Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj softwarového nástroje pro diagnostiku dat

Jan Dort

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Dort

Informatika

Název práce

Vývoj softwarového nástroje pro diagnostiku dat

Název anglicky

Development of software tool for data diagnostics

Cíle práce

Cílem bakalářské práce je vytvořit softwarový nástroj pro zlepšení komfortu při vývoji diagnostických EKG zařízení, nebo diagnostického SW. Výsledný nástroj pomůže vyhodnocovat sady EKG záznamů ve formátu XML, čímž bude možné lépe využít tyto záznamy ve výzkumu a vývoji.

Dílčí cíle bakalářské práce jsou:
- Vytvořit literární rešerši týkající se problematiky vývoje software.
- Analyzovat uživatelské potřeby a vytvořit model softwarového nástroje.
- Vyvinout algoritmus softwarového nástroje pro vyhodnocování EKG záznamů.
- Syntetizovat výsledky práce, formulovat přínosy a závěry práce.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Popisuje teoretické principy EKG signálu, možných sledovaných ukazatelů a metod zpracování signálu s přehledem nejpoužívanějších softwarových nástrojů pro práci s EKG signálem.

Vlastní řešení je realizováno metodami z oblasti algoritmizace, programování a softwarového inženýrství.

Na základě shrnutí teoretických poznatků a výsledků vlastní části práce budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

30-40 stran

Klíčová slova

vývoj software, software, programování, softwarové inženýrství, diagnostika EKG signálu,

---

Doporučené zdroje informací

BAATZ, G. – LANDA, L. *EKG u psa a kočky : technika, vybavení, interpretace*. Praha: Grada, 2006. ISBN 80-247-1394-2.

KHAN, M I G. *EKG a jeho hodnocení.* Praha: Grada, 2005. ISBN 80-247-0910-4.

VRANA, I. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA. *Software engineering.* Praha: Česká zemědělská univerzita, 2013. ISBN 978-80-213-2349-0.

ZVÁROVÁ, J. *Biomedicínská statistika. Díl 1, Základy statistiky pro biomedicínské obory.* Praha: Karolinum, 2011. ISBN 978-80-246-1931-6.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

doc. Ing. Jan Tyrychtr, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 08. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj softwarového nástroje pro diagnostiku dat" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14.3.2023 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu své práce doc. Ing. Janu Tyrychtrovi, Ph.D za vedení a pomoc se zpracováním tématu do závěrečné práce. Dále bych velice rád poděkoval své ženě Šárce za velkou dávku trpělivosti a celému svému okolí za to, že je velice tolerantní.

Vývoj softwarového nástroje pro diagnostiku dat

Abstrakt

Tato bakalářská práce se zabývá studiem problematiky vývoje softwaru z pohledu vývojových metodik a jejich využití, zejména metodám Waterfall a Scrum. Dle zadání je v práci zpracován návrh a implementace jednoduchého nástroje, algoritmu, pro pomoc vývojovému týmu při vyvíjení nových funkcionalit automatické diagnostiky EKG signálu. K návrhu algoritmu jsou využity metody softwarového inženýrství. Algoritmus je zapsán jazykem Python a je navržen pro čtení informací ze souborů v adresáři na dané cestě. Výsledkem je zápis hodnot do souboru typu Microsoft Excel. K tomu je využito znalostí z oblasti programování a informací získaných studiem odborné literatury.

Klíčová slova: vývoj software, software, programování, softwarové inženýrství, diagnostika EKG signálu

Development of a software tool for data diagnostics

Abstract

This bachelor thesis deals with the study of software development from the perspective of development methodologies and their use, especially methods Waterfall and Scrum. According to the assignment, the thesis deals with the design and implementation of a simple tool, an algorithm, to assist the development team in developing new functionalities for automatic ECG signal diagnosis. Software engineering methods are used to design the algorithm. The algorithm is written in Python programming language and is designed to read information from files in a directory on a given path. As a result, the values are written to a Microsoft Excel file. This is done using knowledge of programming and information obtained by studying the literature.

Keywords: software development, software, programming, software engineering, ECG signal diagnostics

Obsah

1 Úvod

Během vývoje medicínského software se vývojové týmy společností setkávají s celou řadou specifických výzev, které jsou způsobeny nutností dbát zajištění vysoké úrovně bezpečnosti a spolehlivosti těchto aplikací. Díky tomu je vývoj medicínského software řazen k těm nákladnějším a časově náročnějším, než běžný software. K hladkému dosažení cílů projektů je nutné dodržovat pravidla v každé fázi životního cyklu takového software. Z pohledu vývojových metodik, které taková pravidla určují, samozřejmě došlo k historickému vývoji a dnes se můžeme setkat s více přístupy k řízení vývoje.

Jelikož již delší časové období sbírám praktické zkušenosti ve firmě zabývající se vývojem a výrobou lékařských přístrojů a softwaru, včetně těch zaměřených na kardiologii, zohlednil jsem to ve výběru tématu pro svou bakalářskou práci. Problematika vývoje softwaru a možnost podílet se na novinkách v oblasti lékařského vybavení mne motivovala se problémem z tohoto odvětví zabývat i v mé bakalářské práci. Z toho důvodu jsem se rozhodl pro volbu tématu s cílem navrhnout a implementovat softwarový nástroj, který by mohl usnadnit práci vývojovému týmu při navrhování nových směrů ve zpracování diagnostických dat. V bakalářské práci se zabývám problematikou vývoje softwaru v oblasti zdravotnictví, konkrétně v oboru kardiologie.

