na projekt [12, 15].

Všechny tyto informace se následně zanáší do projektové dokumentace, podle které se v další fázi postupuje.

2.2.3 Řízení

Druhou fází je řízení projektu. Tato fáze se týká především správné organizace, efektivního využívání zdrojů, kontroly, rozhodovacího systému a snahy dodržovat předem stanovený plán [15]. V průběhu projektu může nastat odchýlení od původního plánu a je tedy možné plán určitým způsobem modifikovat.

Jedním ze základních stavebních kamenů řízení projektu je získávání informací a komunikace projektovým týmem. Pomocí ní lze sledovat progres prací na projektu, koordinovat úkoly a podobně. Dobrým prostředkem pro komunikaci jsou porady s různým zaměřením, s různou pravidelností.

Komunikace nemusí probíhat jen osobně, ale i formou nějakého dokumentu. Zástupcem mohou být například reporty o stavu projektu a dokumenty pro schvalování změn.

Úlohou projektového manažera během řízení projektu je i motivace členů týmu. Musí podporovat zájem o vyřešení přiděleného problému a vytvářet příjemnou atmosféru v kolektivu. Během průběhu projektu totiž může dojít k rozporu v projektovém týmu ať už kvůli osobním antipatiím nebo neshodě na výbraném řešení problému. V zájmu úspěšného dokončení projektu je dobré tyto stavy eliminovat.

Během řízení také nesmíme opomenout monitoring projektu. Mezi nejdůležitější patří

- kontrola dodržování termínů,
- monitoring nákladů, zda nedochází k překročení nákladů,
- monitoring rizik, především sledování trendu stanovených rizik a hledání možných nových.

2.2.4 Ukončení

Po dodání smluvených produktů nebo služeb nastává fáze ukončení [5]. Podle [14] se jedná nejčastěji o nesprávně vykonávaný proces projektového řízení. Ukončení projektu se buď zcela ignoruje nebo rychle odbude. Ve fázi ukončení je důležité oddělit projektovou fázi od fáze provozní, jinak se může velmi lehce stát, že se projekt bude neustále protahovat. Prakticky se to stává v případě, kdy je produkt dodán zákazníkovi. Ten ovšem má požadavky na drobné úpravy, změny nebo nové funkce. Projekt tak prakticky nemusí mít nikdy konec.

Do fáze ukončení se projekt může dostat po splnění stanovených podmínek. Projekt musí splnit jak očekávání zákazníka, tedy smluvenou funkčnost produktu a průchod akceptačními testy, tak i očekávání naše, jakožto nějaké organizace, které musí přinést zisk [14, 5].

Při ukončení projektový manažer vytvoří vyhodnocení a dokumentaci k projektu, kde jsou shrnuty všechny zkušenosti (pozitivní i negativní) nabyté bě-

hem průběhu projektu. Též je třeba udělat finanční vypořádání se všemi zúčastněnými stranami, uzavřít účetnictví a vypočítat celkové náklady na projekt [5].

Na závěr je nutné všechny tyto dokumenty řádně a přehledně archivovat včetně výstupů dodaných zákazníkovi. Lze z nich tak čerpat pro budoucí projekty.

2.2.4.1 Mimořádný konec

Projekt, který se nachází v problémech, by mělo být možné v jakékoliv fázi předčasně ukončit. Pokud je projekt předmětem smlouvy, s ukončením musí souhlasit obě zúčastněné strany. Problémy, které mohou vést k mimořádnému konci jsou podle [5] například

- hluboká krize v projektu, kterou se nepodařilo odstranit,
- změna firemní strategie nebo
- válečný stav.

2.2.4.2 Pozastavení projektu

Pozastavení znamená dočasné přerušení prací na projektu. Může se tak stát z mnoha důvodů — jako zástupce lze například jmenovat mimořádnost v projektu nebo legislativní změny. V rámci pozastavení projektu se můžeme setkat s termínem „aktivizace," který označuje opětovné zahájení prací na projektu. Při aktivizaci často dochází ke změnám zakládací listiny. Cílem změn jsou často „termíny ukončení, navýšení rozpočtu, přeplánování některých akcí nebo změny ve složení projektového týmu" [5].

2.3 Standardy projektového řízení

Na světě existuje několik standardů projektového řízení. Jde o osvědčené postupy a zkušenosti projektových manažerů, které jsou vedeny formou doporučení. Neznamená to tedy nutně dodržovat všechny kroky uvedené v příslušném standardu. Důležité je pro projekt vybrat správnou metodu v závislosti na jeho velikosti a dalších aspektech.

Mezi nejpoužívanější standardy ve světě patří níže uvedené, které v krátkosti představím.

2.3.1 PRINCE2

Tento standard je původem z Britských ostrovů, kde byl reakcí britské vlády na různou kvalitu projektů z IT. [5] Tento standard je využíván především v Evropě. I když byl původně zaměřen na IT projekty, dnes ho lze využít prakticky v jakémkoliv odvětví.

PRINCE2 je rozdělen na čtyři základní části — principy, procesy, témata, které se každé dále dělí na 7 částí, a prostředí projektu [16]. Principy vyjadřují teoretický základ, na kterém další části staví. Procesy se zabývají životním cyklem projektu od jeho založení až po uzavření. Témata se pak zaměřují na řešení situací, jako jsou rizika, mechanismy pro sledování projektu nebo plánování. Poslední část uvádí, jak přizpůsobit tento standard různé velikost projektu a jeho délce.

2.3.2 PMBOK

Standard má svůj původ v USA, kde je také nejpoužívanější. Patří pod organizaci PMI, která jej spravuje a nadále vyvíjí. [5].

PMBOK se dělí na devět znalostních oblastí a pět procesních skupin [17]. Skupiny opět popisují životní cyklus projektu od zahájení projektu po jeho ukočení. Oproti PRINCE2 se v PMBOK řeší i řízení lidských zdrojů. Zařazeny sem jsou metody pro motivování a rozvoj pracovních zdrojů, jejich organizování, řešení konfliktů v týmu a další.

2.3.3 IPMA ICB

IPMA je nejstarším standardem, který vznikl v 60. letech minulého století. Tento standard využívá rozdělení na tzv. „kompetenční oblasti" (technické, behaviorální a kontextové), které se dále dělí na „elementy kompetencí." [5] Není tedy jasně stanovený proces, jakým má projekt procházet, ale jde spíše o sadu doporučení.

2.4 Identifikační listina projektu

Zakládací (identifikační) listina projektu je dokument, kterým se formálně zahajuje projekt. Tento dokument se používá během celého průběhu projektu. Jsou v něm typicky uvedeny finanční, časové a zdrojové mantinely. V případě překročení těchto mantinelů dochází k zásadní změně v projektu, kterou je potřeba neprodleně řešit.

ILP je řešena v projektové fázi v zahájení, kde je stanovován projektový tým. Do ILP se uvádí projektový manažer a kostra projektového týmu, kde jsou jednotlivým členům týmu přiřazeny role. Jako další parametry se zde uvádí konkrétní výstupy, záměr, cíl, důležité milníky a kritéria úspěšnosti [5, 18].

2.5 Logický rámec

Logický rámec je pomůcka pro stanovování cílů a jako podpora pro jejich dosahování. Má formu matice o čtyřech sloupcích a čtyřech řádcích a pohodlně

Záměr	Objektivně ověřitelné ukazatele	Zdroje informací k ověření	nevyplňuje se
Cíl	Objektivně ověřitelné ukazatele	Zdroje informací k ověření	Předpoklady
Výstupy	Objektivně ověřitelné ukazatele	Zdroje informací k ověření	Předpoklady
Klíčové činnosti	Zdroje	Časový rámec aktivit	Předpoklady
			Předběžné podmínky

Tabulka 2.1: Logický rámec, vytvořeno podle [5]

se vejde na jeden nebo dva listy A4. Logický rámec nám pomáhá projekt zasadit do širšího kontextu [5].

2.5.1 Formát

V této sekci popisuji, jaká data se v jednotlivých polích tabulky nacházejí a jaké mají náležitosti. Pro lepší pochopení je v tabulce 2.1 uveden logický rámec včetně informací, do kterého políčka se vyplňují jaké informace.

2.5.1.1 Sloupec cílů

Začneme od prvního sloupce, který se nazývá sloupcem cílů. V jeho nejspodnější buňce se nachází „klíčové činnosti." Není cílem vyjmenovat všechny aktivity, které budou v projektu potřeba, ale ty nejhlavnější, které budou tvořit páteř projektu.

V poli nad klíčovými činnostmi se nacházejí výstupy. Zde se uvádí, jaké produkty projekt dodá — tedy odpovídá na otázku co.

Posuneme-li se opět o pole výš, zde se nachází cíl. V rámci projektu je možné stanovit si maximálně jeden cíl, který je v rámci celého projektu neměnný. V případě více cílů je nutné pro každý cíl vytvořit samostatný projekt. Pole cíl poskytuje odpověď na otázku, proč projekt realizujeme.

Úplně nejvýše se pak nachází záměr. To je dlouhodobá strategie, ke které náš projekt jen přispívá nebo je nutnou součástí pro jeho dosažení.

2.5.1.2 Objektivně ověřitelné ukazatele a zdroje ověření

Druhý sloupec se zabývá objektivně ověřitelnými ukazateli. Abychom mohli ukazatele ověřovat, musí být měřitelné, tedy ke každému ukazateli musí být přiřazena hodnota, kterou chceme dosáhnout nejpozději s koncem projektu.

V tabulce je dobré mít minimálně dva na sobě nezávislé ukazatele. V případě měření pouze jednoho ukazatele můžeme dostat zkreslené výsledky.

Ve třetím sloupci jsou pak uvedeny zdroje informací k ověření. V tomto sloupci nalezneme jak budou výše zmíněné ukazatele ověřeny. Uvádí se sem i kdo, kdy a jaký časový inteval bude potřebný pro ověření ukazatelů, případně kolik to bude stát. Pro ověřování je vhodné oslovit nezainteresovanou osobu nebo nezávislý orgán.

2.5.1.3 Předpoklady

V posledním, čtvrtém, sloupci jsou uvedeny předpoklady, které jsou podmínkou úspěšné realizace projektu, a podle [14] i rizika. V prvním řádku se předpoklady nevyplňují, naopak se pod čtvrtý sloupec přidává pole „předběžné podmínky," kde jsou uvedeny podmínky pro to, aby se o projektu začalo vůbec uvažovat.

2.5.1.4 Řádek klíčových činností

Malá odlišnost se vyskytuje na řádku klíčových činností. Jak jsem zmínil výše, v prvním sloupci se nacházejí klíčové činnosti, které jsou potřeba pro vykonání projektu. V dalších sloupcích ovšem nenalezneme objektivně ověřitelné ukazatele a zdroje ověření.

Do druhého sloupce se vpisují pracovní zdroje, finance a případně materiál, který bude v rámci projektu potřeba. Tyto informace lze rozepsat k jednotlivým činnostem uvedených v prvním poli pomocí odrážkového seznamu.

Ve třetím sloupci je uveden hrubý časový rámec projektu. Zde se uvádí, jak dlouho jednotlivé klíčové činnosti budou probíhat a zapisuje se opět pomocí odrážkového seznamu k jednotlivým činnostem.

2.5.2 Vazby

V logickém rámci existuje vertikální a horizontální logika. Vertikální logika je vidět ve sloupci cílů. Ve směru shora dolů jsou vidět hierarchické vazby, ve směru zdola nahoru pak vztahy příčina — následek. Slovy řečeno: Splněním činností dosáhneme výstupů, které nám pomohou naplnit cíl. Tento cíl pak přispěje našemu záměru.

Horizontální logikou je myšleno, že vše uvedené v sloupci cílů má přiřazeno objektivně ověřitelné ukazatele, zdroje k ověření a předpoklady, u klíčových činností vstupy a termíny. Logický rámec se pak čte zleva doprava, využiji-li slovo z [5], „cik—cak." Slovně vyjádřeno: „Když budou splněny předpoklady pro projekt — tak můžeme provést aktivity s potřebnými zdroji a v uvedených termínech a také s uvažováním potřebných rizik. Když je splněno vše v tomto řádku — tak splníme výstupy projektu. Toto je třeba ověřit a také v souvislosti s výstupy projektu je třeba uvažovat uvedená rizika" [14]. A tak dále, dokud se nedostaneme na úroveň záměru.

2.5.3 Postup tvorby

Logický rámec lze vytvořit v jedenácti krocích [5]:

1. stanovení cíle projektu,
2. stanovení výstupů projektu,
3. stanovení klíčových činností,
4. stanovení záměru,
5. ověření vertikální logiky,
6. stanovení objektivně ověřitelných ukazatelů u záměru, cíle a výstupů,
7. stanovení prostředků a způsobu ověření,
8. stanovení předpokladů s postupen zdola nahoru,
9. určení nákladů a časového rámce,
10. vykonání kontrolního testu kvality logického rámce (v [5] je uvedeno 28 kontrolních otázek)
11. porovnání s podobnými již realizovanými projekty.

2.6 Projektová dokumentace

Tento dokument se zabývá otázkami, jak celý projekt bude probíhat a kdy a jak skončí. Projektová dokumentace se vytváří specifikací požadavků zákazníka. Podle [19] musí projektová dokumentace odpovídat na následující otázky:

- proč — aneb jaký problém má projekt vyřešit a stanovení cílové vize,
- co — definování výstupů projektu po jeho ukončení, stanovení cílů
- kdo — vymezení účastníků na projektu a jejich organizace
- kdy — stanovení časového harmonogramu lidských zdrojů a ohraničení fází projektu nějakým konkrétním termínem.

Podle [20] by se projektová dokumentace měla zabývat následujícími částmi:

- plán řízení projektu — stanovení hlavních milníků, plán řízení změn harmonogramu,
- plán řízení předmětu projektu — detailní harmonogram prací, řízení změn,

- plán řízení nákladů — rozpočet, jak se budou řídit odlišnosti od plánu [5],
- plán obsazení projektu — stanovení rolí, odpovědnosti účastníků na projektu,
- plán řízení projektové dokumentace — popis komunikace,
- plán řízení subdodávek — koordinace dodavatelů,
- plán řízení rizik — evidence rizik a plán reakcí na riziko,
- plán řízení kvality — jak bude dosaženo kvality.

V závislosti na velikosti projektu lze některé části vynechat. Jak je uvedeno v [20], tento dokument se může omezit až na „harmonogram, rozpočet a pravidla komunikace a řízení změn."

2.7 WBS

WBS je metoda, která pomáhá nechat rozpadnout velké úkoly nebo celé projekty na jednotlivé pracovní balíky. Pracovní balík, nejmenší možná jednotka ve struktuře WBS, se skládá z několika úkolů. Každý úkol, který je potřeba na projektu vykonat, by tak měl být obsažen v některém z pracovních balíků. Celá struktura WBS by neměla mít více než 3-4 úrovně, neboť struktura se pak stává nepřehlednou.

S pracovními balíky z WBS se dále pracuje při naceňování projektu jako celku, tvorbě harmonogramu nebo přiřazování balíku spolupracovníkům na projektu.

WBS lze vytvářet dvěma způsoby [21]:

- produktově orientovaná — necháváme projekt rozpadnout na samostatné celky, jednotlivými pracovními balíky jsou fáze životního cyklu produktu,
- fázově orientovaná — necháváme rozpadnout podle životního cyklu, jednotlivé pracovní balíky jsou konkrétní výstupy.

Oběma způsoby se lze dostat k totožným výsledkům, což se dá využít k základní validaci správného vytvoření WBS.

Výše zmíněné způsoby se získávají pomocí dekompozice, kdy máme hlavní výstup a necháváme ho dále rozpadat, jedná se tedy o filozofii TOP—DOWN. Druhou možností je BOTTOM—UP, kdy si určíme jaké všechny výsledky požadujeme a z těch pak postupně skládáme celek [5].

2.8 Časový plán

Uvažujeme, že máme dekomponovaný náš hlavní výstup na menší části — pracovní balíky. Těm je nyní potřeba přiřadit odhadovanou dobu trvání a vytvořit mezi nimi návaznost.

Dobu trvání nelze určit přesně. Může nastat problém, se kterým jsme původně nepočítali a činnost nám prodlouží. Dobu trvání lze odhadnout osobní zkušeností s podobným typem práce, odhadem od experta nebo kvantitativním odhadem. Například pokud víme, že potřebujeme položit 500 m kabeláže a 100 m kabeláže nám zabere 2 hodiny, pak budeme potřebovat na položení 500 m kabeláže 10 hodin [14]. Do úvahy však nepřichází složitost přístupu, která výsledný čas může ovlivnit.

2.8.1 PERT

Další metodou používanou pro odhad času je PERT. Tato metoda využívá ke stanovení doby trvání statistiku, konkrétně rozdělení beta. Využívá se u projektů, kde jednotlivé činnosti nemají přesně danou dobu trvání. Střední doba trvání se proto odhaduje pomocí určení nejkratší možné doby (optimistický odhad) — t_{min}, nejkratší doby (pesimistický odhad) — t_{max} a doby trvání za normálních podmínek — t_{mid}. Výpočet střední hodnoty doby trvání je ukázan na vzorci 2.1 [22].

$$t = \frac{t_{min} + 4t_{mid} + t_{max}}{6} \tag{2.1}$$

2.8.2 Síťový graf

Pomocí síťového grafu můžeme znázornit návaznost jednotlivých pracovních balíků a vytvořit logickou posloupnost. Grafy rozlišujeme na: [5]

- uzlově orientované — v tomto případě uzel představuje činnost, která je ohodnocena dobou trvání,
- hranově orientované — hrany představují činnosti, uzly hranice začátku a konce činnosti.

Uzlově orientovaný graf se dnes používá v drtivém počtu případů, hranově orientovaný graf se takové popularitě nětěší a prakticky se nepoužívá.

2.8.3 Ganttův graf

Ganttův graf vychází z Ganttova diagramu, kde jsou jednotlivé činnosti znázorněny jako obdélníky. Délka obdélníků reflektuje stanovenou dobu trvání. Ganttův graf oproti diagramu obsahuje také vazby mezi jednotlivými činnostmi.

Dalo by se tedy říct, že Ganttův graf je vylepšením síťového grafu. Dnes se Ganttovy a síťové grafy prolínají a často se navíc doplňují dalšími informacemi, jako například přiřazené zdroje apod. [5].

Příklad Ganttova grafu lze vidět v projektové dokumentaci na obrázku C.4.

2.8.4 Metoda kritické cesty

Metoda kritické cesty staví na uzlově orientovaném síťovém grafu. Narozdíl od klasického síťového grafu, uzel nenese jen informaci o délce trvání, ale i o nejdřívějších a nejpozdějších začátcích a koncích a časových rezevách. U této metody rozlišujeme dvě časové rezervy

- celková rezerva (RC) — doba, o kterou lze odložit začátek činnosti nebo ji prodloužit, aniž by se změnila doba trvání projektu,
- volná rezerva (RV) — doba, o kterou lze odložit začátek činnosti nebo ji prodloužit, aniž bychom ovlivnili začátek následující činnosti.

Pomocí metody kritické cesty můžeme stanovit tzv. kritickou cestu. To je cesta grafem s nejdelší dobou trvání. Uzly s celkovou rezervou 0, leží na kritické cestě. Zpoždění těchto činností má za následek, že se zpozdí celý projekt [5].

2.9 Plánování pracovních zdrojů

Čas pro naplánování pracovních zdrojů přichází tehdy, pokud máme připravený rámcový odhad časového rámce celého projektu [5, 14]. V rámci plánování zdrojů můžeme v některých případech počítat s nepřímou úměrou mezi délkou trvání a počtem přiřazených zdrojů. Neplatí to ovšem obecně, neboť některé činnosti zkrátka mají stanovenou dobu, kterou jakýkoliv počet zdrojů nijak neurychlí (například tvrdnutí betonu).

Jako první krok v plánování pracovních zdrojů je stanovení profesí k jednotlivým pracovním balíkům, jejich počet a umístění. Dalším krokem je zjištění skutečného stavu — tedy to, kolik je k dispozici zdrojů v námi potřebný čas a zda nejsou alokovány na jiných projektech společnosti.

Dalším krokem je porovnání. V tom ideálním případě jsou všechny potřebné zdroje dostupné a plánování pracovních zdrojů pro nás skončilo. Častější verzí je ovšem kolize, kterou lze vyřešit některým z následujících způsobů:

- přesun nebo rozložení činnosti v rámci jejich rezerv,
- přesun činnosti mimo rezervy, z čehož může vyplynout prodloužení projektu,
- vyšší využívání zdrojů (přesčasy), které se ovšem dají praktikovat jen krátkodobě,

- najímání dalších zdrojů, které zvyšuje náklady nebo
- zadání zakázky externímu dodavateli.

Plánování zdrojů se dá vyjádřit i graficky. Jednou z možností je histogram, kde na vodorovné ose máme čas a na vertikální počet zdrojů. Lze tak v rámci celého projektu sledovat vývoj potřebných zdrojů a porovnávat ho s dostupnými pracovními zdroji.

2.10 Náklady na projekt

Rozpočet projektu je zajímavý pro všechny — vlastníky projektu zajímá cena a kolik projekt vydělá, koordiátory finance, které mají vynaložené na aktivity a také zaměstnance, kteří se zajímají o svou mzdu [5]. Náklady vnímáme jako spotřebu peněz. Ty nějakou formou zpracováváme a vytváříme výsledný produkt. V nákladech je dobré nechat si určitou rezervu pokrývající rizika (ať už identifikovaná nebo ne), která mohou nastat v průběhu projektu.

Náklady na projekt už jsou stanovovány v předprojektové fázi v logickém rámci a jsou jedním z jeho výstupů. Vzhledem k tomu, že ještě nejsou rozepsány konkrétní činnosti, které je potřeba vykonat, jedná se o velmi hrubý odhad.

2.10.1 Přímé a nepřímé náklady

Náklady na projekt lze rozlišit na přímé a nepřímé. Do přímých nákladů můžeme započítat činnosti, hmotné a nehmotné majetky, které se vážou ke konkrétnímu projektu. Jako příklad můžeme uvést cestovné, náklady na nákup služeb a pořízení hmotného a nehmotného majektu.

Nepřímé náklady jsou takové náklady, které nelze přisoudit k nějakému konkrétnímu projektu. Jedná se například o pronájmy budov nebo kanceláří, náklady na podpůrná oddělení. Do nepřímých nákladů se také započítávají daně.

2.10.2 Metody odhadování nákladů

Odhadování nákladů můžeme rozdělit na shora—dolů a zdola—nahoru. Míra přesnosti je svázána nákladností takového odhadu.

2.10.2.1 Shora—dolů

Tento typ odhadování je velmi rychlý, ale velmi nepřesný. Zároveň není zapotřebí detailního plánu projektu. Odhad se nakonec oproti skutečnosti může lišit v řádech [5]. První metodou shora—dolů je metoda analogického odhadu. Využívá již uzavřených projektů a jejich konečných nákladů na projekt. Lze

se tedy podívat zpět do historie a porovnat aktuální činnost s již dříve vykonanou.

Parametrický model je druhá metoda odhadu. Tato metoda se využívá u projektů s nějakou charakteristickou vlastností. Je stanovena jednotková cena, od které se vypočítá cena celého projektu [14].

2.10.2.2 Zdola—nahoru

Tento typ je narozdíl od předchozího přesnější, ale také časové náročný a nákladný [14]. Jednotlivé činnosti na projektu se odhadují zvlášť a celková cena projektu je pak součtem jednotlivých odhadů. Proto se využívá až v částech, kdy jsou přesně známy všechny pracovní činnosti.

2.10.2.3 Expertní odhady

Mezi metody odhadování nákladů můžeme zařadit i expertní odhady. Jedná se o člověka nebo skupinu lidí se znalostmi dané problematiky. „Tato varianta se nejčastěji používá v případech, kdy je časově náročné nebo nákladné zjišťovat ceny z ověřitelných zdrojů" [5].

2.11 Komunikace

Nejen mezi členy projektového týmu musí fungovat komunikace. Pomocí ní předáváme informace o průběhu nebo problémech, které nastaly. O průběhu projektu je také vhodné informovat vlastníka projektu, jak se práce vyvíjí. Ne každá informace je vhodná pro všechny zainteresované strany. Je nutné oddělit vnitřní záležitosti, reportování vlastníkovi projektu a pokud je potřeba, rozlišit marketingovou komunikaci.

2.11.1 Typy komunikace

Rozlišujeme tři typy [14]: povinná, nepovinná a marketingová. Povinná je zaměřena na kontrolu projektu, monitorování nebo podávání zpráv.

Druhým typem komunikace je komunikace nepovinná, též nazývaná jako informační. Pomocí této komunikace pracovníci projektu získají informace o projektu pomocí konzultací nebo z určených dokumentů.

Posledním typem je marketingová komunikace. Jejím hlavním účelem je vytvoření zájmu o projekt a jeho výstupy.

2.11.2 Plán komunikace

Komunikační plán nám poskytuje informace o tom, „kdo potřebuje jakou informaci, kdy ji bude potřebovat a jak mu bude předána" [14]. Plán komunikace můžeme názorně vytvořit jako komunikační mapu [5] nebo ji podrobněji rozepsat do tabulky.

Celý plán pak dokáže odpovědět na otázky co je za potřebí, kdy je informace potřeba, respektive jak často, komu ji předat a jakou formou bude předána. Do tabulky se také uvádí jakého typu komunikace je.

U komunikačního plánu je důležité zvolení vhodné technologie pro předávání informací.

2.12 Kontrola projektu

Na začátku projektu jsme stanovili nějaký plán, která se snažíme dodržet. Jak je uvedeno v [5]: „Ohleně plánu máte jedinou jistotu — tak jak je naplánován, projekt nikdy neproběhne." Cílem kontroly projektu je sledovat vývoj a reagovat na změny a vzniklé odchylky.

V podstatě se jedná o kolečko, kdy do našich činností vstupují náhodné vlivy. Ty nějakým způsobem ovlivňují náš projekt. Ze zprávy o projektu se dozvíme reálný stav projektu, který můžeme porovnat s naším plánem. S vysokou pravděpodobností bude projekt odchýhlen od plánu. Tyto odchylky musíme rozeznat a zasáhnout.

2.12.1 Zprávy o průběhu

V průběhu práce na projektu jsou podávány zprávy, někdy se tato činnost též označuje jako reporting. Tyto zprávy se podávají v pravidelných intervalech a je vhodné je uvést do plánu komunikace. Frekvenci těchto zpráv je dobré volit takovou, aby managemet stačil tyto zprávy zpracovávat. Interval se nejčastěji stanovuje jako třetina z průměrné délky činnosti.

Pokud projekt probíhá podle stanoveného plánu, není potřeba podávat podrobnou zprávu.

2.12.2 Porovnávání se skutečností

Možnou metodou pro porovnání se skutečností je metoda procentního plnění. Do Ganttova grafu se dopisuje procento splnění konkrétní činnosti. Využívá se především u menších projektů a tam, kde se sleduje jen jedna složka plnění.

Další metodou je stavové sledování projektu. Každá činnost může mít tři různé stavy 0, W a 100, které značí, že se činnost neprovádí, pracuje se na ní anebo je hotová. Existuje i modifikace s přidáním 90. Značí, že je činnost hotova, ale ještě čeká na schválení.

Poslední metodou, kterou zde zmíním, je metoda milníků. Vytvoříme větší počet milníků než je významných událostí v projektu. V tyto dny pak vyhlásíme tzv. kontrolní den, kdy se vyhodnocuje stav projektu. Na tento den musí být připraveny všechny zprávy o stavu projektu a často se také vytvářejí zprávy o budoucím výhledu [5].

2.13 Odpovědnosti členů týmu

Pro stanovení odpovědností členů týmu na projektu se využívá matice odpovědnosti. V hlavičce tabulky jsou uvedeni členové týmu včetně projektového manažera a v prvním sloupci pak jednotlivé pracovní balíky vzniklé z WBS. Ke každému pracovnímu balíku jsou následně přiřazovány odpovědnosti.

Variant odpovědností je několik, nejpoužívanější je ovšem varianta RACI [23]. Ke každému pracovnímu balíku se přiřazují pracovní zdroje s následující odpovědností:

- Responsible (odpovědný) — přiřazuje se člověku nebo skupině lidí, která je odpovědná za vypracování pracovního balíku,
- Accountable (schvalování) — přiřazuje se právě jednomu člověku, který je za projekt zodpovědný, nemusí se přímo podílet na tvorbě,
- Consulted (konzultace) — většinou se přiřazuje odborníkovi na konkrétní činnost, se kterým může být práce konzultována,
- Informed (informování) — přiřazuje se lidem, kteří by měli být informování „o průběžném stavu a výstupech z dané oblasti" [23].

Do tabulky se zpravidla píší jen začáteční písmena konkrétní zodpovědnosti.

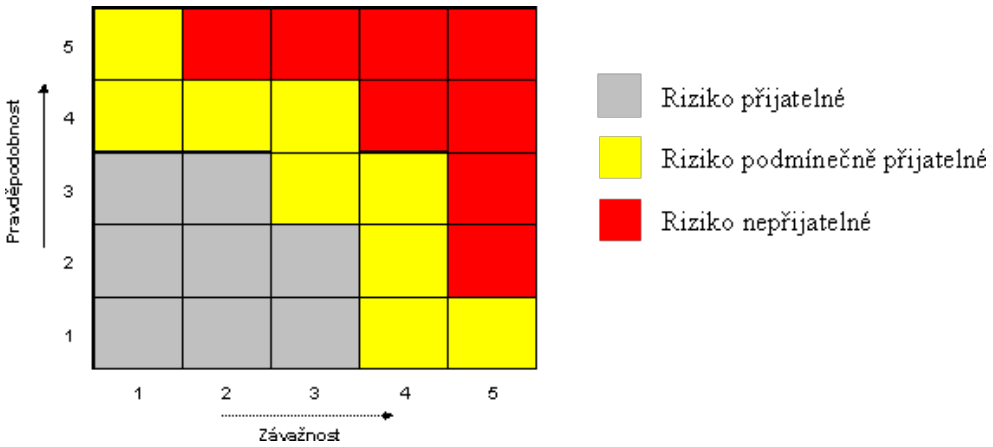
2.14 Rizika projektu

Riziko projektu je událost, která nastane s určitou pravděpodobností a nějakým negativním způsobem ovlivní náš projekt. Mezi typické negativní ovlivnění patří prodloužení projektu anebo vyšší náklady na projekt. Abychom udrželi náklady a stanovený harmonogram, musíme rizika v projektu identifikovat.

Odhalováním rizik se u menších projektů zabývá projektový manažer nebo člen projektového týmu. Některé společnosti si mohou vést checklisty rizik, které postupně pověřený člověk prochází a hledá taková rizika, která by mohla nastat. Druhou možností je brainstorming. Během odhalování rizik je také dobré zamyslet se nad symptomy, které riziko spouští [14]. Výčet odhalených rizik není konečný, vždy se může objevit nové, neidentifikované riziko.

2.14.1 Hodnocení významu rizika

Rizikům je potřeba stanovit pravděpodobnost výskytu. Hodnota pravděpodobnosti se během projektu neustále mění, a proto je třeba ji sledovat. K tomu je vhodným nástrojem registr rizik, který může mít mnoho podob a v dnešní době na vytvoření registru existují speciální programy. V podstatě se jedná o tabulku, ve které jsou všechna evidovaná rizika a k nim další informace.



Obrázek 2.2: Matice kvalitativního hodnocení rizik [2]

Riziku je potřeba přisoudit i jeho vliv na projekt. Společně s pravděpodobností výskytu rizika tyto dva atributy určují celkový význam rizika v projektu. Tyto dva atributy dohromady určují význam na projektu.

2.14.1.1 Kvalitativní hodnocení

Při kvalitativním hodnocení rizika využíváme slovní stupnice. Základ tvoří stupně nízká, střední a vysoká. Tuto stupnici můžeme rozšířit o další dva stupně — velmi vysoká a velmi nízká. Záleží na konkrétní společnosti, jak podrobnou stupnici chce evidovat.

Podobným způsobem ohodnotíme i vliv na projekt. Ohodnotit můžeme vliv na náklady, čas a kvalitu. Hodnocení probíhá na základě kvalitativní stupnice, kde ke každé dimezi projektu a vlivu přiřadíme hodnody anebo stanovíme míru ovlivnění.

Nyní máme ke každému riziku stanovený vliv a jeho pravděpodobnost a můžeme posoudit jeho celkový význam v projektu. K tomu se využívá matice rizik uvedená na obrázku 2.2. Celkový význam v projektu nám vyjde takový, kde se střetne pravděpodobnost a vliv v tabulce. Oblasti tabulky pak určí, jak velký význam dané riziko má. Význam se pak zaznamenává do tabulek.

2.14.1.2 Kvantitativní hodnocení

Kvalitativní hodnocení je velmi podobné předchozímu typu. Nyní ale již nepracujeme se škálou, ale konkrétními čísly. Je proto nutné znát číselné ohodnocení vlivu i pravděpodobnosti.

První metodou je statistická peněžní hodnota. K práci je zapotřebí znát cenu projektu. Cenu projektu vynásobíme pravděpodobností nastání rizika a vlivem rizika na projekt. Výsledek, který touto operací získáme, nazýváme peněžní hodnotou rizika [14].

Druhou metodou je citlivostní analýza, která zkoumá potenciální vliv rizikové události na zkoumaný objekt [14]. Předpokládá se využití matematické formule, do které jsou zahrnuty všechny zkoumané atributy. Atributy mají dopředu stanovené nebo odhadnuté hodnoty. Hodnotu jednotlivých atributů postupně zvyšujeme o stejné procento a vypočítáme celkový výsledek a procentní změnu oproti původnímu výpočtu. Atribut s největší procentní změnou oproti původnímu je nejcitlivější a tedy pro nás i nejvíce rizikový.

Často se stává, že rizika lze zachytit ve formě stromu. Metoda stromu rizik se zakresluje ve formě grafu, kde se nachází kořen, větve a listy. V kořenu se nachází nějaká hrozba a té přisoudíme pravděpodobnost nastání. Dále na řadu přichází diskuze možných následků, které větvíme a zapisujeme do listů a ohodnotíme pravděpodobností. Lze pak jednoduše vypočítat pravděpodobnost konkrétního případu vynásobením hodnot pravděpodobností v kořeni a v listu. [5]

2.14.2 Reakce na rizika

Identifikovali jsme rizika a nyní je vhodné uvažovat nad tím, jak na takovou situaci reagovat. Reakce můžeme rozdělit do několika kategorií [14].