V práci se věnuji teoretické literární rešerši s cílem obsáhnout problematiku vývoje softwaru v prostředí zdravotnictví. Takové prostředí skýtá mnohá úskalí, se kterými je nutno během vývoje počítat, jako je například nutnost klást zvýšené standardy na kvalitu dodávaného řešení. Vzhledem k tomu, že takový software, který je nasazen v nemocnicích, ambulancích a dalších specializovaných pracovištích, je velmi komplexní, tak jeho vývoj vyžaduje také velkou míru robustnosti. K tomu slouží různé vývojové metody a směry. Takové metody pomáhají vývojovým pracovníkům lépe pracovat k naplnění jejich cíle a managementu pomáhají celý vývoj řídit. Zároveň také velkou měrou přispívají k dodržování standardů a norem, čímž je zaručena ona zmíněná kvalita.

Elektrokardiografie, zkráceně EKG, je jedním ze základních metod určování zdravotního stavu člověka. Samotné čtení takového signálu je stále velmi rychle se vyvíjející obor, stejně jako tvorba nových zařízení pro snímání signálu a jeho interpretaci. Mým

parciálním cílem je takovému vývoji pomoci, a proto je celá tato práce věnována nejen čistě metodám a přístupům z oblasti informačních technologií, ale je také zpracována teorie EKG z fyziologického pohledu.

Práce je v praktické části velmi úzce zaměřena na parciální část vývoje automatizovaného vyhodnocení EKG signálu pomocí softwaru s uživatelským rozhraním na základě algoritmického zpracování těchto dat. Samotná práce řeší právě jeden takový podpůrný algoritmus, který vývojovému týmu pomáhá v další práci při vývoji nových algoritmů pro strojové vyhodnocování EKG signálu. Mým cílem specifikovaným v zadání je věnovat se návrhu takového algoritmu spolu s dodáním kódu, který specifikovanou činnost uživateli, vývojáři, bude vykonávat.

V průběhu mé bakalářské práce se zaměřuji na vývoj samotného softwarového nástroje pro zlepšení komfortu při vývoji diagnostiky dat v medicínském oboru kardiologie. Hlavním cílem práce je pomoci k vývoji softwaru, který bude schopen lépe interpretovat signál EKG a tím pomoci lékařům v diagnostice zdravotního stavu pacientů.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je vytvořit softwarový nástroj pro zlepšení komfortu při vývoji diagnostických EKG zařízení, nebo diagnostického softwaru. Výsledný nástroj pomůže vyhodnocovat sady EKG záznamů ve formátu XML, čímž bude možné lépe využít tyto záznamy ve výzkumu a vývoji.

Dílčí cíle bakalářské práce jsou:

- Vytvořit literární rešerši týkající se problematiky vývoje software.
- Analyzovat uživatelské potřeby a vytvořit model softwarového nástroje.
- Navrhnout algoritmus softwarového nástroje pro vyhodnocování EKG záznamů.
- Syntetizovat výsledky práce, formulovat přínosy a závěry práce.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů s vytvořením literární rešerše. Literární rešerše popisuje metodiky řízení vývoje softwaru a přístupy k vývoji softwaru například i z pohledu vhodných programovacích jazyků pro práci s daty.

Vlastní řešení je realizováno metodami z oblasti algoritmizace, programování a softwarového inženýrství. Konkrétně se jedná o použití algoritmizace pro využití vytvořených funkcí pro práci s více soubory v daném adresáři, kdy se jedná o využití jednoduchého jednovláknového přístupu zpracování.

Ke zpracování návrhu řešení je využito metodik softwarového inženýrství ze znalostí nabitých při studiu, včetně diagramů datového toku. Diagram datového toku DFD je nástroj, který se používá k popisu pohybu dat v informačním systému. Pomocí symbolů a šipek ukazuje procesy, databáze a interakce s externími entitami. Skládá se ze 4 hlavních symbolů:

- proces – je znázorněn kroužkem,
- tok dat – je znázorněn šipkou,
- data store – je znázorněn obdélníkem,
- terminátor – je znázorněn dvojicí rovnoběžných čar.

Tento diagram poskytuje přehled o tom, jak data v systému putují a jak jsou zpracovávána. V podstatě se jedná o grafické znázornění toku dat v informačním systému. DFD nepopisuje informace o řízení a časování systému.

V implementační fázi praktické části práce je využito znalostí programování, konkrétně programovacího jazyku Python. Python je použit nejen pro samotné zapsání kódu algoritmu, ale také k jeho otestování.

Na základě shrnutí teoretických poznatků a výsledků vlastní části práce jsou formulovány závěry bakalářské práce nejen formou závěru, ale také diskuze nad dosaženými poznatky.

3 Teoretická východiska

V teoretické části práce je věnován prostor pohledu na vývoj software ze strany metodik řízení procesu vývoje, kdy se zaměřuje na krátký popis lineárních a agilních metodik, s hlavní částí věnovanou specializaci tématu této práce, tedy vývoj softwaru určeného pro skupinu uživatelů z oblasti zdravotnictví, respektive kardiologie. V této části práce je dále nastíněna problematika vývoje software z pohledu bezpečnosti pro koncového uživatele, tedy pacienty, kdy je zmíněno několik zásadních regulací, které musí výsledný produkt splňovat. V závěru teoretické části práce nechybí popis a základní vysvětlení EKG a popis samotné EKG signálu neboli EKG křivky.

3.1 Obecně o metodách řízení vývoje softwaru

Podle historického vývoje v oblasti problematiky vývoje softwaru se do dnešní doby vyspecifikovaly dva stěžejní pohledy, jakým způsobem přistupovat k vývoji. Metody řízení vývoje softwaru se dle většiny dostupných zdrojů základně dělí na tradiční metodiky a agilní metodiky.