- Riziko odmítneme. V praxi se jedná o vytvoření upraveného plánu, ve kterém dané riziko není součástí.
- Snažíme se riziko redukovat tak, aby nenastalo.
- Nouzové řešení. Riziko 100% nastane a naším cílem je co nejvíce zmírnit dopad na projekt.
- Přesuneme riziko na třetí stranu, ve většině případů na pojišťovnu. To s sebou nese i jisté náklady, které je potřeba započítat do projektových nákladů.
- Riziko přijmeme. Děláme v případech, „kdy je výhodnější nedělat nic, než realizovat samotné opatření."
- Sdílení rizika s dalšími subjekty. Takový případ je vhodné ošetřit smlouvou. V případě nastání rizika se tyto subjekty podílí na jeho dopadech. Pokud je projekt úspešný, mohou dostat podíl ze zisku. [13]

2.15 Změnové požadavky

Během realizace projektu se mohou vyskytnout požadavky na změnu. Změnu mohou vyvolat vnější podněty jako například dodatečné požadavky zákazníka nebo vnitřní podněty, které se týkají nejistoty v projektu.

Změny musíme řídit. Nelze implementovat každý nápad a stejně tak všechny nápady ignorovat.

Požadavek na změnu má obecně následující průběh [5, 24]:

- Identifikace změny — v první řadě musí být požadavek na změnu odeslán podle stanovených pravidel. V požadavku musí být uvedeno co se má měnit a z jakého důvodu.
- Analýza a způsob provedení — nárh je důkladně analyzován a rozhoduje se, jaký dopad bude mít na cíle, náklady a harmonogram. V tomto okamžiku jsou také analyzovány možné způsoby provedení.
- Schválení nebo zamítnutí — v závislosti na velikosti projektu a změny schvalování podléhá manažerovi projektu nebo vedení společnosti.
- Aktualizace plánu a provedení — pokud byla změna přijata, je nutné aktualizovat dotčené dokumenty jako například časový harmonogram anebo náklady. Změna se též uvádí do reportů.

Kapitola 3

Projektová dokumentace

Jedním z výstupů mé bakalářské práce je projektová dokumentace. V této části jsem se zaměřil na naplánování projektu tak, aby byl úspešný a nepřekročil stanovené hranice času a nákladů. Postup jsem volil podle výkladu z [5] a [14] a zaměřil jsem se na využití ověřených nástrojů projektového řízení.

3.1 Dekompozice projektových výstupů

V první řadě bylo zapotřebí dekomponovat systém jako celek. K tomu jsem využil WBS. Bylo možné vybrat jednu ze dvou možností — produktově orientovanou WBS nebo fázově orientovanou WBS. Jelikož se pomocí obou můžeme dostat k podobným nebo prakticky stejným výsledkům, zvolil jsem nakonec větvění podle fáze. Jednotlivé listy tedy vyjadřují činnosti, které je nutné vykonat k vytvoření produktu jako celku.

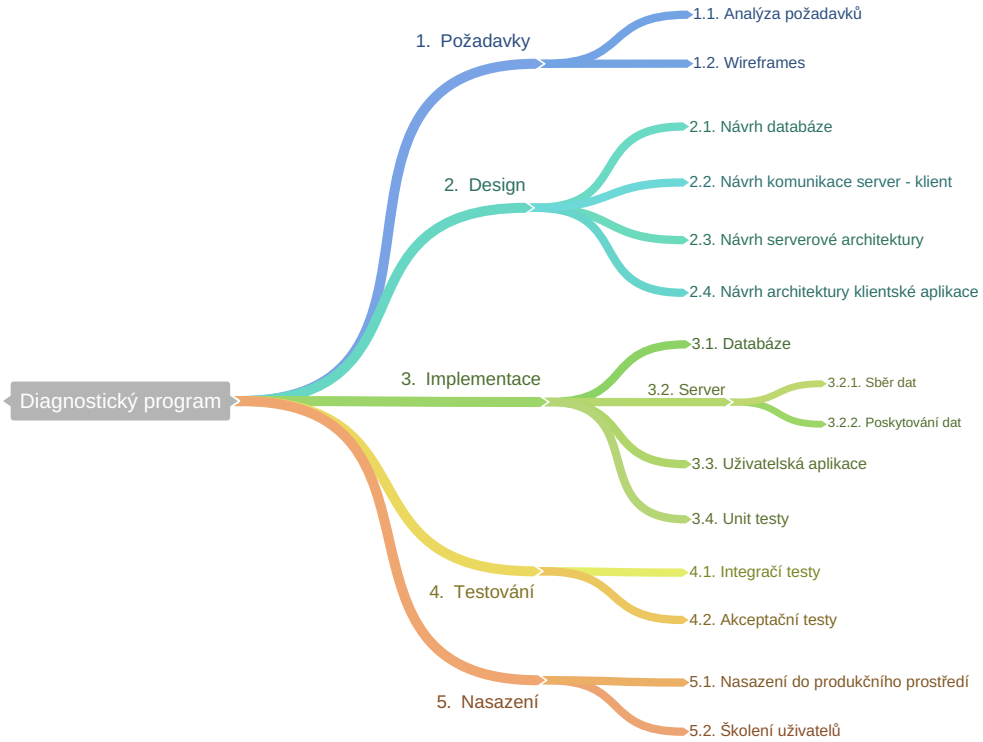
Jako fáze jsem zvolil životní cyklus software, tedy fáze analýzy, návrhu, samotné implementace, testování a nasazení. Každou tuto fázi jsem dále dekomponoval na jednotlivé pracovní balíky, které jsou již dále nedělitelné. Jedinou výjimku jsem vytvořil u implementace serveru, která se dala rozdělit na komunikaci s klientem a sběr dat z dopravního sálu.

Vytvořená WBS pro diagnostický program je vidět na obrázku 3.1. WBS byla vytvořena v programu [25].

3.2 Stanovení časového harmonogramu

Dále na řadu přišlo stanovení časového harmonogramu. Celý projekt jsem nejdříve rozdělil na jednotlivé fáze, které jsem dále popsal včetně konkrétních činností, které je potřeba v těchto fázích vykonat. Fáze jsem popisoval stejně jako při rozpadu pomocí WBS.

První fází byla analýza, kde jsme společně se zákazníkem stanovili cíle a výstupy, které požaduje. Další fází je návrh. V této fázi je zapotřebí vytvořit



Obrázek 3.1: WBS pro Diagnostický program

návrh aplikace, podle kterého se bude následně implementovat. Návrh jako celek je popsán v další kapitole.

Implementace je třetí fází. Zde jsem se hodně zaměřil na organizaci verzování softwarového produktu. Rozhodl jsem se využít verzovací nástroj git pro svou rozšířenost i velkou oblibu mezi programátory. Je zde popsáno, jaké větve je potřeba vytvořit a které za jakých okolností spolu budou spolupracovat. Model verzování byl převzat z [4]. V této části jsou definovány i technologie, tedy programovací jazyk a vývojové prostředí, které bude zapotřebí pro realiazaci.

Další fází je testování a nasazení. Zde jsem v krátkosti shrnul, jakým typům testů bude program podroben. Na závěr jsem zminil i údržbu a rozšiřování, které už prakticky nejsou zahrnuty v projektové fázi.

3.2.1 Harmonogram prací

Po popisu prací jsem vytvoři harmonogram prací. V této fázi jsem využil programu [26], který nabízí možnost rozplánovat jednotlivé činnosti, přiřadit k nim pracovní zdroje a jednotlivé pracovní balíky na sebe navázat.

Pro zobrazení harmonogramu jsem zvolil Ganttův graf, který ukazuje posloupnost jednotlivých činností. Ten je zobrazen na obrázku C.4.

V harmonogramu je možné si všimnout velké prodlevy mezi návrhem a implementací. Přibližně v době od poloviny května 2015 nebudou k dispozici pracovní zdroje, konkrétně programátor. Zároveň od přibližně stejného termínu do konce června bude na ČVUT končit semestr a probíhat zkouškové období. V tuto dobu je možné, že lektoři Dopravního sálu nebudou plně k dispozici a v Dopravním sále mohou probíhat další akce.

I z tohoto důvodu jsem se rozhodl projekt v toto období přerušit a opět začít v klidnější fázi o prázdninách. Další sedmi denní pauza je naplánována po konci implementace. Je to především z důvodů možného zpoždění.

3.3 Pracovní zdroje

Harmonogram byl stanoven. Na řadu přišlo přiřazení jednotlivých pracovních zdrojů. Pro vytvoření tohoto projektu jsem definoval, že v projektovém týmu budou zapotřebí následující role:

- projektový manažer,
- softwarový analytik,
- softwarový architekt,
- programátor a
- tester.

V rámci této bakalářské práce jsem si vyzkoušel role projektového manažera, kdy jsem vytvářel projektovou dokumentaci, softwarového analytika, když jsem analyzoval situaci v Dopravním sále, softwarového architekta při vytváření návrhu systému diagnostického programu a ve své podstatě i testera, když jsem navrhoval funkční a akceptační testy.

Situace mi tedy byla o něco ulehčena, jelikož jsem v jedné osobě představoval hned několik rolí na projektu. Bylo ale důležité nepřetížit pracovní zdroje, tedy abych pracoval maximálně na 100% na jedné konkrétní činnosti. I z tohoto důvodu jsem zvolil vývoj softwaru metodou vodopád, kde jednotlivé pracovní balíky navazují na sebe a k přetížení tedy nemůže dojít.

Vytváření histogramů jednotlivých zdrojů by tedy ukázalo jedině to, že po celou dosavadní dobu projektu jsem byl vytížen nejvíce na 100% a k přetížení zdrojů tedy nedošlo.

3.3.1 Matice odpovědností

Pro přiřazení pracovních zdrojů k jednotlivým pracovním balíkům jsem využil matici odpovědností. Je ideálním kandidátem. Vzhledem k velikosti projektu dokáže jednoznačně popsat přidělené povinnosti. I když standardů je několik,

zvolil jsem obecně nejoblíbenější metodu RACI, kterou jsem pro zachování jednotnosti jazyka převedl do české verze.

Zde jsem ke každému pracovnímu balíku přiřadil, kdo bude na balíku pracovat a jakým způsobem. Důležité bylo přiřadit ke každému pracovnímu balíku alespoň jednu pracující roli a právě jednoho schvalovatele.

3.4 Náklady

Číslem, které jistě každého zákazníka nejvíce zajímá, jsou finance. Pro odhad ceny jsem využil metodu zdola—nahoru, která by měla být přesnější a je vhodná pro použití v případě, že znám plán projektu.

Celkovou cenu jsem stanovoval podle jednotlivých pracovních balíků, do projektové dokumentace jsem následně přidal jen rozdělení na jednotlivé fáze. Je to především z důvodů skrytí částek za konkrétní úkony. Do této částky je potřeba ještě započítat nepřímé interní náklady.

Jednotlivé pracovní balíky jsem rozebral, stanovil jsem si, jaké úkony je potřeba vykonat a jaké zdroje jsou k úkolům přiřazeny. Bylo potřebné také myslet na to, že na úkolu se nepřímo podílí i projektový manažer, jehož cílem je projekt řídit a členy týmu motivovat. Tyto náklady byly započítány do nepřímých nákladů.

Nejvyšší částkou byla ohodnocena implementace. Zde bylo potřeba uvažovat s poměrně náročnou situaci, a to zasvěcení programátora do problematiky. Následně zahrnout implementaci samotnou, které rovněž nebude příliš jednoduchá. Druhou nejdražší položkou je návrh systému. Správný návrh systému je základním stavebním kamenem každého projektu v IT, a proto je vhodné se této fázi věnovat a investovat do ní.

Menšími položkami byly analýza a testování. V analýze je potřeba zaznamenat všechny požadavky zákazníka a analyzovat jej jako celek. Všechny tyto dokumenty pak pečlivě a především přehledně uschovat pro další pokračování projektu. Testování se pak zaměřuje na vykonání integračních a systémových. Testování se může potenciálně prodražit, pokud budou v systému nalezeny chyby. Tento proces se pak může prodlužovat a s tím i zákonitě prodražovat.

Nejlevnější fází je nasazení a do něj zahrnuté školení uživatelů. Jedná se o fyzickou instalaci na hardwaru zákazníka a následné proškolení uživatelů v přístupu, jak program používat.

Průměrné náklady byly vyčísleny na 400 Kč/h. Do této částky jsou započítány přímé náklady na zaměstnance i nepřímé režijní náklady na vedení projektu. Člověkodny a náklady na jednotlivé části včetně celkových nákladů jsou uvedeny v tabulce 3.1.

Práce	MD	Cena (v Kč)
Analýza	18	57 600
Návrh systému	36	115 200
Implementace	91	291 200
Testování	12	38 400
Nasazení a školení	7	22 400
Celková cena	164	**524 800**

Tabulka 3.1: Souhrn odhadované ceny projektu

3.5 Rizika

Ke každému projektu patří určitá rizika, která jsem se v projektové dokumentaci také snažil zachytit. V rámci rizik jsem se zaměřoval především na časové umístění projektu. Jeho implementační část se bude vykonávat o prázdninách. To je jeden z problémových faktorů, které díky absenci někoho z projektového týmu mohou projekt prodloužit. Druhým faktorem možného rizika je malá zkušenost projektového týmu, respektive projektového manažera, s projekty podobného typu. To může vyústit ve špatný nebo nerealizovatelný plán, kde bude potřeba dělat různé změny a projekt v konečném důsledku může být neúspěšný. I proto jsem toto riziko ohodnotil nejvyšší procentní hodnotou.

V projektové dokumentaci jsou zanesena rizika do registru rizik. Ke každému riziku byl uveden jeho stav, který slouží k základnímu monitorování rizik. Dále jsem uváděl, jaký dopad má riziko na projekt, plány pro jeho zmírnění a i krizový plán.

Jak jsem uváděl v teoretické části, rizika můžeme evidovat kvalitativně nebo kvantitativně. Rozhodl jsem se vyjádřit riziko kvantitativně, jelikož je možné dále pracovat při výpočtu hodnoty rizika. Procentní ohodnocení jsem dále využil v metodě statické peněžní hodnoty pro určení hodnoty rizika. Tyto výpočty jsem prováděl v tabulce 3.2. Jak je vidět z tabulky, největší peněžní hodnotu má riziko R04, tedy že se projeví nezkušenost projektového týmu. Tato hodnota tvoří 12% z celkových nákladů projektu. Jako opatření proti tou, aby tato situace nastala, bylo naplánováno zvýšené monitorování a tím zajištění případného včasného zásahu.

3.6 Komunikace a kontrola

V této části řeším komunikaci a kontrolu projektu. To lze rozdělit na dvě části: komunikaci interní a se zákazníkem.

Riziko (tabulka)	Pravděpodobnost	Vliv na projekt	Hodnota (v Kč)
R01 (C.1)	10%	20%	10 496
R02 (C.2)	20%	20%	20 992
R03 (C.3)	10%	10%	5 248
R04 (C.4)	30%	40%	62 976
R05 (C.5)	20%	20%	20 992
R06 (C.6)	10%	30%	15 744
R07 (C.7)	10%	50%	26 240

Tabulka 3.2: Peněžní hodnota rizik

3.6.1 Interní komunikace a kontrola

Interně bude celý systém veden elektronicky v systému Trello. Jde o online aplikaci, která nabízí prostor pro monitorování a realizaci projektu. Umožňuje vytváření karet, které mohou být ekvivalentem k pracovním balíkům. Zde pak bude podrobný popis aktivity, checklisty s provedenými pracemi a možnost konverzace nad daným úkolem. Cílem je uchovávání informací a know-how na jednom místě, ke kterému budou mít přístup všichni členové týmu. Zároveň bude mít celý projektový tým přehled o stavu projektu.

Je samozřejmě chybou, když se využívá jen jednoho komunikačního kanálu a v případě Trello ještě nepřímého. Například v situacích, kdy by bylo potřeba něco vyřešit urgentně, je dobré mít mezi sebou další komunikační kanály, ať už vyměněná telefonní čísla nebo real time messengery.

3.6.2 Komunikace se zákazníkem

Byla také stanovena komunikace se zákazníkem. S ním byla diskutována také možnost alternativního spojení pomocí různých komunikačních prostředků, které umožňují schraňovat všechny dokumenty a konverzace na jednom místě. Po analýze zákazníka jsem nakonec vybral emailovou konverzaci jako hlavní kanál a to zejména kvůli zvyku zákazníka. Při vhodném systému ukládání se v emailových klientech žádná zpráva neztratí.

Zákazníkovi bude zasílán report o stavu projektu. Podle [14] je u malých projektů vhodné zákazníka informovat každých 14 dní nebo měsíčně. Vzhledem k tomu, že projekt není až tak dlouhý, rozhodl jsem se pro čtrnáctidenní interval.

3.7 Změnové požadavky

V rámci projektové dokumentace jsem se zaměřil i na možné změny během realizace. Hlavním důvodem je stanovení mantinelů pro případné změny a možnost vyjasnit si se zákazníkem, že určité změny nebude možné provést a je proto nutné, aby s tím byl seznámen.