Tradiční metodiky jsou zatíženy pevným a neměnným plánem vývoje, kdy se jednotlivé fáze projektu řadí za sebe do předem určeného pořadí a vykonávají se jedna za druhou. Metodiky tohoto typu počítají s tím, že požadavky jsou pevně stanoveny a po celou dobu vývoje jsou neměnné. (Myslín, 2016, str. 23-24)

Agilní metodiky se vyznačují iterativním a inkrementálním přístupem k vývoji software. Požadavky na produkt jsou pravidelně aktualizovány během procesu vývoje po konzultacích se zákazníkem. Tyto metodiky umožňují lepší a rychlejší reakci na změny v požadavcích a umožňují užší a kvalitnější spolupráci mezi dodavatelem a zákazníkem. (Myslín, 2016, str. 29-30)

Z pohledu vývoje softwaru se přistupuje k výběru vhodné metodiky vývoje softwaru podle náročnosti řešené problematiky a předpokládané velikosti projektu, kdy pro každou

situaci může být vhodné použití jiné metodiky. Obecně se však přihlíží ke komplexitě řešeného problému. (Myslín, 2016, str. 24-26)

V následujících odstavcích budou představeny oba přístupy. Vzhledem k tomu, že jako nejčastější bývají v literatuře zmiňovány konkrétní implementace dvou hlavních směrů, tedy Waterfall a Agilní metodiky Scrum, bude věnován prostor těmto praktickým použitím zástupců obou hlavních směrů. Obecně se dá tvrdit, že Waterfall se hodí pro menší projekty lineárního charakteru, kdežto Agile se používá pro komplexní, rozsáhlé projekty skrz spektrum vědních oborů a disciplín. (Myslín, 2016, str. 23, 30)

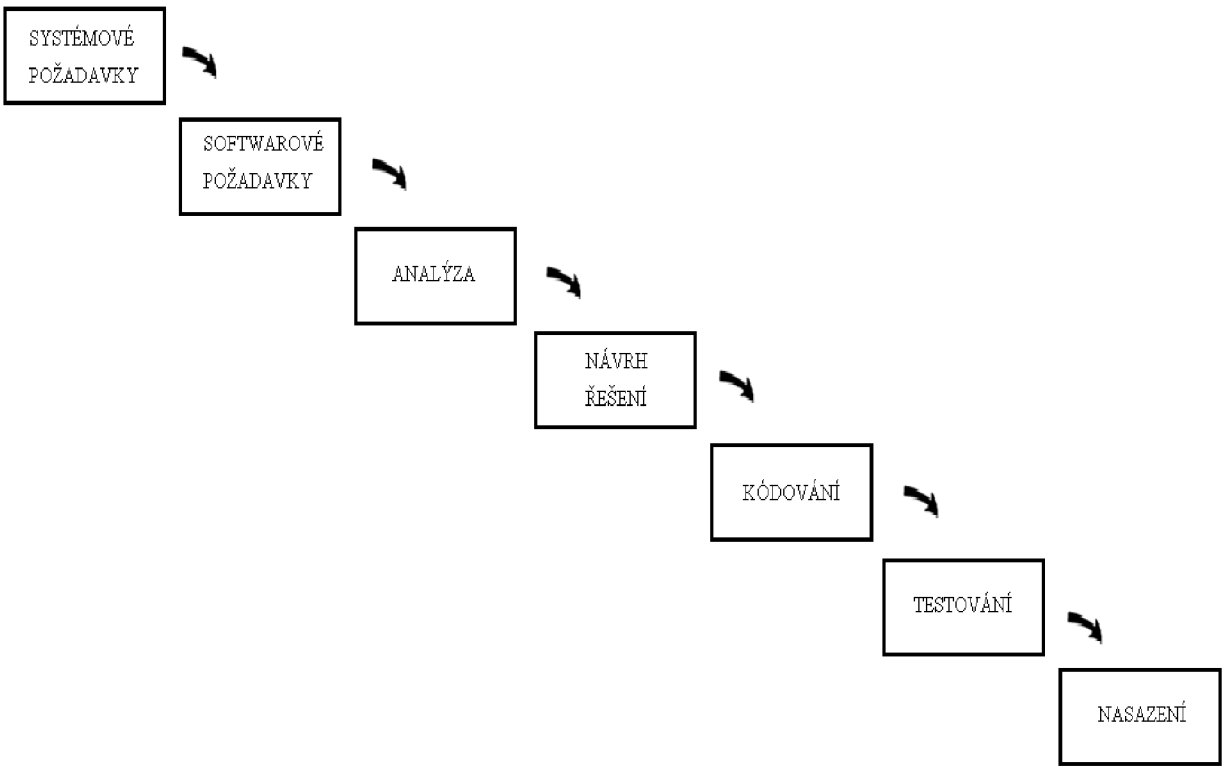
3.2 Tradiční metodiky vývoje software

Jednou z prvních metodik vývoje softwaru, nikoliv v dnešním pojetí kompletního managementu projektů, ale hlavně se zaměřující na proces vývoje, kdy hlavním úkolem bylo sledování jasně definovaných částí vývoje, je "Životní cyklus software"(z anglického "software development life cycle"). Každá část vývoje zahrnutá v celkovém navržení, tedy od počátečního impulsu, nebo požadavku, či potřeby, až po samotné dodání produktu, je v tomto pojetí přesně vymezena. Tato metodika je Elliotem vnímána jako jedna z prvních metodik vývoje, tedy jedná se o první z tradičních popsaných metodik vývoje softwaru. Tento pojem, "Životní cyklus software", se v přeneseném a zobecněné smyslu používá pro popsání vývojových metodik napříč jejich spektrem v popisování přístupu k vývoji softwaru, od počátečního vytyčení potřeb klienta, až po dodání a údržbu řešení dodavatelem. (Elliot, 2004 – str. 87)

Tradiční (lineární) metodiky vývoje softwaru se vyznačují právě snahou o obsáhnutí požadavků na celý produkt již od počátku v jednom velkém souhrnném plánování, a hlavně čistě sekvenčním přístupem. Velký důraz je kladen na dokumentaci. Nejznámějším zástupcem tradičních metodik je metodika Waterfall, která bude v následujících odstavcích představena. (Myslín, 2016, str. 23,25)