Důležité je, aby změnový požadavek obsahoval nejen změnu samotnou, ale i důvod, proč změnu provádět a také určit, komu takový požadavek směřovat. V tomto případě je logickým výsledkem kontaktování projektového manažera.

Kapitola 4

Návrh systému

V této části se věnuji návrhu systému diagnostického programu. Program jako celek jsme společně se zákazníkem vydefinovali během analýzy. Zároveň jsme si ujasnili, jaký vzhled přibližně aplikace bude mít.

Pro tento systém jsem využil architektury klient—server. Tato architektura se nabízela především z důvodu požadavku přístupu k datům z více klientských stanic.

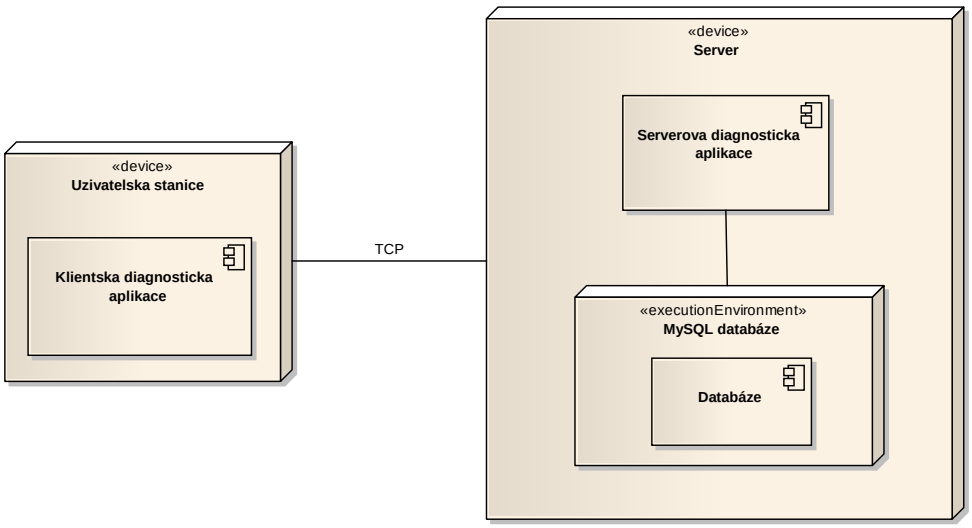
Celý systém byl navržen pro programovací jazyk C#, ve kterém je napsána aplikace sběru dat z diagnostikovaných zařízení v Dopravním sále. Vytvořit aplikaci v jiném programovacím jazyce jsem uvážil jako cestu špatným směrem, která by v konečném důsledku mohla napáchat více škody než užitku. Bylo by potřeba vytvářet nějaký konektor mezi vybraným programovacím jazykem a C#, proto jsem od tohoto úmyslu upustil.

Celý systém jsem koncipoval tak, že bude vytvořena aplikace běžící na serveru, která bude mít přístup k MySQL databázi. Na počítačích v Dopravním sále poté poběží aplikace diagnostického programu. Celá tato myšlenka je zachycena na obrázku 4.1.

V rámci návrhu architektury jsem se snažil zachovávat jmennou konvenci, všechny třídy programovat proti rozhraní a nikoliv proti implementaci, jak je uvedeno v [27]. Interfacy jsou označeny počátečním písmenem „I," třídy začínají velkými písmeny. Veškeré obrázky byly vytvořeny v aplikaci [7].

4.1 Databázový model

V první řadě jsem se věnoval databázi. Jak už jsem zmiňoval v analýze, databázi jsem navrhl rozdělit na dvě části. V té první půjde o uchovávání dat z Dopravního sálu. Budou zde evidovány všechny stavy, signály a monitorované moduly. V druhé oddělené databázi budou zaznamenávány stavy, které server bude identifikovat jako chybové.



Obrázek 4.1: Diagram nasazení

4.1.1 Produkční databáze

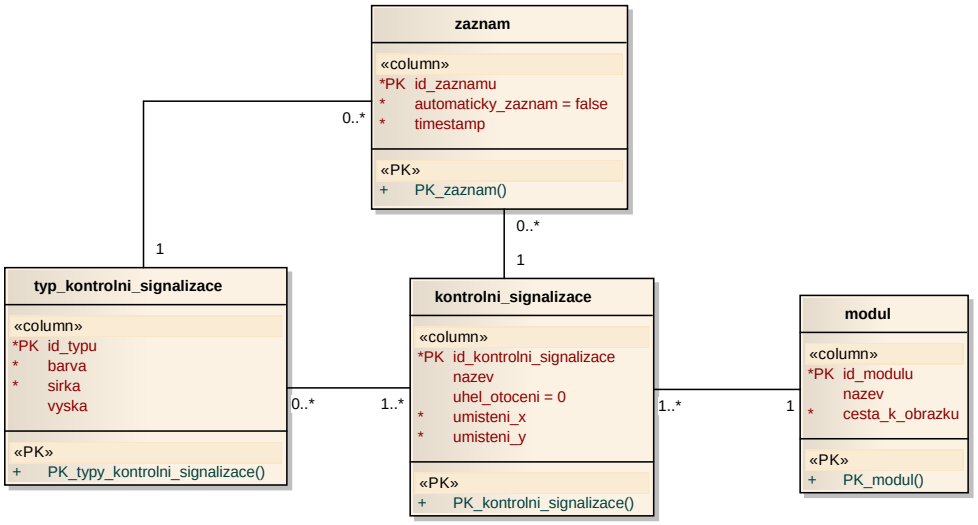
Produkční databáze se skládá ze čtyř entit a je zobrazena na obrázku 4.2. V první řadě se jedná o tabulku „modul". Tato tabulka povede informace o všech modulech, které budou diagnostikovány v rámci této aplikace. Vedle názvu, který bude zobrazován v klientské aplikaci, bude evidována také adresa k obrázku, která bude podkladem při zobrazování.

Každý modul má nějaké kontrolní signalizace. Ty budou uloženy v tabulce „kontrolni_signalizace." Pro orientaci bude možné ke každé kontrolní signalizaci přidat i její název, kterým bude jednoznačně identifikovatelná. S těmito atributy bude uloženo v databázi ještě umístění definované pomocí souřadnic a úhel otočení na zobrazované ploše. Tento atribut budou využívat především podlouhlé signalizace, proto je atribut nepovinný a výchozí hodnotu má nastavenou na 0.

Ke kontrolní signalizaci můžeme přidat množinu stavů, kterých může kontrolní signalizace nabývat.

To bude evidováno v tabulce „typ_kontrolni_signalizace." Dalšími atributy zde budou barva evidovaná v hexadecimálním zápisu a velikostní parametry.

Tabulky „kontrolni_signalizace" a „typ_kontrolni_signalizace" jsou propojeny s tabulkou „zaznam", do které budou zapisovány údaje získané z Dopravního sálu. Jak bylo uvedeno v požadavcích, periodicky se mají uložit všechny stavy v Dopravním sále. Tyto stavy jsem označil pomocí atributu a bodou tak jednoduše identifikovatelné. Tabulku „modul" není potřeba s tabulkou „zaznam" propojovat. Už samotná kontrolní signalizace totiž určuje,



Obrázek 4.2: Produkční databáze

o jaký modul se jedná.

4.1.2 Chybová databáze

Chybová databáze se bude skládat z jediné tabulky. Bude sloužit pro evidenci stavů, které program posoudil jako chybné. Do chybové databáze bude zapsán čas, kdy k chybě došlo. V tabulce také bude uvedena zpráva, proč aplikace posoudila stav jako nevyhovující.

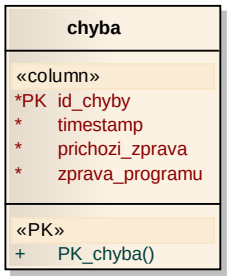
Otázkou zůstávalo, jakým způsobem evidovat chyby, ke kterým došlo. Data z Dopravního sálu nesou informaci o modulu, kontrolní signalizaci a jaký stav daná kontrolní signalizace má. Může se totiž stát, že data přijdou poškozená, pak je nebude možné jakkoliv propojit. Nejvýmluvnější informační hodnotu má příchozí zpráva sama o sobě. Tu jsem se nakonec jako jedinou rozhodl ukládat do tabulky.

Chyby jsem se rozhodl evidovat do databáze především díky lepším možnostem filtrování oproti ukládání logů do souboru. V případě ukládání do souboru by bylo nutné část zápisu synchronizovat.

4.2 Architektura serveru

Dále jsem vytvořil návrh serverové architektury. Celý systém by se dal rozdělit do několika drobných částí, které spolu navzájem komunikují. Tyto části následně popíšu. Návrh celého systému architektury je na obrázku 4.4.

Vzhledem ke všem činnostem, které server musí zvládnout vykonat, musí být aplikace vícevláknová.



Obrázek 4.3: Chybová databáze

4.2.1 Parametrizace serveru

První částí je startování serveru a jeho parametrizace. Dle požadavků zákazníka je potřeba parametrizovat dobu, za kterou se má zálohovat databáze s daty, a doba automatického uložení všech stavů do databáze. Dále je dobré parametrizovat port, na kterém má server naslouchat, a přístupové údaje do databáze.

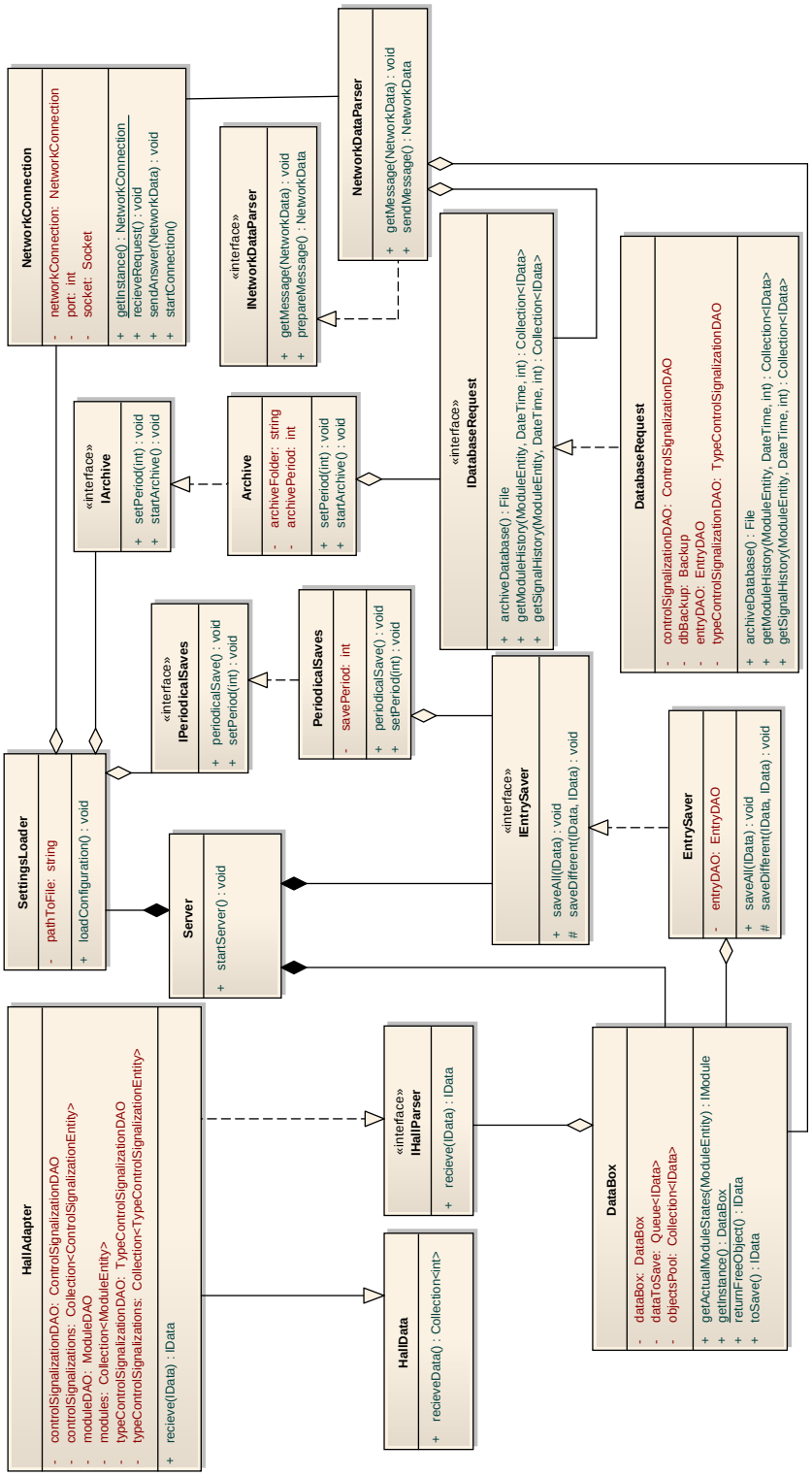
Všechny tyto údaje by měly být dostupné a snadno modifikovatelné. Špatným kandidátem je v tomto případě parametrizace ve zdrojovém kódu. Při úpravě přístupových údajů je možné omylem smazat nebo nevhodně modifikovat některou část kódu a dočasně tak znefunkčnit server. Druhou možností je přenést parametrizaci na klientské aplikace. Zde ovšem může docházet ke kolizím v nastavování z více uživatelských aplikací. Také je nutné serverovou aplikaci při změně parametrů restartovat a tím pádem mezi klientem a serverem skončí komunikace.

Jako nejlepšího kandidáta jsem vyhodnotil parametrizaci pomocí souboru. Soubor lze kdykoliv změnit a server restartovat až později ve vhodnou dobu.

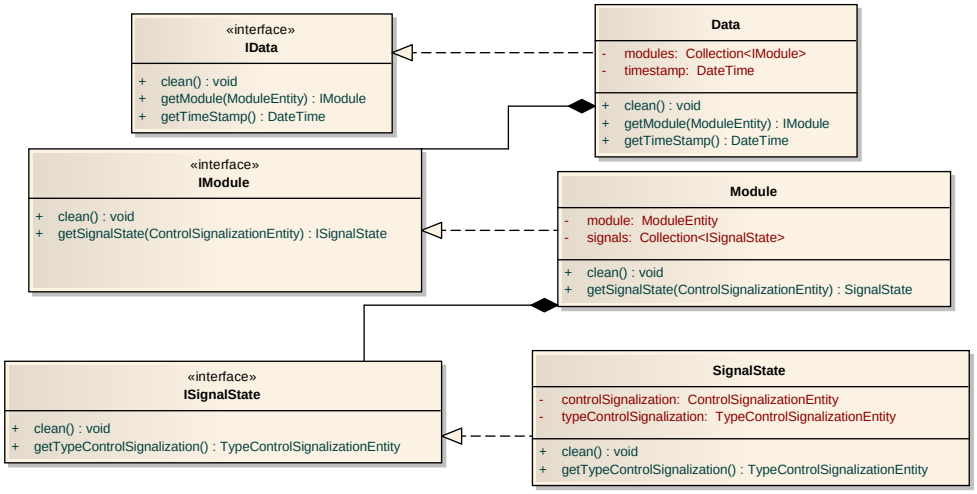
O parametrizaci se stará třída „SettingsLoader," která načte soubor a z něj příslušnými hodnotami posléze nastaví všechny dotčené třídy. Těmi jsou „PeriodicalSaves" pro periodické ukládání všech stavů z Dopravního sálu a „Archive" pro archivování celé databáze na disk. Dotčenou třídou bude i třída zajišťující síťové připojení „NetworkConnection", které je potřeba nastavit port, na kterém bude naslouchat.

4.2.2 Data z dopravního sálu

Další částí je sběr dat z Dopravního sálu. Zde bude jedno z kritických míst, neboť na server dorazí každých 100 ms nová data z diagnostikovaných zařízení. O toto se stará třída „HallParser", na kterou byl aplikován návrhový vzor adaptér, který jsem implementoval podle [28]. Adaptér mění rozhraní na to, které my potřebujeme pro náš program. To se tady ve skutečnosti děje. Data jsou poskytována v binární formě a mým cílem je data získat ve formě, která mi nejvíce vyhovuje.



Obrázek 4.4: Architektura serveru



Obrázek 4.5: Struktura uchovávání dat

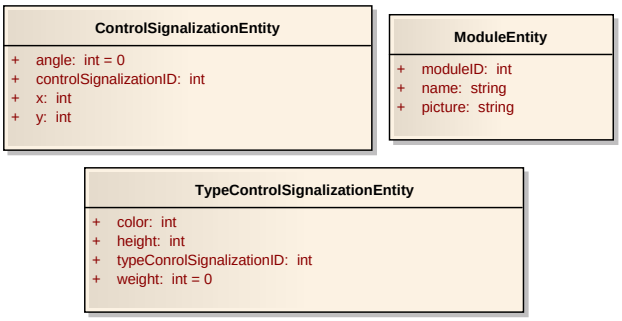
Zmíněná forma je stromová struktura, ve které jsou uloženy všechny údaje, které jsou dále potřeba pro zpracování. Stromová struktura je uvedena na obrázku 4.5.

4.2.3 Ukládání do paměti

Poté, co data budou přijata z diagnostikovaných zařízení a zpracována pomocí adaptéru, je potřeba je někam dočasně uložit do paměti, než přijde řada na jejich zpracování. O to se stará třída „DataBox". Tato třída využívá návrhového vzoru jedináček (singleton) implementovaného podle [28]. Jedináček zajišťuje, že v celém programu bude právě jedna instance. A to je přesně stav, kterého je potřeba v tomto programu dosáhnout. Třídu DataBox totiž volá více tříd. Pokud by si každá třída vytvořila svou vlastní instanci, byla by data rozstříštěná. Pomocí tohoto návrhového vzoru jsem schopen zajistit jedno místo, kde budou uloženy všechny zpracovávané jednotky.

Třída DataBox spolupracuje s adaptérem, od kterého získává data a ukládá je k sobě do paměti. Naopak data, která má tato třída uvnitř sebe uschovaná, poskytuje databázi a třídě, která se stará o připojení k síti. Ta má přístup k této třídě z důvodu poskytování aktuálních stavů modulů, které jsou využívány při online módu pro klientskou aplikaci.

Uvnitř této třídy jsem navrhl fond úschovných tříd. Vytváření objektů něco stojí a není důvod mazat instanci, kterou budeme o chvíli později potřebovat. Zvlášť v případě, kdy nová data dorazí každých 100 ms na server. Proto již použité úschovné třídy budou vyčištěny a použity znovu. Na druhé straně bude fronta objektů, které čekají na uložení do databáze. Pokud bude nedostatek úschovných tříd, budou vytvářeny nové.



Obrázek 4.6: Schránky pro udržování dat

4.2.4 Struktura uchovávání dat

Uchovávání dat nejlépe vystihnul strom ukázaný na obrázku 4.5. Třída „Data" uchovává data v určitém časovém okamžiku. Jako svoji vnitřní proměnnou nese časovou stopu, kdy byla data přijata. Tento čas se poté uvádí do databáze. Tato třída v sobě taktéž nese množinu tříd „Module," která reprezentuje jednotlivé moduly v Dopravním sále. Zde je vnitřní proměnnou schránka „ModuleEntity," která nese informace o konkrétním modulu. Za listy pak lze označit třídu „SignalState" nesoucí v sobě informaci o konkrétní kontrolní signalizaci a jejím stavu.

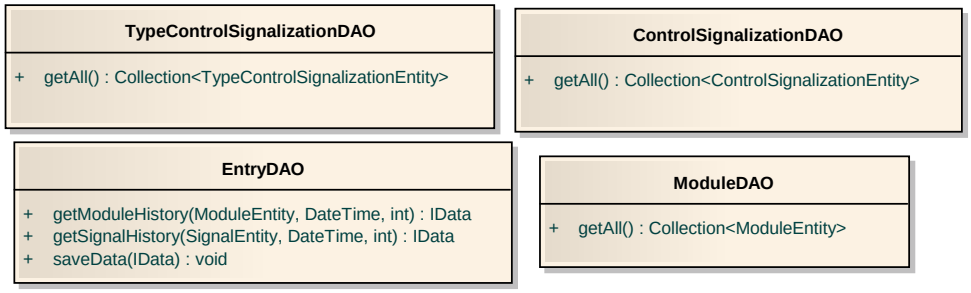
Uvnitř těchto tříd bude vždy uložena reference na „přepravku", která ponese všechny údaje z databáze. Přepravka je vzor popsaný v [27] a zde jsou ukázány na obrázku 4.6. Jde v podstatě o data načtená z databáze do tříd se stejnou strukturou. Přepravky mají konstantní atributy, a tudíž je nelze změnit.

4.2.5 Přístup do databáze

Přístup do databáze je umožněn pomocí DAO objektů. Přístup ke kontrolní signalizaci, signálům a modulům umožňuje jednoduchá třída s metodou, která vrátí všechny prvky v databázi. Formou výstupu je kolekce přepravek, které v sobě nesou informace uložené v databázi. Více není potřeba. Využití nachází v parsování dat přijatých z Dopravního sálu. Pomocí serveru ani pomocí aplikace totiž není možné tyto tabulky upravovat.

Zápis a čtení z databáze bude umožňovat třída „EntryDAO". Tato třída umožňuje zápis do databáze a dokáže také vrátit data podle konkrétního modulu a času. To samé pro stanovený signál.

Všechny třídy jsou ukázány na obrázku 4.7.



Obrázek 4.7: Objekty pro přístup do databáze

4.2.6 Připojení k intranetu

Server je potřeba připojit k síti. O to se stará třída „NetworkConnection". I zde jsem využil třídy návrhového vzoru jedináček. Tato třída totiž vyvolává, v případě přijetí požadavku, „NetworkDataParser" do separátního vlákna. Úkolem této třídy je zajistit dodání správných dat zpět klientovi. Po zjištění požadavků se třída rozhodne, zda se obrátit na databázi, pokud klient požaduje data z historie, nebo na data uložená v paměti, pokud se jedná o online režim.

Připojení k síti je zařízeno pomocí TCP protokolu. Je tedy zajištěno, že všechna data dorazí ke klientovi beze ztrát.

4.3 Architektura klientské aplikace

Klientská aplikace byla na návrh o něco jednodušší. Celkový návrh je vidět na obrázku 4.8. Aplikace, narozdíl od serveru, nemusí být vícevláknová.

Při startu klientské aplikace se načtou veškerá data o modulech, kontrolních signalizacích a stavech ze serveru do klientovy paměti. Tím je zachován požadavek na rozšiřitelnost, kdy klientské aplikace mohou procházet jen formální údržbou a není do nich potřeba větší mírou zasahovat. Aplikace bude muset mít přístup k podkladovým obrázkům. Ty budou nahrány přímo na klientské stanici.

Stejně jako u serveru, i zde bylo potřeba na jednom místě koncentrovat všechna přijatá data ze serveru. Proto jsem se rozhodl pro podobný princip ukládání. Data zde jsou opět uložena do stromu, jako je ukázáno na obrázku 4.5. Jednotlivé stromy jsou následně řazeny za sebou podle času, kdy byly přijaty na serveru.

Pokud klient potřebuje určitá data z historie, volá třídu „DataGetter". Ta funguje jako prostředník mezi lokálně uloženými daty a síťovým připojením. V první řadě se pokusí vyhledat data v lokálním prostředí. Pokud neuspěje, požádá o data server skrze síťové připojení. Klient tak nejen šetří čas sobě, ale ani zbytečně nevytěžuje server. Klientovi jsem tuto třídu přiřadil jako

prostředníka, na kterého přenáším zodpovědnost za rozhodování. Samozřejmě, pokud chce klient online data, nebude se zdržovat hledáním v lokálním úložišti a okamžitě odesílá požadavek o online přenos na server.

Ať už data byla získána jakýmkoliv způsobem, následně bude vyvolána třída Drawer, která se postará o vykreslení modulu a všech stavů.

4.4 Komunikace serveru a klienta

Posledním bodem k vyřešení je vzájemná komunikace serveru s klientem. Všechny klientské stanice i server jsou připojeny k intranetu, kterého je ideální využít. Nejlepším řešením je přenos pomocí protokolu TCP. Ten zaručí, že veškerá vyžádaná data od serveru dorazí ke klientovi v pořádku.

Otázkou tedy jen zůstává, jakým formátem mezi těmito uzly na síti data přenášet. Prakticky jiná možnost než serializace objektů mě nenapadla.

Všechny třídy, které budou přenášeny přes síť, musí být serializovatelné. To se týká objektů Data, Module, SignalState, NetworkData a potomků, TypeControlSignalizationEntity, ControlSignalizationEntity a ModuleEntity.

4.4.1 Formát dat

Důležité je zachovat jednotný formát přenosu dat, který bude čitelný pro obě strany. Tomu napomáhá třída NetworkData. V této třídě lze nalézt časovou stopu a řetězec reprezentující požadavek.

Od této třídy dále dědí další třídy. První je „NetworkInit". Ta se zabývá přenosem nejaktuálnějších dat ze serveru na klienta. Tato data si následně klient uloží do své paměti a později s nimi pracuje.

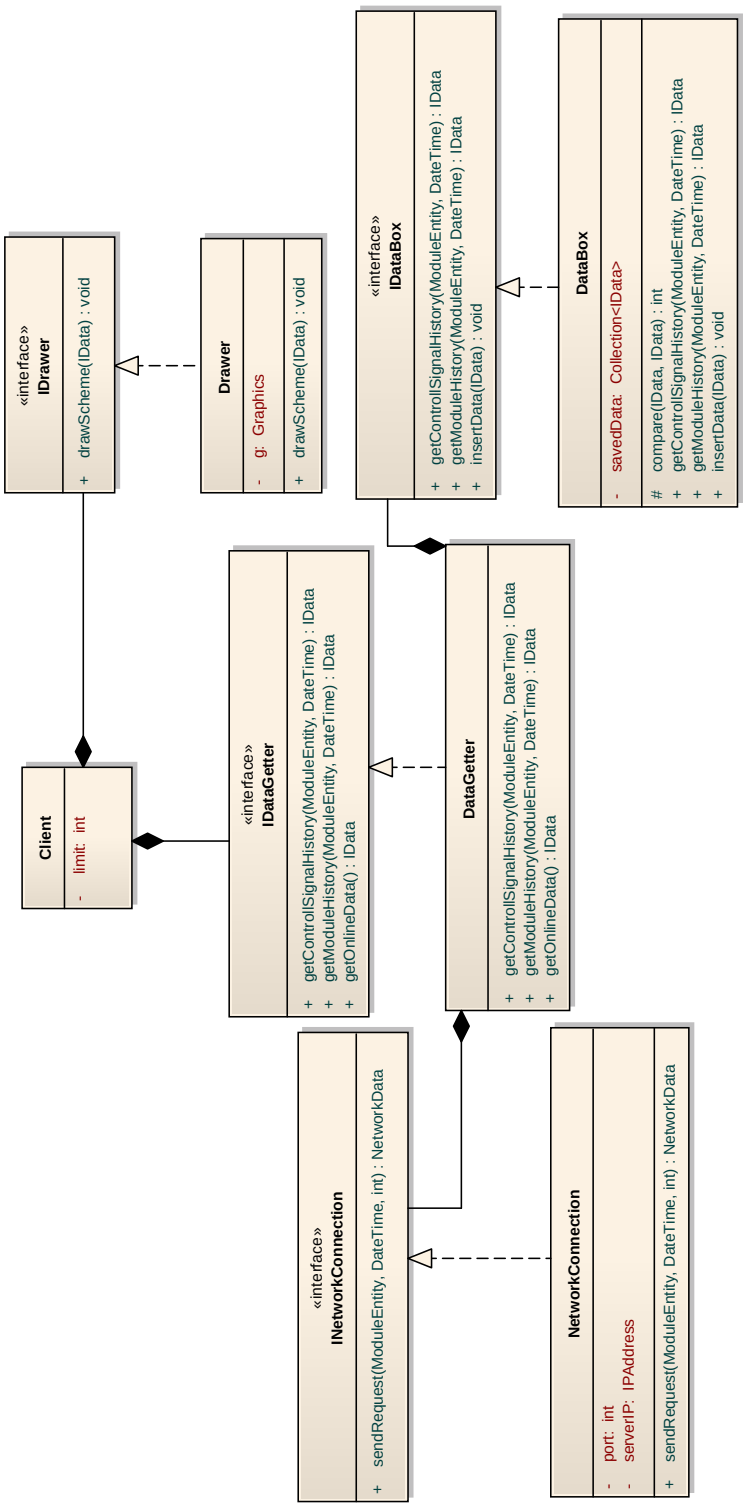
Ze strany klienta dochází zpráva ve formátu „NetworkRequest". Zde jsou obsaženy údaje o tom, který modul nebo stav je požadován, zda se jedná o online režim. V případě offline režimu se pak pracuje s počtem dat, která mají dorazit ke klientovi. Tento parametr je modifikovatelný v rámci klientské aplikace.

Odpověď ze serveru přichází jako „NetworkResponse". Nese v sobě počet dat, která byla stanovena předaným parametrem od klienta. V případě online módu zde bude jen jedna stromová struktura, která bude ihned zobrazena.

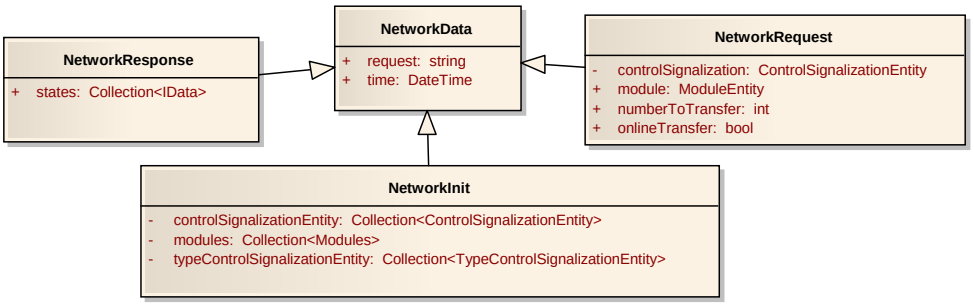
Všechny tyto třídy jsou znázorněny na obrázku 4.9.

4.5 Důraz na rozšiřitelnost

V rámci architektury bylo potřeba zamyslet se nad možností rozšiřitelnosti tohoto programu. Jinou možností, než jaká je zde uvedena, je zápis všech dat přímo do klientské aplikace. Tím by nebylo potřeba na začátku spuštění aplikace přenášet data ze serveru.



Obrázek 4.8: Architektura klientské aplikace



Obrázek 4.9: Třídy pro přenos dat

Jak už jsem ovšem zmiňoval výše, na pevno dané atributy do tříd ve zdrojovém kódu při úpravě znamenají potenciální riziko chyby. Rovněž je důležité mít neustále kompatibilní serverovou a klientskou část. A nakonec, při každé změně je nutné nejdříve aplikaci vytvořit a poté ji nahrát na všechny uživatelské stanice, kde má být diagnostický program k dispozici.

Mnou navržené řešení většinu těchto problémů odbourává. Do zdrojového kódu se v rámci rozšíření o další zobrazované moduly nebude muset zasahovat a klient se serverem je neustále synchronizován. Samozřejmě vytvoření načítání všech údajů z databáze a správné přenesení ke klientovi dá v rámci implementace více práce, ovšem do budoucna se jedná o ušetření práce neustálými aktualizacemi.

Kapitola 5

Testování

Poslední částí mé bakalářské práce bylo vytvoření funkčních a akceptačních testů. Pomocí testování zajišťujeme kvalitu dodávaných výstupů projektu, tedy že se náš program chová podle požadavků stanovených zákazníkem. Testování nám také může ušetřit nemalé finanční výdaje. Dřívější odhalení chyb totiž znamená nižší náklady [29].

Testování probíhá formou testovacích scénářů. Ty obsahují slovní popis testu a jednotlivé body, které je nutné pro vykonání testu provést. V závěru se poté uvádí očekávané reakce programu.

5.1 Fáze testování

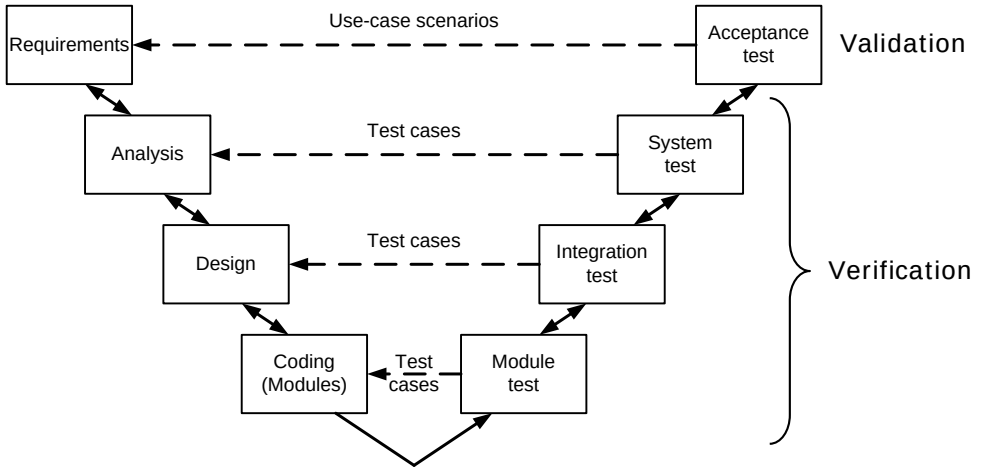
Testování softwaru se dělí na několik fází. Postupuje se od těch nejmenších jednotek v našem návrhu, tedy tříd a jednotlivých funkcí, až po testování systému jako celku. Tyto jednotlivé testy odpovídají úrovním specifikace. Vazby se graficky znázorňují pomocí V-modelu, který ze zobrazen na obrázku 5.1, a prakticky zachycuje celý životní cyklus software od požadavků po akceptaci produktu.

5.1.1 Unit testy

Unit testy jsou první fází v testování. Někdy unit testům může ještě předcházet testování programátorem. Princip spočívá v kontrole zdrojového kódu jiným programátorem než tím, který kód napsal [30].

U unit testů jsou testovány nejmenší funkční celky. V objektovém programování se jedná o jednotlivé třídy. Pokud je to možné, testy se provádějí bez závislosti na dalších komponentech systému.