3.2.1 **Tradiční metodiky - Waterfall**

Metodika Waterfall byla původně popsána v textu "*Managing the development of large software systems*", autora Winstona W. Royce publikovaného v roce 1970, nicméně metodika v textu nebyla takto pojmenována. Autor zde v první fázi staví do porovnání základní fungování vývoje softwaru s dvěmi fázemi: analýza a kódování spolu s návrhem, či popsáním, modelu o sedmi fázích. Přídávají se systémové požadavky, softwarové požadavky, program design, testing a operations (provoz/údržba). Tyto požadavky společně plní spolu s předchozími fázemi Analýza a Kódování sekvenci 7 kroků, kdy ovšem autor zmiňuje rizika této metodiky z hlediska možnosti dodání konečného produktu a obecně možnosti selhání během procesu. (Royce, 1970, článek)



Obrázek 1 "Waterfall" – Vlastní tvorba podle Royce (1970)

Plusy	Mínusy
Jednoduchost.	Velký časový rozptyl mezi zadáním a hotovým produktem.
V každém okamžiku přesně víme, v jaké fázi se projekt nachází.	Pozdní testování a nesnadné odhalování chyb.
Rozumné možnosti plánování.	Minimální zapojení zákazníka do procesu.
Jednoznačné zadání s minimem změn.	Velká linearita.

Tabulka 1 Klady a zápory metody Waterfall – Vlastní tvorba podle Myslína, 2016, str. 26

3.2.2 Fáze modelu Waterfall podle Royce

1. Systémové požadavky,
2. softwarové požadavky,
3. analýza,
4. návrh řešení,
5. kódování,
6. testování,
7. nasazení.

Model Waterfall prošel v průběhu jeho vývoje proměnami a v různých implementacích má různý počet fází. V různých zdrojích se uvádí nejméně 4 fáze, a nejvíce 10 fází. Setkáváme se i s pojmenováním „hybridní“ model, který buďto vykazuje známky různých kombinací modelu Waterfall, nebo i kombinaci s prvky z Agilních metodik. (Myslín, 2016, str. 23-30)

3.2.2.1 Požadavky a jejich analýza

Analýza je součástí prvních fází vývojové metodiky Waterfall. Obecně jde v této fázi o zpracování požadavků zákazníka a vydefinování softwarových požadavků. Hlavním cílem těchto prvních fází ve vývojové metodice Watefall je vytvoření souhrnného dokumentu, který bude sloužit jako osnova po celou dobu vývoje produktu. Tento dokument se nazývá SRS (Software Requirements Specification) a sdružuje detailní specifikace požadavků, které byly definovány v souladu se zákazníkem a které poté software má splňovat. (Pressman, 2010, str. 39-41, 123)

Analýza jako proces v rané fázi vývoje software začíná tím, že se setkává vývojový tým a zákazník. Spolu diskutují zákazníkovy požadavky a očekávání. Dalším krokem je vytvoření seznamu všech očekávaných funkcionalit, které software na konci vývoje musí obsahovat a soupis veškerých známých omezení, které mohou při tvorbě produktu nastat. Ty musí následně být brány v úvahu pro tvorbu samotného dokumentu SRS, případně projektového plánu. (Pressman, 2010, str. 39-41, 123)

Dalším krokem je vytvoření dokumentu SRS, který obsahuje všechny předem vydefinované požadavky na vyvíjený software. SRS jako celkově pojatý dokument, obsahující požadavky a potřeby zákazníka na software, se zahrnutými možnými požadavky ze strany regulací a norem, je spolu se zázazníkem schválen ještě před samotným vývojem. Po dokončení prvních fází metodiky Waterfall, týkající se analýzy, přichází na řadu další fáze projektu, a to je návrh řešení jakým způsobem bude produkt implementován. (Pressman, 2010, str. 39-41, 123)

Obecně lze shrnout fázi analýzy jako klíčovou fázi projektu, během které jsou hromaděny veškeré požadavky na software a během které vstupuje do projektu zákazník velkou měrou. Fáze analýzy udává celkový směr projektu a je tedy klíčová pro úspěch celého projektu. (Pressman, 2010, str. 60-62)

3.2.2.2 **Návrh řešení**

Další fází vývoje softwaru podle metody Waterfall je fáze návrhu řešení. V této fázi se tvoří detailní návrh jednotlivých komponent navrhovaného produktu za účelem sestavení takto jednotlivě detailně navržených částí do celkové koncepce. Taková koncepce se nazývá architekturou a označuje, jakými vztahy jsou spolu jednotlivé komponenty provázány. (Sommerville, 2011, str. 179,180)

Navrhování detailů jednotlivých funkčních prvků systému se skládá z pravidla z návrhu používaných algoritmů a struktury dat. Do takové kategorie návrhu jednotlivých celků se řadí i návrh uživatelského prostředí konečného produktu. Takto navržené prvky se nazývají moduly, a právě takto vydefinované moduly se poté spojují do dalších funkčních celků v rámci návrhu architektury. (Sommerville, 2011, str. 180-182)

Architektonický návrh popisuje celkový vztah komponent systému a jejich vazby. Nedílnou součástí je také návrh vztahu systému směrem na externí systémy nebo služby. Návrhem architektury je také určení rozhraní, jakým způsobem spolu komponenty komunikují. (Sommerville, 2011, str. 180-182)

Výsledkem této návrhové fáze jsou dokumenty Design Specification a Test Specification. Design Specification je souborem popisu jak jednotlivých prvků systému, tak právě i vazeb mezi nimi, tedy architektury systému. Test Specification popisue testovací procedury pro zaručení správnosti fungování celého systému, ale i jednotlivých komponent. Testovací procedury by měly vycházet právě z Design Specification. (Pressman, 2010, str. 293, 449)

3.2.2.3 **Implementace a testování**

Během fáze implementace se provádí samotné programování softwaru či produktu. V této fázi jsou naplno zužitkovány poznatky a návrhy z předchozích fází metodiky Waterfall. Během fáze implementace dochází k přeměně návrhu na skutečně existující produkt za pomoci projektového plánu. Jednotlivé části, nebo moduly, jsou implementovány zvlášť a poté jsou spojovány do navržené architektury. (Sommerville, 2011, str. 29-32)