Unit testy mohou být prováděny automatizovaně i manuálně a mají podobu programového kódu.



Obrázek 5.1: V-Model, převzato z [3]

5.1.2 Integrační testy

Další fází jsou testy integrační. Při těchto testech se testuje komunikace mezi komponenty systému nebo komunikace systému s externími komponenty, se kterými systém spolupracuje. Tyto testy se provádějí postupně — nejdříve je zkoumána vybraná dvojice komponentů a následně se k těmto komponentům přidávají další [30].

5.1.3 Systémové testy

Během systémových testů se ověřuje celková funkcionalita systému z pohledu zákazníka. Zkoumají se funkční i nefunkční požadavky na systém. Po dokončení této fáze je systém pro zákazníka do produkčního prostředí.

5.1.4 Akceptační testy

Akceptační testy se od systémových prakticky neliší. Hlavní rozdíl spočívá v místě testování a testerech, kteří patří zákazníkovi [31]. Akceptační testy probíhají v prostředí u zákazníka, většinou v nějakém testovacím prostředí.

Všechny chyby nalezené týmem testerů zákazníka jsou reportovány zpět vývojovému týmu. Je proto vhodné domluvit se na formě předávání informací o chybách.

5.2 Funkční testy

Ve funkčních testech se provádí kontrola tak, že aplikace obsahuje všechny zákazníkem vydefinované požadavky. Testují se všechny funkce systému a

Název scénáře	Online zobrazování
Popis scénáře	Testování, zda se do jedné sekundy od provedené změny projeví změna i v diagnostickém programu
Kroky	• Zapněte diagnostický program a vyberte online zobrazování • Vyberte modul, na kterém budete provádět změnu stavu • V diagnostickém programu vyberte tentýž modul • Proveďte změnu na modulu v Dopravním sále • Sledujte změnu v diagnostickém programu • Kontrolujte aktuálnost času v okně aplikace
Očekávaný vysledek	Nejpozději jednu sekundu po změně diagnostický program hlásí změnu stavu u sledovaného modulu

Tabulka 5.1: Test 1

Název scénáře	Procházení historie
Popis scénáře	Procházení historie vybraného modulu a ve vybraném čase
Kroky	• Přepněte aplikaci do offline módu — sledování historie • Vyberte čas, vůči kterému chcete začínat procházet historii • Počkejte na příchozí data • Pohybujte se v historii a pokud je to možné, porovnávejte se skutečností
Očekávaný vysledek	Diagnostický program ukazuje data, která jsou shodná se skutečností v příslušném čase

Tabulka 5.2: Test 2

správná odezva. Během funkčních testů je také potřeba brát v úvahu všechny související situace, které s používáním mohou nastat, a řádně je otestovat. Funkčního testování se nejčastěji využívá v na integrační, systémové a akceptační úrovni testování. [32, 33]

5.3 Návrhy testů

V poslední části uvádím navržené testy. Testy jsou v tabulkách 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 a 5.8.

Název scénáře	Spolupráce s třídou z Dopravního sálu
Popis scénáře	Kontrola serveru, zda správně komunikuje s diagnostikovanými zařízeními
Kroky	• Spusťte server • Proveďte změnu na libovolném modulu v Dopravním sále • Sledujte databázi na serveru
Očekávaný vysledek	Server správně rozeznal signál, kontrolní signalizaci i modul a vše správně zaznamenal do databáze

Tabulka 5.3: Test 3

Název scénáře	Detekce chyby
Popis scénáře	Server detekuje špatné vstupy, které neodpovídají datům z dopravního sálu
Kroky	• Spusťte server • Snažte se měnit mnoho stavů ve stejnou dobu • Simulujte nevalidní data pomocí simulátoru • Sledujte chybovou databázi
Očekávaný vysledek	V chybové databázi se objeví nerozeznaný výsledek včetně času a dat v binární podobě

Tabulka 5.4: Test 4

Název scénáře	Připojení z více klientských stanic
Popis scénáře	Uživatelé Diagnostického programu se připojí na dvou a více počítačích
Kroky	• Spusťte diagnostický program na dvou nebo více pracovních stanicích zároveň • Aktivujte na obou zařízeních online režim • Na každém zařízení vyberte jiný zobrazovaný modul • Měňte stavy na diagnostikovaných zařízeních a sledujte, zda diagnostický program odpovídá realitě • Změňte u jednoho zařízení režim na offline • Listujte v historii
Očekávaný vysledek	Server odpovídá ve stanoveném čase 1 sekudna v případě online režimu a podává reálná data i pro listování v historii

Tabulka 5.5: Test 5

Název scénáře	Zálohování
Popis scénáře	Server automaticky zálohuje veškerá data uložená v databázi
Kroky	• Nakonfigurujte serveru testovací čas zálohování dat a složku pro uložení dat • Spusťte server • Změřte čas mezi jednotlivými zálohami databáze • Porovnejte změřený čas s nastaveným parametrem • Zkontrolujte, že zálohy databáze jsou uloženy na správném místě
Očekávaný vysledek	Server provedl všechny činnosti včas a soubory uložil na parametrem dané místo

Tabulka 5.6: Test 6

Název scénáře	Akceptace parametrů pro server
Popis scénáře	Server akceptuje parametry zadané v konfiguračním souboru
Kroky	• Nastavte všechny dostupné parametry na rozumnou délku pro testování • Připojte se na nastavený port serveru • Sledujte chování serveru a činností • Změřte délku prodlevy před činností a porovnejte s nastavenými parametry
Očekávaný vysledek	Server vykonává všechny činnosti ve stanovený čas a správně

Tabulka 5.7: Test 7

Název scénáře	Automatické zálohování všech stavů
Popis scénáře	Server zálohuje všechny stavy v pravidelných intervalech daných parametrem
Kroky	• Nastavte parametr pro ukládání všech stavů do databáze • Spusťte server • Po vykonání činnosti zkontrolujte databázi serveru
Očekávaný vysledek	Server vykonává činnost správně, včas a všechny dotčené stavy správně označí jako automaticky uložené

Tabulka 5.8: Test 8

Závěr

V této bakalářské práci jsem se zabýval jako projektový manažer plánováním projektu pro Dopravní sál Fakulty dopravní ČVUT. Dále jsem se zabýval návrhem diagnostického programu a následně i návrhem akceptačních testů, které mohou být použity po implementaci.

Během analýzy jsem si v Dopravním sále vyzkoušel různá zařízení, která jsou v něm k dispozici. Rovněž jsem si své znalosti rozšířil o funkčnost železniční dopravy a jejího zabezpečení.

Mým výstupem bylo vytvoření projektové dokumentace, kde jsem si vyzkoušel některé techniky projektového řízení a uplatnil je v rámci tohoto projektu. Tato část mě utvrdila v tom, že před vytvořením každého projektu je více než dobré všechny činnosti naplánovat a využívat již osvědčených postupů a nevymýšlet si své vlastní. Identifikoval jsem rizika, vytvořil harmonogram s ohledem na skutečný vývoj a stanovil celkové náklady na projekt. Naučil jsem se také techniky, o kterých jsem buď jen slyšel, nebo vůbec nevěděl. Po nastudování této problematiky si ovšem myslím, že jsem vytvořil adekvátní projektovou dokumentaci, podle které lze projekt vést.

Druhým výstupem bylo navrhnutí architektury diagnostického programu. Tu jsem navrhl jako architekturu klient–server. Server v této architektuře slouží jako sběrač dat a poskytovatel dat klientům. Klient zde figuruje jako vizualizátor vybraných dat na uživatelských stanicích. Během návrhu jsem kladl důraz na co nejjednodušší rozšiřitelnost.

Mým posledním výstupem byly akceptační a funkční testy. Tyto testy budou využity při předávání výstupu zákazníkovi a pomocí nich bude ověřena funkčnost celého diagnostického programu, a to jak klientské aplikace, tak i serveru.

Veškerých stanovených cílů v zadání tedy bylo dosaženo. Dopravní sál Fakulty dopravní byl analyzován a byla vytvořena projektová dokumentace k Diagnostickému programu. Architektura systému byla navržena tak, aby splňovala požadavky na přístup z více klientských stanic a rozšiřitelnost. Nakonec byly navrženy funkční a akceptační testy.

Další pokračování tohoto projektu je jasné — podle harmonogramu následuje implementace navrženého systému, kde si zřejmě vyzkouším roli programátora. V tuto chvíli je projekt přerušen a pokračovat bude podle harmonogramu začátkem července 2015.

Literatura

[1] pixgood.com: *Pmp Project Management Life Cycle [online]*. [cit. 2015-05-10]. Dostupné z: http://pixgood.com/pmp-project-management-life-cycle.html

[2] BOZPInfo: *Příklad stanovení kategorií přijatelnosti rizik prostřednictvím matice rizik [online]*. [cit. 2015-04-28]. Dostupné z: http://www.bozpinfo.cz/josra/josra-04-2008/horehledova_komplex.html

[3] Hahman, T.: Software Development Process Models [online]. [cit. 2015-04-29]. Dostupné z: http://www.cs.toronto.edu/~torsten/THahmann_CSC444_Tutorial1_SWDevProcesses.pdf

[4] Driessen, V.: A successful Git branching model [online]. [cit. 2015-03-27]. Dostupné z: http://nvie.com/posts/a-successful-git-branching-model/

[5] Doležal, J.; Máchal, P.; Lacko, B.; aj.: *Projektový management podle IPMA*. Grada Publishing, a.s., druhé vydání, 2012, ISBN 9788024742755.

[6] Kučerová, H.: Specifikace požadavků na informační systém [online]. [cit. 2015-03-18]. Dostupné z: http://info.sks.cz/users/ku/PRI/specifik.htm

[7] Enterprise Architect [počítačový program]. Program pro modelování a design software. Dostupné z: http://www.sparxsystems.com.au/

[8] Monson-Haefel, R.: *97 klíčových znalostí softwarového architekta*. Brno: Computer Press, 2010, ISBN 9788025133132.

[9] Wiles, F.: Three things you should never put in your database [online]. [cit. 2015-03-19]. Dostupné z: http://www.revsys.com/blog/2012/may/01/three-things-you-should-never-put-your-database/

[10] iplotz.com [webová aplikace]. Program pro vytváření wireframů. Dostupné z: https://iplotz.com/app/

[11] Prokopec; Houfek; Veselá; aj.: *Lokální diagnostický systém LDS-3*. AŽD Praha s.r.o., ředitelství společnosti — technický úsek, 2012.

[12] RVP.cz: *Řízení projektů — projektový management*. [cit. 2015-04-18]. Dostupné z: http://clanky.rvp.cz/wp-content/upload/prilohy/2698/rizeni_projektu.pdf

[13] Bendová, K.; Nechvílová, S.; Seitl, M.; aj.: Základy projektového řízení [online]. [cit. 2015-04-19]. Dostupné z: http://www.ff.upol.cz/fileadmin/user_upload/FF-katedry/psychologie/publikace/Bendova/Bendova_K_a_kol_zaklady_projektoveho_rizeni.pdf

[14] Skalický, J.; Jermář, M.; Svoboda, J.: *Projektový management a potřebné kompetence*. Západočeská univerzita v Plzni, Vydavatelství, 2010, ISBN 9788070439753.

[15] Pakosta, J.: Obecné principy řízení projektů [online]. Dostupné z: http://www.ccvj.cz/UserFiles/File/euprolek/M4/obecne-principy-rizeni-projektu-cast-1-studijni-text.pdf

[16] Hinde, D.: *Prince2: Study guide*. Chichester: Wiley, 2012, ISBN 9781119970781.

[17] Duncan, W.: *A Guide To The Project Management Body Of Knowledge [online]*. [cit. 2015-04-21]. Dostupné z: http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/reporty/pmbok.pdf

[18] Projekt manažer: *Identifikační listina projektu [online]*. [cit. 2015-04-23]. Dostupné z: http://www.projektmanazer.cz/sites/default/files/dokumenty/1-5identifikacnilistinaprojektu.pdf

[19] Management Mania: *Projektový plán [online]*. [cit. 2015-03-11]. Dostupné z: https://managementmania.com/cs/plan-projektu

[20] Šafář, P.: *IT podpora projektového řízení [online]*. Bakalářská práce, Masarykova univerzita, Ekonomicko-správní fakulta, 2009, [cit. 2015-04-26].

[21] Projekt manažer: *Work breakdown structure [online]*. [cit. 2015-03-11]. Dostupné z: http://www.projektmanazer.cz/sites/default/files/dokumenty/2-1wbs.pdf

[22] Friebelová, J.: *Síťová analýza [online]*. [cit. 2015-04-19]. Dostupné z: http://www2.ef.jcu.cz/~jfrieb/rmp/data/teorie_oa/SITOVA%20ANALYZA.pdf

[23] Management Mania: *Matice odpovědností [online]*. [cit. 2015-04-26]. Dostupné z: https://managementmania.com/cs/matice-odpovednosti

[24] Projekt manažer: *Změnový požadavek [online]*. [cit. 2015-04-28]. Dostupné z: http://www.projektmanazer.cz/sites/default/files/dokumenty/3-6zmenovypozadavek.pdf

[25] Coggle.it [webová aplikace]. Program pro vytváření myšlenkových map a strukturovaných grafů. Dostupné z: https://coggle.it

[26] ProjectLibre [počítačový program]. Program pro vytváření harmonogramu projektů a Ganttových diagramů. Dostupné z: http://www.projectlibre.org/

[27] Pecinovský, R.: *Návrhové vzory*. Brno: Computer Press, první vydání, 2007, ISBN 9788025115824.

[28] Bishopová, J.: *C#: Návrhové vzory*. Brno: Zoner Press, první vydání, 2010, ISBN 9788074130762.

[29] Čermák, M.: Testování SW [online]. [cit. 2015-04-29]. Dostupné z: http://www.cleverandsmart.cz/testovani-sw/

[30] Hlava, T.: Fáze a úrovně provádění testů [online]. [cit. 2015-04-29]. Dostupné z: http://testovanisoftwaru.cz/druhy-typy-a-kategorie-testu/faze-testu/

[31] Míka, T.: *Návrh nového přístupu k řízení kvality při vývoji software [online]*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, 2013, [cit. 2015-04-29].

[32] SW Testování: *Funkční vs nefunkční testování [online]*. [cit. 2015-04-28]. Dostupné z: http://www.swtestovani.cz/index.php/uvod-do-testovani/22-funkni-vs-nefunkni-testovani

[33] Hlava, T.: Funkční a nefunkční testy [online]. [cit. 2015-04-29]. Dostupné z: http://testovanisoftwaru.cz/druhy-typy-a-kategorie-testu/funkcni-a-nefunkcni-testy/

Příloha **A**

Seznam použitých zkratek

DAO Data Access Object

DLA Diagnostic Local Access — lokální přístupový počítač

DLS Diagnostický Lokální Server

DSFD Dopravní sál Fakulty dopravní

FD ČVUT Fakulta dopravní — Českе vysoké učení technické

GUI Graphical user interface

ILP Identifikační (zakládací) listina projektu

MD Man day — člověkoden

OS Operační systém

PERT Program evaluation and review technique

PMI Project Management Institute

SW Software

TCP Transmission Control Protocol

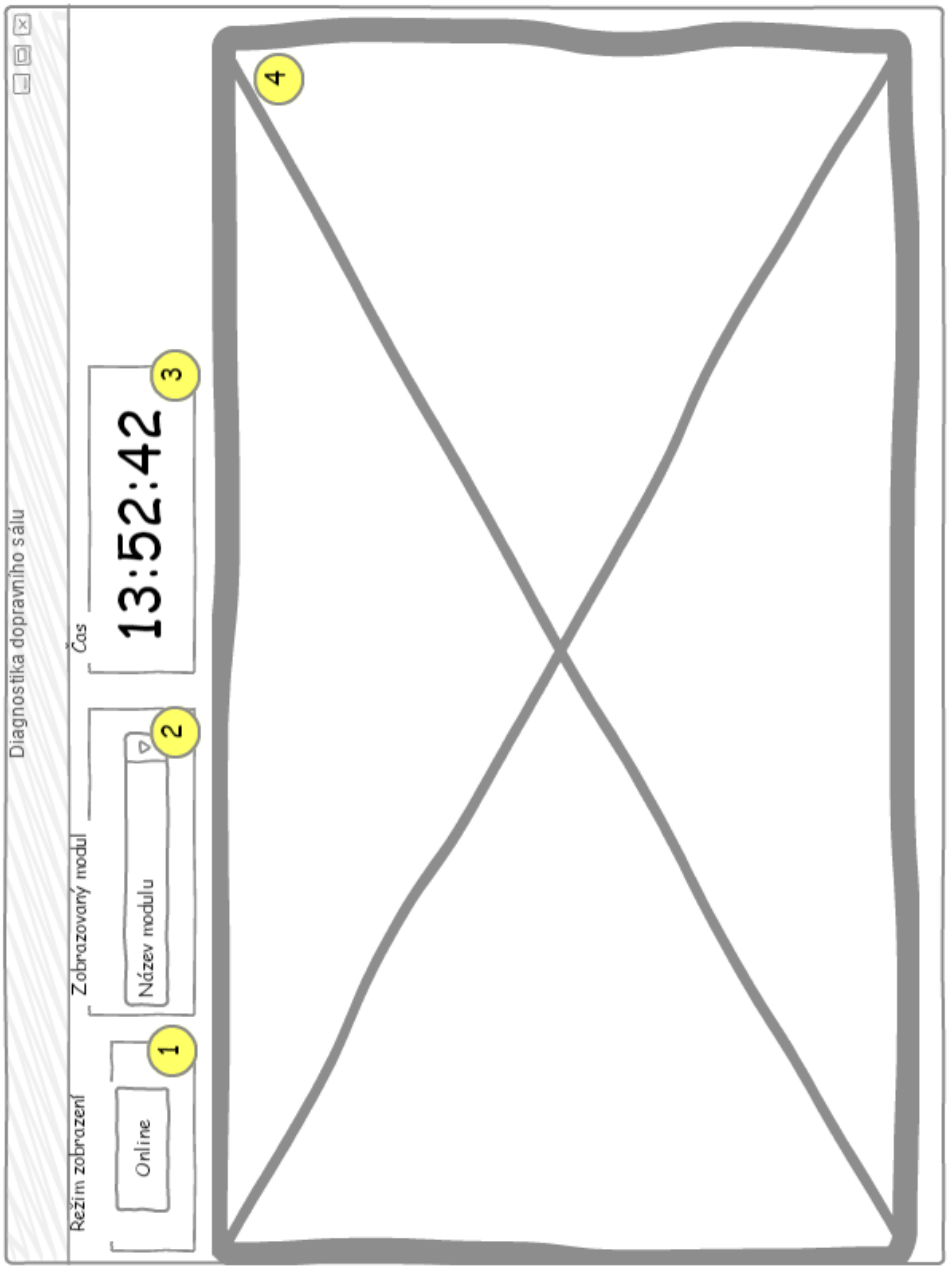
USA Spojené státy Americké

WBS Work breakdown structure

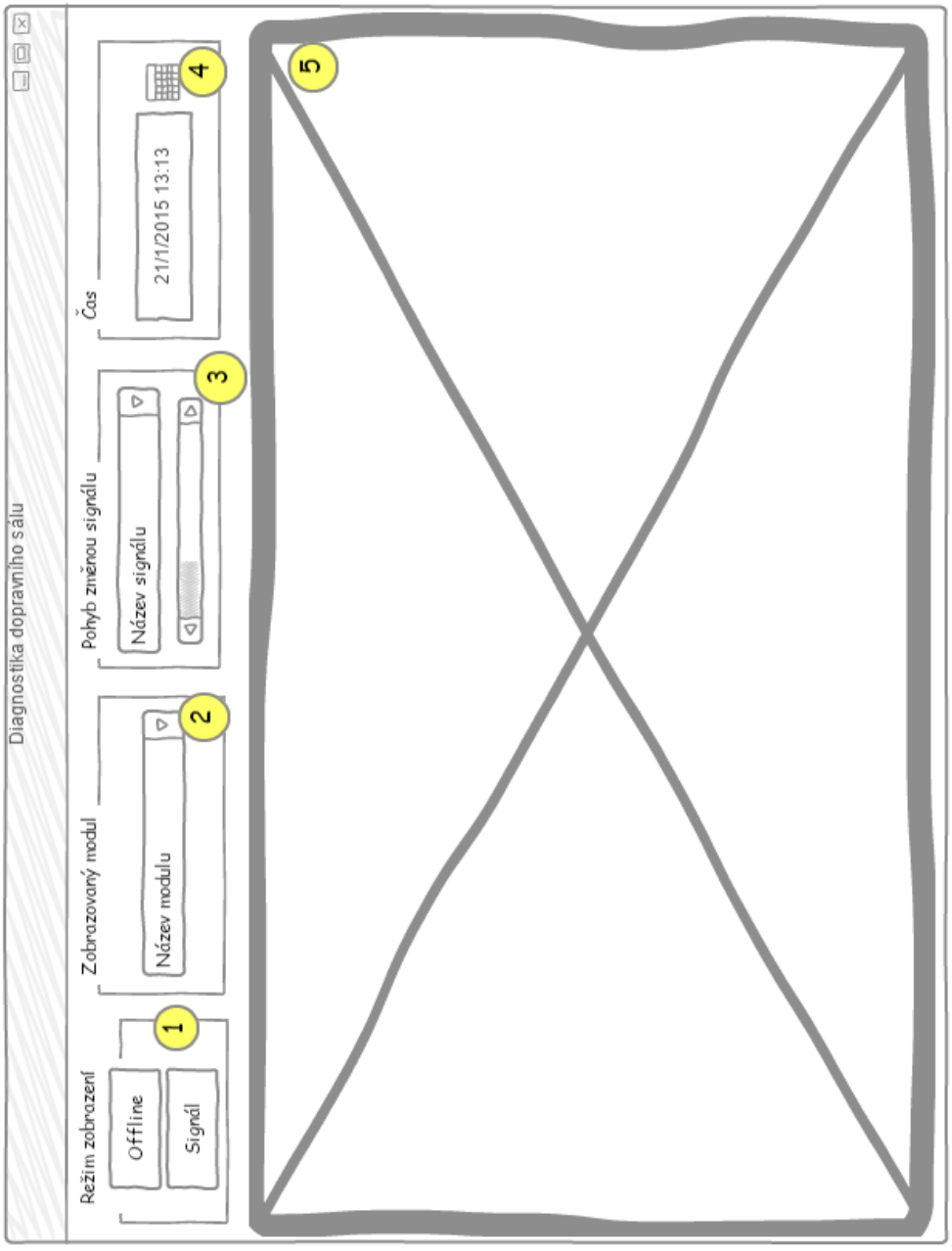
Příloha B

Wireframy aplikace

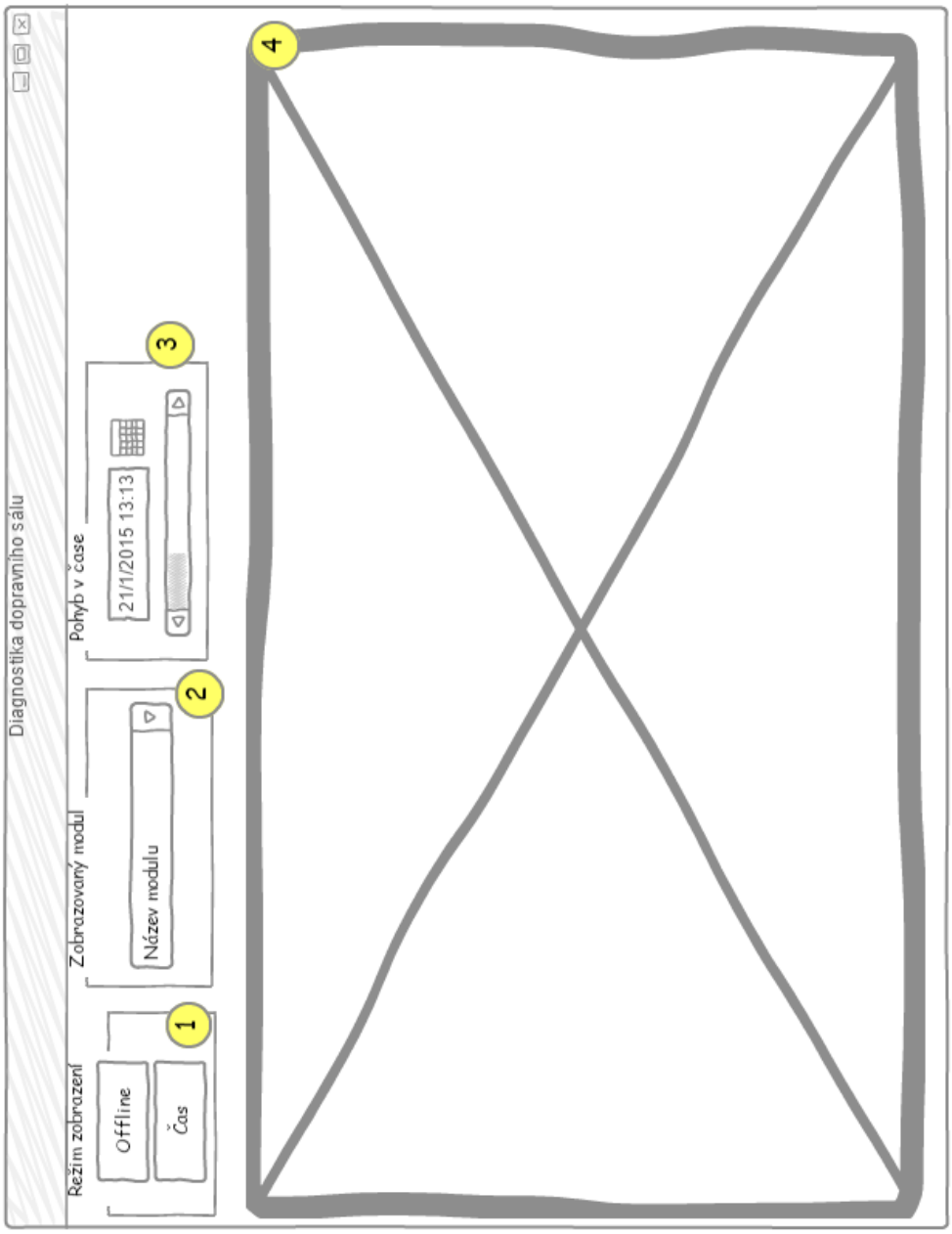
V této příloze jsou k dispozici wireframy aplikace, které byly popsány v části 1.6.



Obrázek B.1: Vizualizace při online módu



Obrázek B.2: Vizualizace sledování konkrétního signálu při offline módu



Obrázek B.3: Vizualizace sledování konkrétního modulu při offline módu

Příloha C

Projektová dokumentace

C.1 Úvod

Data jsou v dnešní době jednou z klíčových součástí systému. Většinou pomocí softwaru můžememe data lépe přetvářet k obrazu svému a následně je poté lépe číst. To bude umožňovat i diagnostický program pro Dopravní sál Fakulty dopravní ČVUT.

Tento projekt si klade za cíl vytvoření softwarového produktu pro zaměstnance Dopravního sálu a to konkrétně diagnostického programu, díky němuž budou moci zaměstnanci pracovat efektivněji.

Jedná se o systém archivování dat a jejich následné zpětné prezentace v grafické podobě pomocí různých filtračních možností. Archivovaná data jsou poskytována ze systémů nacházejících se v Dopravním sále, se kterými server musí umět spolupracovat a získávat z nich data.

Je nutné, aby šel systém spustit na více pracovních stanicích zároveň, byl spolehlivý a především rozšiřitelný — aby jej bylo možné do budoucna rozšiřovat o další moduly.

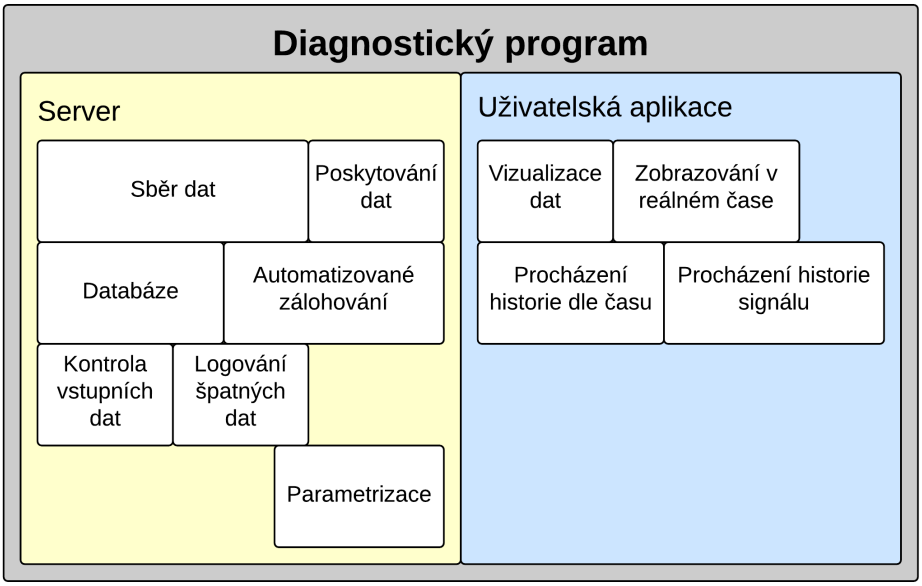
Vedle sběru dat bylo navrhnuto i zaznamenávání špatných vstupních dat, která mohou být způsobena například hardwarovou chybou.

Požadavkem zákazníka je začít se sběrem dat co nejdříve.

C.2 Fáze projektu

Celý projekt byl po analýze projektu rozdělen na několik na sebe navzájem navazujících částí. Tzv. „vodopád", který je ideální pro menší projekty a zákazníky, kteří vědí, co požadují. Znázornění návaznosti jednotlivých etap projektu je ukázáno na obrázku C.2.

V první řadě proběhne návrh celého systému s ohledem na všechny požadavky zákazníka. Na to následně naváže implementace celku, která je rozdělena na dvě části. Poté proběhne testování, od testování nejmenších jednotek systému až po uživatelské, a pokud systém obstojí, přejde se k nasazení.



Obrázek C.1: Grafické znázornění systému

Celý projekt bude pro přehled všech zůčastněných monitorován a organizován pomocí nástroje Trello, kde bude přehledně vidět, na kterých částech projektu se momentálně pracuje, kdy je jejich plánovaný konec a kdo je za daný úkol zodpovědný.

C.2.1 Analýza

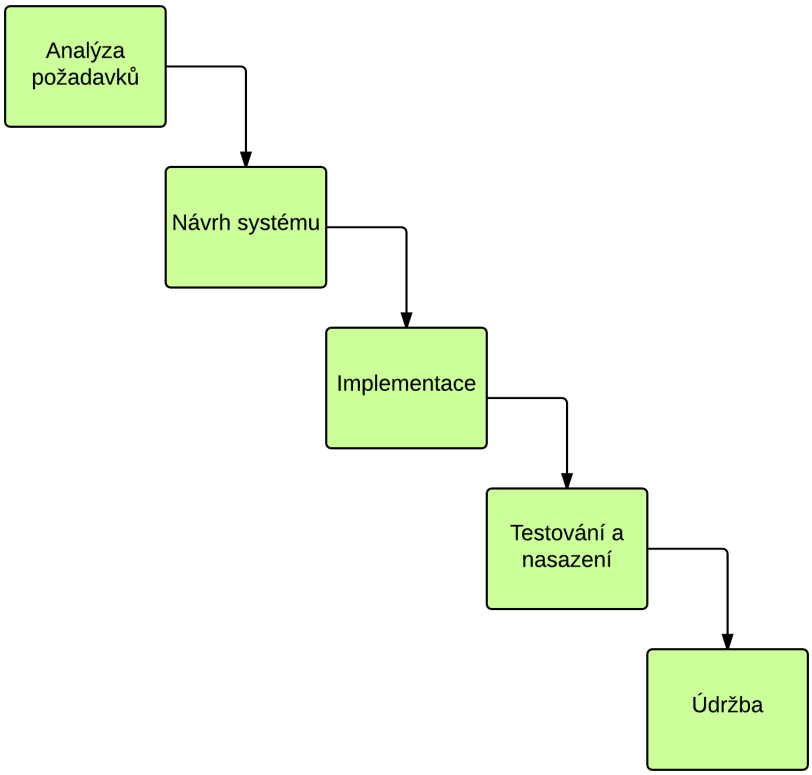
V prvotní fázi je potřeba zjistit, co zákazníka trápí a sestavit společně seznam požadavků, které si zákazník na systém klade. V rámci schůzek se zákazníkem a zjišťování potřeb, byly sestaveny i základní návhry obrazovek uživatelského systému.

Rozdělení na komponenty a jejich hlavní funkčnosti je uvedeno na obrázku C.1.

C.2.2 Návrh systému

V tomto kroku budou v zásadě navrženy dva modely — architektura aplikace a databázový model. V první řade bude navržen databázový model a to tak, aby dokázal uložit všechna data požadovaná zákazníkem.

Následně na to bude navržena serverová část aplikace, která bude mít přímý přístup do databáze. V tomto stavu bude také vyspecifikováno, jak data budou přenášena po síti k uživateli (v jakém tvaru).



Obrázek C.2: Jednotlivé fáze projektu a jejich návaznost

Dále bude navržena uživatelská aplikace, kde bude potřeba zohlednit jak uživatelské rozhraní, na kterém jsme se spolu se zákazníkem dohodli, tak i omezení pro zobrazování dat v reálném čase.