Samotná fáze implementace obsahuje několik částí, kdy první je část programovací, během které vývojáři pomocí technik programování vytvoří funkční kód dané části vyvíjeného produktu. Dále dochází k programovému testování funkčních celků, neboli Unit testingu. Jedná se o naprogramovaný kód, který po spuštění testuje dané metody jednotlivých modulů. Během integračního testování dochází k ověření funkčnosti komunikace mezi jednotlivými komponentami podle architektonického návrhu. Již v této fázi může docházet k uživatelskému testování, za předpokladu, že je již propojen dostatek komponent s uživatelským rozhraním. Výstupem testování by měl být protokol o testování, který dokazuje průběh testovacích procedur. (Sommerville, 2011, str. 30-40)

Během testování samozřejmě může dojít k objevení chyb v naprogramovaném modulu. V tuto chvíli vývojový tým spolupracuje na odstranění těchto nedostatků. Konečnou fází implementace části vyvíjeného produktu je dokumentace takové části. Mělo by se jednat

o popsání samotného kódu, dále dokumentace kódu a uživatelská dokumentace v podobě uživatelského manuálu. Taková dokumentace je velice důležitá pro týmovou spolupráci z důvodu snadnějšího sdílení výstupů a zároveň z důvodu udržitelnosti takového kódu. (Sommerville, 2011, str. 30-40)

3.2.2.4 **Nasazení a údržba**

Poslední fází vývoje software podle metodiky Waterfall je udržba a nasazení softwaru do produkčního prostředí u zákazníka. Do této fáze lze postoupit až v momentě, kdy je produkt dokončen nejen po stránce programování, ale také musí být řádně otestován a zdokumentován. Takový proces musí projít fází dokončení a schválení. V této fázi dochází k nasazení software do produkčního prostředí a sleduje se jeho provoz tak, aby byl zajištěn bezproblémový chod. (Sommerville, 2011, str. 31-33)

I fáze nasazení je rozdělena do několika částí. Je nutné naplánovat přesný postup, jakým bude produkt u zákazníka nasazen, včetně lidí, kteří budou za takové nasazení zodpovědní. Takové kroky jsou velice důležité pro hladký proces nasazení a spokojenost zákazníka, který očekává dodání stabilní aplikace. (Sommerville, 2011, str. 31-33)

Během nasazování dochází ke kooperaci mezi vývojovým týmem a IT týmem dané instituce nebo zákazníka. Dochází k přípravě prostředí a následné instalaci produktu. Po nasazení dochází k překlenovací době, kdy je nasazený produkt sledován a je vyhodnocováno, zda jeho provoz odpovídá očekávání. V případě, že nastanou chyby, tak je nutné je řešit v co nejkratším čase ze strany vývojového týmu. Velice důležitou věcí při nasazování je nutnost zálohování práce. Dojde tím k minimalizaci následků v případě, že nasazovaný produkt není provozován na virtuálním prostředí a sdílí zdroje s dalšími softwary. (Sommerville, 2011, str. 31-33)

3.3 Agilní metodiky vývoje software

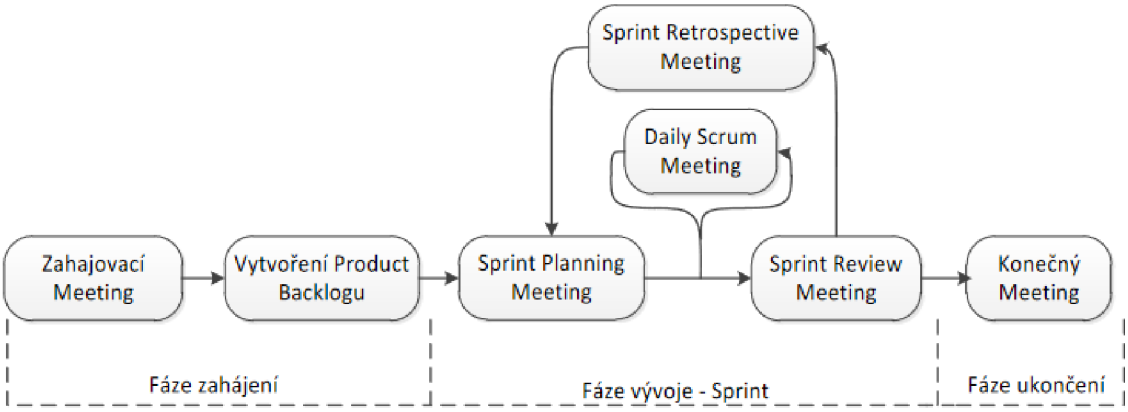
Inkrementální metodiky řízení vývoje softwaru, oproti např. metodice Waterfall, umožňují pružnější reagování na zákaznické požadavky v různých hladinách náročnosti změny požadavků. Tyto změny lze hodnotit jak z hlediska drobných úprav v návrhu dílčích

funkcí software, tak do míry úplné změny požadavků zákazníka. Na tyto případy by správně nastavená agilní metodika řízení vývoje měla být schopna reagovat pružně. (Myslín, 2016, str. 26, 29)

Agilní metodiky vývoje softwaru jsou řízeny filozofií přístupu k vývoji softwaru, která vznikla na přelomu milénia a byla popsána v textu "Manifest agilního vývoje software" skupinou autorů v roce 2001. Hlavní směr metodiky byl popsán následovně: „*Jednotlivci a interakce před procesy a nástroji; Fungující software před vyčerpávající dokumentací; Spolupráce se zákazníkem před vyjednáváním o smlouvě; Reagování na změny před dodržováním plánu.*" Text dále zmiňuje "*dvanáct principů agilního vývoje software*". (Agile Manifesto, 2001)