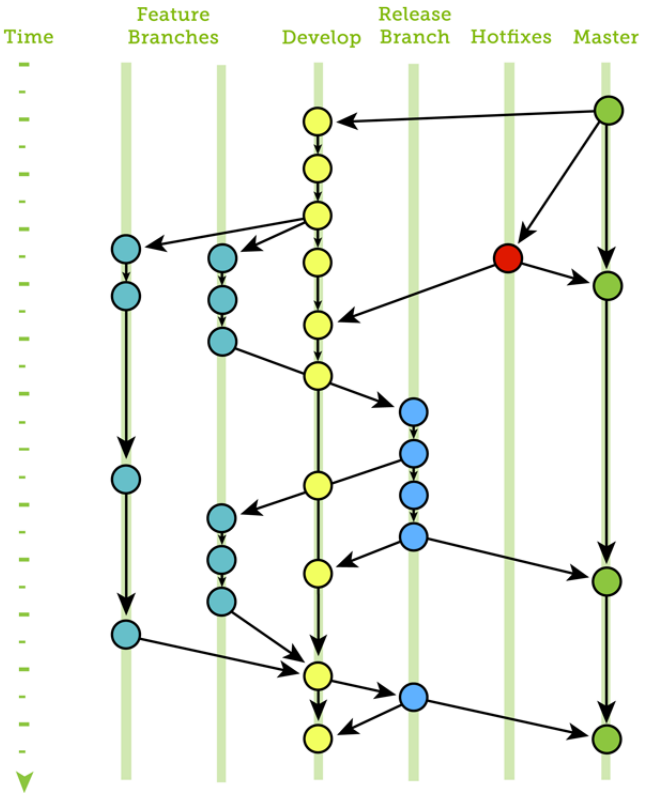
Nakonec budou navrženy testy pro ověření správné funkčnosti aplikace.

C.2.3 Implementace

Implementační fáze bude rozdělena stejně jako při návrhu na tři části. Veškeré zdrojové kódy budou pod správou verzovacího software git. Zároveň s tím bude ve verzovacím systému vytvořeno několik větví, jak je uvedeno na obrázku C.3, čímž se docílí přehlednějšího vývoje i lepší zpětné detekce chyb.

Během implementace bude za potřebí programovací jazyk C#, který se využívá v celém sále a je v něm napsána i aplikace pro sběr dat z diagnostikovaných zařízení. Dále bude využito frameworku .NET a vývojového prostředí Microsoft Visual Studio.

Veškerý vývoj se bude odehrávat ve větvi Develop, kde v posledním commitu této větve bude nejhůře nestabilní verze. Jednotlivé komponenty systému



Obrázek C.3: Větve ve verzovacím systému, převzato z [4]

se budou vyvíjet ve Feature větvích, které budou označeny prefixem „f-". Tyto větve se budou vždy slučovat do větve Develop.

Větev Release, která symbolizuje stabilní verzi k otestování, bude slučována s větví Develop a budou se zde řešit i případné opravy při nalezených chybách. Po otestování a schválení bude konkrétní release sloučena do větve Master, která bude shodná s verzí nahranou v produkčním prostředí.

V případě závažných chyb bude vytvořen Hotfix, který se bude vyvíjet ve stejně pojmenované větvi. Po opětovném nasazení do větve Master bude konkrétní hotfix nahrán i do vývojářské větvě Develop [4].

C.2.3.1 První část

V rámci této release bude vytvořen server a databáze pro jeden konkrétní modul. Cílem této release bude především sběr dat, jejich následná analýza a odhalování chyb s plynoucím časem — tedy praktické zaměření na sběr dat.

C.2.3.2 Druhá část

Druhá release se bude zabývat implementací serveru jako celku a bude de facto rozšířením první release. Zaměřena bude především na komunikaci server — klient. Tato část bude zaměřena především na vytvoření správné funkčnosti a správné synchronizace této vícevláknové aplikace.

C.2.3.3 Třetí část

Součástí třetí release bude vytvoření uživatelské aplikace a její napojení na serverovou část. Aplikace bude schopna připojit se k serveru a získávat vizualizovaná data.

C.2.4 Testování a nasazení

Po dokončení implementace přijde na řadu testování. To lze opět rozdělit na několik částí.

V první řadě půjde o tzv. Unit testy, kdy budou otestovány samostatné jednotky systému. Druhou fází budou integrační testy, které se zaměří na správnost zpracovávání informací z jednotlivých modulů v dopravním sále, a dále i komunikace s uživatelskou aplikací.

Nakonec se dostanou na řadu akceptační testy, které budou probíhat v produkčním prostředí přímo u zákazníka.

Po splnění těchto testů bude systém finálně nasazen a předán zákazníkovi do vlastnictví.

C.2.5 Údržba a rozšiřování

Po úspěšném nasazení přechází systém do stavu údržby. V tomto stavu bude systém procházet drobnými změnami a rozšířením o další vizualizovaná data.

Jelikož zákazník dopředu požadoval rozšiřitelnost o další moduly, je velmi pravděpodobné, že s příchodem dalších systémů do Dopravního sálu bude potřeba systém modifikovat a to samozřejmě co nejjednodušším způsobem. Tato část již není součástí projektu.

C.3 Rizika projektu

Součástí každého projektu jsou i rizika. Po analýze požadavků byla identifikována i možná rizika projektu, která mohou nastat.

Každé riziko má svůj faktor a jednoznačný identifikátor. Ke každému riziku byla stanovena pravděpodobnost výskytu, případný dopad na celý projekt a možnosti, které se nabízí, aby riziko nenastalo.

Rizika jsou uvedena v tabulkách C.1, C.2, C.3, C.4, C.5, C.6 a C.7.

Rizikový faktor	Vytíženost zadavatele
Stav	Potenciální
Pravděpodobnost výskytu	10%
Dopad	Zpoždění etapy, případně celého projektu
Mitigace rizika	Zvolení dostatečné časové rezervy Dostatečně dopředu domluvené schůzky s klientem
Krizový plán	Posunutí termínu nasazení

Tabulka C.1: Popis rizika R01

Rizikový faktor	Dovolená členů projektového týmu během prázdnin
Stav	Potenciální
Pravděpodobnost výskytu	20%
Dopad	Zpoždění etapy, případně celého projektu
Mitigace rizika	Stanovení časového harmonogramu s ohledem na členy týmu Záložní pracovní zdroje
Krizový plán	Posunutí termínu nasazení Outsourcing

Tabulka C.2: Popis rizika R02

Rizikový faktor	Požadavek na novou funkčnost v době realizace
Stav	Potenciální
Pravděpodobnost výskytu	10%
Dopad	Zdržení projektu, oddálení termínu nasazení
Mitigace rizika	Pečlivá analýza požadavků klienta Odmítnutí změnového požadavku projektovým týmem
Krizový plán	Úprava návrhu systému podle nových požadavků Předčasné ukončení projektu

Tabulka C.3: Popis rizika R03

Rizikový faktor	Projevení nezkušenosti projektového týmu
Stav	Potenciální
Pravděpodobnost výskytu	30%
Dopad	Zpoždění nebo prodražení projektu
Mitigace rizika	Konzultace se zkušeným projektovým manažerem, zvýšené monitorování
Krizový plán	Doplnění projektového týmu o zkušenější pracovní zdroje

Tabulka C.4: Popis rizika R04

Rizikový faktor	Zpoždění dodávky systému
Stav	Potenciální
Pravděpodobnost výskytu	20%
Dopad	Zaměstnanci budou moci používat systém později, než bylo smluveno
Mitigace rizika	Stanovení reálného termínu nasazení Sledování stavu projektu v průběhu realizace
Krizový plán	Oznámení zadavateli, že se systém nepodaří nasadit včas

Tabulka C.5: Popis rizika R05

Rizikový faktor	Nepřijetí systému zaměstnanci zákazníka
Stav	Potenciální
Pravděpodobnost výskytu	10%
Dopad	Odmítání pracovat se systémem
Mitigace rizika	Školení zaměstnanců
Krizový plán	Dodatečné školení zaměstnanců Vytvoření přívětivějšího uživatelského rozhraní

Tabulka C.6: Popis rizika R06

Rizikový faktor	Problémy při nasazování systému
Stav	Potenciální
Pravděpodobnost výskytu	10%
Dopad	Opoždění finálního nasazení
Mitigace rizika	Zkouška nasazení v rámci testování, správné provedení integračních testů
Krizový plán	Zjištění závad, instalace potřebných aplikací třetích stran

Tabulka C.7: Popis rizika R07

C.4 Matice odpovědností

Celý projekt se nechal rozpadnout na jednotlivé pracovní balíky, ke kterým byli následně přiřazeni jednotliví členové realizačního a implementačního týmu.

Aby byla tabulka jednoduše přístupná na jednom listu, bylo nutné zavést několik zkratek. V hlavičce následující tabulky jsou uvedeni členové týmu, kteří budou projekt realizovat:

- PROM — projektový manažer,
- SWAN — softwarový analytik,
- SWAR — softwarový architekt,
- PROG — programátor,
- TEST — tester.

Dále je potřeba vydefinovat druhy odpovědností, které můžou mít jednotlivý členové týmu k danému pracovnímu balíku:

- S — schvalování,
- P — provádění,
- K — konzultace,
- I — informování o stavu.

Matice odpovědností byla vytvořena podle standardu RACI. Přiřazení úkolů jednotlivým členům týmu je ukázáno v tabulce C.8.

Pracovní balík	PROM	SWAN	SWAR	PROG	TEST
Analýza požadavků	P, S	P			
Wireframy	S	P			
Návrh databáze	I	S	P		
Návrh komunikace server — klient	I	S	P		
Návrh serverové architektury	I	S	P		
Návrh architektury klientské aplikace	I	S	P		
Implementace databáze	I		K, S	P	
Implementace serveru — sběr dat	I		K, S	P	
Implementace serveru — poskytování dat	I		K, S	P	
Uživatelská aplikace	I		K, S	P	
Unit testy	I		S	P	P
Integrační testy	I		S		P
Akceptační testy	I		S		P
Nasazení do produkčního prostředí	I, S	P		P	
Školení uživatelů	I, S	P			
Vyhodnocení projektu	P, S				

Tabulka C.8: Matice odpovědností

C.5 Změnové požadavky

Během projektu je možné vznést změnové požadavky na projekt obecně. Nemusí se tedy jednat jen o změnu systému, ale například i o žádost o posunutí termínů. Všechny tyto požadavky budou muset být ohlášeny manažerovi projektu.

Každý změnový požadavek musí podrobně specifikovat změnu a důvod. Každým změnovým požadavkem se bude projektový manažer a jeho tým zabývat a analyzovat jej.

Pokud bude požadavek zamítnut, projektový manažer odešle zákazníkovi vyrozumění s důvody. V případě přijetí změnového požadavku bude změna zanesena do plánů projektu. Je ovšem potřeba počítat s možností prodloužení projektu nebo zvýšení nákladů. O všech těchto skutečnostech bude zákazník předem informován.

C.6 Komunikace se zákazníkem

Komunikace se zákazníkem bude probíhat prostřednictvím emailu. Zákazník bude v pravidelných čtrnáctidenních intervalech informován o průběhu projektu, především o dodržování časového harmonogramu a stanovených financích. V dokumentu bude také uveden postup prací, zmíněna rizika projektu, jejich pravděpodobnost nastání a změny pravděpodobnosti nastání. Za komunikaci se zákazníkem je zodpovědný manažer projektu.

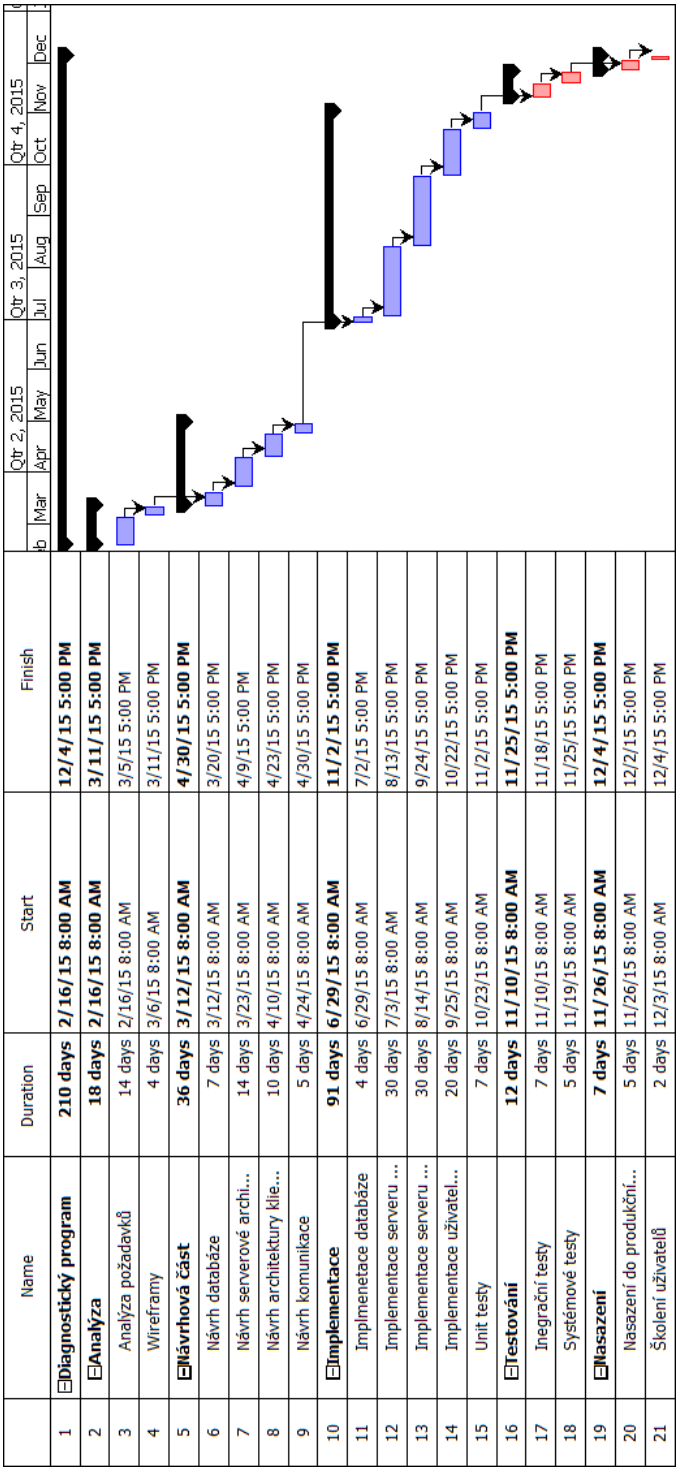
C.7 Harmonogram

Byl stanoven harmonogram prací, který je zobrazen spolu s Ganttovým grafem na obrázku C.4. Ganttův graf přehledně ukazuje návaznosti a délku trvání jednotlivých etap.

Projekt byl odstartován 16.2.2015 a plánované dokončení projektu bylo stanoveno na 4.12.2015.

C.8 Náklady

Na závěr v tabulce C.9 je uveden souhrn jednotlivých fází a jejich odhadovaná cena a celková odhadovaná cena za realizaci projektu.



	Name	Duration	Start	Finish
1	**Diagnostický program**	**210 days**	**2/16/15 8:00 AM**	**12/4/15 5:00 PM**
2	**Analýza**	**18 days**	**2/16/15 8:00 AM**	**3/11/15 5:00 PM**
3	Analýza požadavků	14 days	2/16/15 8:00 AM	3/5/15 5:00 PM
4	Wireframy	4 days	3/6/15 8:00 AM	3/11/15 5:00 PM
5	**Návrhová část**	**36 days**	**3/12/15 8:00 AM**	**4/30/15 5:00 PM**
6	Návrh databáze	7 days	3/12/15 8:00 AM	3/20/15 5:00 PM
7	Návrh serverové archi...	14 days	3/23/15 8:00 AM	4/9/15 5:00 PM
8	Návrh architektury klie...	10 days	4/10/15 8:00 AM	4/23/15 5:00 PM
9	Návrh komunikace	5 days	4/24/15 8:00 AM	4/30/15 5:00 PM
10	**Implementace**	**91 days**	**6/29/15 8:00 AM**	**11/2/15 5:00 PM**
11	Implmenetace databáze	4 days	6/29/15 8:00 AM	7/2/15 5:00 PM
12	Implementace serveru ...	30 days	7/3/15 8:00 AM	8/13/15 5:00 PM
13	Implementace serveru ...	30 days	8/14/15 8:00 AM	9/24/15 5:00 PM
14	Implementace uživatel...	20 days	9/25/15 8:00 AM	10/22/15 5:00 PM
15	Unit testy	7 days	10/23/15 8:00 AM	11/2/15 5:00 PM
16	**Testování**	**12 days**	**11/10/15 8:00 AM**	**11/25/15 5:00 PM**
17	Inegrační testy	7 days	11/10/15 8:00 AM	11/18/15 5:00 PM
18	Systémové testy	5 days	11/19/15 8:00 AM	11/25/15 5:00 PM
19	**Nasazení**	**7 days**	**11/26/15 8:00 AM**	**12/4/15 5:00 PM**
20	Nasazení do produkční...	5 days	11/26/15 8:00 AM	12/2/15 5:00 PM
21	Školení uživatelů	2 days	12/3/15 8:00 AM	12/4/15 5:00 PM

Obrázek C.4: Rorvrh jednotlivých činností a Ganttův graf

Práce	Cena (v Kč)
Analýza	57 600
Návrh systému	115 200
Implementace	291 200
Testování	38 400
Nasazení a školení	22 400
Celková cena	**524 800**

Tabulka C.9: Souhrn odhadované ceny projektu

Příloha D

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
zadani.pdf ..... zadání bakalářské práce
src
    thesis ..... zdrojová forma práce ve formátu LaTeX
logicky_ramec.pdf ..... logický rámec projektu
BP_Hanousek_Tomas_2015.pdf ..... text práce ve formátu PDF
```