3.3.1 Agilní metodiky - Scrum

Metoda Scrum řeší přístup k vývoji softwaru krátkými periodickými inkrementy, kdy by se mělo jednat o periody charakteru maximálně měsíce, nejlépe dvou týdnů. Taková perioda se nazývá sprint. V metodě Scrum se rozlišují 4 role. Jsou jimi Product Owner, Scrum Master, člen týmu (vývojový tým) a zákazník. Úkolem Product Ownera je řízení priorit týmu takovým způsobem, aby bylo dosaženo co největšího zisku z investic do týmu, za předpokladu správného vyhodnocení zákaznických požadavků a jejich profitability. Scrum Master tým podporuje k co největší efektivitě a pomáhá řídit schůzky. Členové týmu rozhodují o tom, jakým způsobem a kdo bude na předem připravených úkolech pracovat. Úkoly jsou připravovány do zásobníku tzv. „Product Backlog", který obsahuje vydefinované úkoly, kdy nutností obsahu je nejen zadání úkolu, ale také popsání stavu, ve kterém lze prohlásit úkol za hotový. Metodika Scrum také řeší definici, kdy lze úkol prohlásit za splněný. Podle toho, kdo je zákazníkem daného úkolu, se také mění právě definice splnění úkolu. Pomocí nástrojů jako je Retrospektiva a Denní Scrum je možné řešit nálady v týmu, případně komunikovat nedostatky navržených či implementovaných řešení. (Sims, Johnson, 2012, str. 3-6)



Obrázek 2 Scrum - HALAMA, Petr - Agilní metody v projektovém řízení

3.3.1.1 **Role ve Scrumu**

Josef Myslín ve své knize rozděluje role do dvou základních skupin, a to sice „Pigs and Chicks“. Toto pojmenování vysvětluje na příběhu otevření restaurace, kdy jedna strana, Pigs, nesouhlasí s pojmenováním a konceptem, jelikož by musela obětovat více, než druhá strana, tedy Chicks, která má dle příběhu přispět menší měrou. Na tomto příběhu je vysvětlena přímá a nepřímá účast na projektu, kdy obecně se dá hovořit o tom, že více zúčastněná strana má větší podíl na práci, nicméně bez té druhé se neobejde. Takovými přímými účastníky jsou členové Scrum týmu, Scrum Master a Product Owner, kdežto nepřímým účastníkem je zákazník, nebo například poradce. (Myslín, 2016, str. 55-56)

Product Owner je zodpovědný za produkt jako takový a je v procesu vývoje do jisté míry zástupcem zákazníka. Za jistých podmínek bývá také určitým prostředníkem mezi zákazníkem a zbytkem týmu. Product Owner přichází s vizí projektu a jeho cílem by mělo být dodržení takové vize, za předpokladu, že je stále v souladu s přáním zákazníka. Definuje také priority pro vývojovou část Scrum týmu a podle toho také řídí uspořádání backlogu. (Myslín, 2016, str. 61-63)

Scrum Master je manažerem Scrum procesu. Na rozdíl od Product Ownera, který určuje priority a „tvrdě“ může zasahovat do průběhu vykonávané práce, tak rolí Scrum Mastera je podporovat a motivovat tým k lepší výkonosti pro splnění Product Ownerem nastavených priorit. Scrum Master není manažerem ostatních v pravém slova smyslu, není

jejich nadřízeným. Pro naplnění své role využívá Scrum Master metod a nástrojů nejen agilních metodik, jako je například Daily Scrum, Retrospektiva, ale také například Facilitace pro vedení schůzek. (Myslín, 2016, str. 73-75)

Vývojář – člen Scrum týmu - je role, která dodává konečnou implementaci produktu. Ve Scrumu má vývojový pracovník velký vliv na průběh implementace v mnoha ohledech. Nejen, že buď přímo vybírá, nebo se účastní výběru použitých technologií, ale také má možnost pracovat na úkolech podle svého vlastního výběru. Z toho tedy plyne, že v definici Scrumu není práce předem přidělována, ale tým a každý jeho člen se sám rozhodne, na jakém úkolu bude pracovat. (Myslín, 2016, str. 76-78)

3.3.1.2 **Měření výkonnosti**

Ve Scrumu se můžeme setkat s více možnostmi, jakým způsobem měřit výkonost týmu, nebo jednotlivců v daném týmu. Účelem takového měření by mělo být získání dat pro vyhodnocení a následné zhodnocení, případně vyvození opatření.

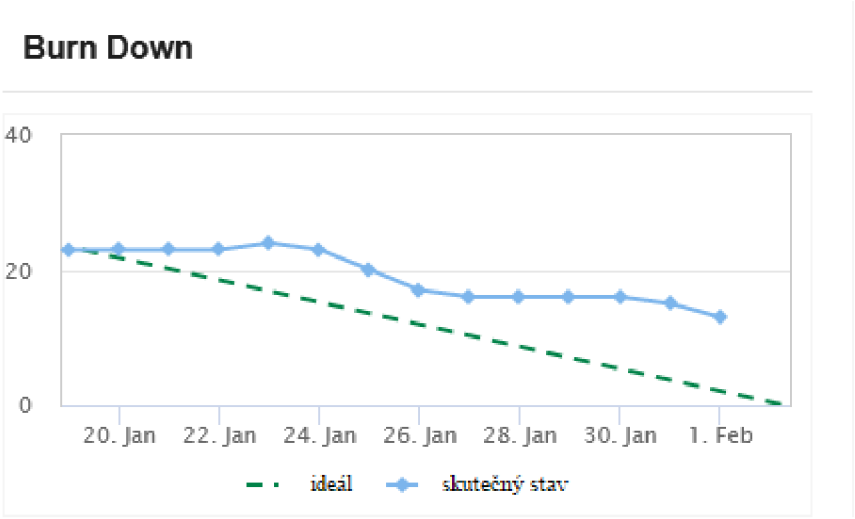
Jednou z těchto metod je Definiton of Done, tedy definice dokončení. Využívá se hlavně u jednotlivých dílčích kroků ve vývoji, jednotlivých inkrementů. Slouží hlavně pro případy zdánlivě nekončících úkolů, kdy touto definicí je poté možné úkol dokončit v rámci daného času na jeden Sprint. (Šochová, Kunce, 2019, str. 84)

Sprint goal, to je pojmenování cíle, který by měl být splnitelný a měl by být splněn v rámci jednoho Sprintu. Měla by tím být označena jedna část, která by měla být jasně definována a mělo by být možné ji demonstrovat na Sprint Review. (Šochová, Kunce, 2019, str. 61)

Scrum zavádí relativní měrné jednotky, které jsou určeny a definovány pro každý daný vývojový tým a nejsou mezi týmy přenosné. Tyto mohou být použity pro ohodnocování práce, ale neměly by být spojovány s časem, nýbrž odhadem náročnosti úkolu. Nejčastěji se setkáváme s pojmem Story Point, který hodnotí náročnost implementace jedné User story. Z definice a pojetí Scrumu plyne, že takto hodnotit by se měly jen a právě User Story. Hodnocení samotných úkolů, na které se User Story „rozpadá“, již nedává smysl a může být

pro tým zdlouhavé. V praxi se však s takovým hodnocením úkolů setkat lze. (Šochová, Kunce, 2019, str. 93)

Burndown diagram je jednoduchý graf, který znázorňuje vývoj náplně práce ve Sprintu pro daný tým. V průběhu času plynutí Sprintu se může zobrazovat libovolná veličina. Může jít například o Story Pointy nebo počet úkolů ve Sprintu. V ideálním případě má graf na začátku Sprintu určitou hodnotu, a s plynutím času se míra práce vyjádřená na grafu zmenšuje. Nemělo by docházet k jeho nárůstu uprostřed Sprintu. (Myslín, 2016, str. 121-123)



Obrázek 3 Ukázka burndown diagramu. Vlastní tvorba.

Popis obrázku č. 3. Na obrázku je vidět burndown diagram, který sleduje ideální postup týmu ve sprintu a skutečný postup týmu ve sprintu. Na ose X je vidět časová osa a na ose Y je zanesen počet úkolů.

3.4 Brainstorming a Facilitace

Brainstorming je metoda řízení schůzek za účelem tvorby nápadů, která se používá v mnohých oblastech lidského snažení, nejenom v IT vývoji. Jinak tomu není ani v případě tvorby nových softwarových produktů. Metoda brainstormingu je velmi účinným prostředkem pro získání nápadů ohledně daného problému ve velmi širokém spektru

přístupů. Primárně důležitou vlastností brainstormingu je podpora kreativity a také to, že podporuje spolupráci v týmu. (Kaner a kolektiv, 2007, str. 117-120)

Během brainstormingu nedochází k facilitaci nebo jinému moderování, a je nechán prostor pro volný běh myšlenkových pochodů všech zúčastněných členů. Nicméně samotné myšlenky jsou nezávislou osobou zaznamenávány pro další zpracování. (Kaner a kolektiv, 2007, str. 117-120)

Pro správné fungování brainstormingu je nutné udržovat pozitivní a nerušenou atmosféru. To je velice důležitým faktorem pro uvolnění případných sociálních blokací u všech účastníků takového setkání, a jen tak je možné získat celou škálu názorů. Je také nutné myslet při sestavování pozvaných členů na to, že je v tomto případě velmi žádané mít zastoupené různé názorové skupiny. A to nejen z pohledu zkušeností, ale také znalostí a dalších možných aspektů. Takovým způsobem mohou vzniknout velice originální nápady, které by jinak zůstaly ležet v myslích účastníků nevyřčeny. (Kaner a kolektiv, 2007, str. 117-120)

Facilitace je metoda vedení schůzek, kdy hlavní rolí v této metodě je role facilitátora. Facilitátor má za úkol pomoci skupině lidí dosáhnout určitého společného cíle s motivací vyřešit řešený problém. Toho se snaží dosáhnout moderací rozhovoru, dává prostor pro vyjádření názoru rovnoměrně a usnadňuje účastníkům interakci. Role facilitátora je velmi náročná na schopnosti, jako je řešení konfliktů, aktivní poslech a další metody sociální psychologie. Facilitace je hojně využívána v oblastech nejen meetingů, ale také různých workshopech, na školeních a dalších. (Kaner a kolektiv, 2007, str. 31-37)

Aby byl proces facilitace a facilitátor samotný úspěšný, tak je především nutné dbát na otevřenost komunikace a dodržovat nastavená pravidla. Především dodržet to, aby všichni účastníci měli možnost vyjádřit svůj názor, a měli možnost cítit se respektovaní.
(Kaner a kolektiv, 2007, str. 31-37)

3.5 Agilní metodika ve vývoji lékařského software

Jak již bylo v práci uvedeno, tak agilní metodiky vývoje softwaru přinášejí hlavně bližší kontakt mezi zákazníkem a vývojovým týmem nejen v podobě dodávání částí vyvíjeného softwaru v krátkých časových intervalech. V této kapitole bude nastíněn střet tohoto teoretického přístupu agilní metodiky vývoje software, respektive jejího praktického použití Scrumu, s praxí vývoje medicínského software, případně přístrojů. Vývoj medicínského softwaru a přístrojů je regulován celou řadou nařízení a vývojový tým, respektive jeho vedení, musí s těmito regulacemi počítat nejen v samotném vývoji, ale také při výběru metodiky vývoje software.

Vzhledem k tomu, že ve zdravotnictví sebemenší chyba při diagnostice nebo následné léčbě může vést k nevratným ztrátám na životech či trvalému poškození pacientů, je zde kladen velký důraz na přesnost produktů využívaných k diagnostice a léčbě pacientů. To také včetně softwaru používaného v lékařských zařízeních, a to nejen ze strany lékařského personálu, ale také ze strany managementu lékařského zařízení. (Kolektiv autorů – 2012, s. 13-15)

Výzvou při vývoji kritického informačního systému, jakým medicínský software bezesporu je, je nutnost zajištění jeho spolehlivosti a dostupnosti 24 hodin denně, 7 dní v týdnu. Software určený do nemocnic musí být schopný pracovat bezchybně v každé situaci, za náročných podmínek. Z těchto důvodů se vývoj medicínského softwaru prodražuje, jelikož je nutno zajistit důkladné a pečlivé testování ve všech oblastech použití softwaru a jeho funkčností. Je také nutno dbát na integrační testy s ostatními systémy, které jsou na daném pracovišti nasazeny. (Materová, Vrublová – 2013, Str. 10-13)

3.6 Kyberbezpečnost informačních systémů

Informační software se stal nedílnou součástí mnoha organizací, od zdravotnictví po finance, a mnoho dalších odvětví. Takový software pomáhá podnikům zpracovávat data a analyzovat je pro další použití. Na základě analýzy dat z informačního systému například management činí rozhodnutí. Proto je nezbytné zajistit, aby byl informační software bezpečný a chráněný proti hrozbám, které mohou ohrozit hladký chod systému, nebo ohrozit správnost zpracovávaných dat. (Pressman, 2010, str. 408,409)

Jedním z klíčových prvků, jakým lze zabezpečení informačního softwaru dosáhnout, je použití kontroly přístupu. Jedná se o bezpečnostní prvek, který řídí, kdo může do systému přistupovat, a co může po přihlášení do systému dělat. Patří sem kontrolní mechanismy ověřování přístupu a oprávnění, jako je například uživatelské jméno a heslo nebo pokročilejší technologie, jako RFID a další. Tyto metody kontroly přístupu pomáhají zabránit neoprávněnému používání systému a chrání například před únikem dat, nebo dalším ohrožením klientů organizace.

Další, dnes již velmi rozšířenou formou ochrany informačního softwaru, je šifrování dat. Šifrování je proces přeměny dat do lidem ani strojům nečitelného formátu, který lze dešifrovat, a tedy přečíst, pouze pomocí dešifrovacího klíče. Existuje celá řada šifrovacích mechanismů, kdy se jedná o velmi účinný způsob ochrany dat před neoprávněným přístupem a využitím. Šifrování tedy hlavně zajišťuje to, že i když jsou data zachycena v přenosu, je těžké, nebo až nemožné, data bez šifrovacího klíče dešifrovat a číst informace.
(Pressman, 2010, str. 549, 550, 819)

Mimo již zmíněné metody řízení přístupu a šifrování existuje celá řada dalších bezpečnostních opatření, která lze použít k ochraně informačního softwaru a v něm uložených dat. Mezi ně patří zejména pravidelné bezpečnostní audity, které řeší odhalování možných bezpečnostních rizik. Velmi důležitou roli hraje také bezpečnostní školení zaměstnanců. Tímto způsobem lze předejít mnoha problémům z hlediska možných bezpečnostních hrozeb. Zaměstnanec, který nemá alespoň základní přehled jakým způsobem chránit například data, se kterými pracuje, může způsobit společnosti velký problém na mnoha úrovních. (Pressman, 2010, str. 398-402)

Vzhledem k tomu, že není vždy možné ošetřit všechny případy, které mohou nastat, a stále tak existuje možnost úspěšného kybernetického útoku, nebo „obyčejné" havárie, tak je nutné dbát na záložní plán při takové situaci. Pro takové případy slouží plány pro obnovu systému. Čím komplexnější systém, tím by měl být také zpracován a udržován plán obnovy takového systému. Nedílnou součástí takových opatření je zálohování dat systému. Plán obnovy by měl být také průběžně testován. Obecným tvrzením souhrnu takových opatření

by mohlo být, že dodržováním zásad a zpracovaných postupů lze předejít velkým škodám na majetku a důvěryhodnosti organizací. (Pressman, 2010, str. 470-471)

Zde je uvedeno několik příkladů demonstrujících implementaci bezpečnostních opatření v informačním systému:

- Logování uživatelských aktivit.
- Logování aktivit systému.
- Automatické odhlašování uživatele po určitém časovém intervalu.
- Řízení přístupu do částí systému daným skupinám uživatelů dle jejich zařazení.
- Demo režim systému (např. pro školení personálu).
- Jasná identifikace uživatele v systému.
- Ochrana dat pomocí šifrování.
- Zálohování systémových dat.
- Šifrování dat.
- Antivirový systém.

(Tanenbaum a Bos, 2015, str. 593-595, 704-705)

3.7 Programovací jazyky pro datovou vědu

Existuje několik programovacích jazyků a přístupů pro práci s daty v oboru datové vědy, jako je například Python, R, Java, Julia nebo také databázový jazyk SQL. Nicméně jazyk Python se umisťuje na prvních příčkách v průzkumech oblíbenosti již několik let. Python se stal oblíbenou volbou vývojářů pro práci s daty díky své jednoduchosti, přizpůsobivosti a dostupnosti knihoven třetích stran. Python se stal obecným standardem pro strojové učení a datovou vědu. Oproti konkurenčním jazykům v těchto oblastech, jako jsou R a Julia, má jazyk Python řadu výhod, jako je stručnost zápisu kódu a s tím spojená rychlost vývoje. Python je skvělým jazykem pro začátečníky, díky své jednoduché syntaxi a snadnému použití. Podle průzkumu Kaggle je Python nejpoužívanějším jazykem pro datovou vědu. Více než 80 % respondentů uvedlo, že Python používá právě pro práci s daty a datovou vědu. (Kaggle, 2022)

Další nespornou výhodou jazyka Python je široké spektrum možností použití. Kromě strojového učení a vědeckých výpočtů, je také možné použít ho během vývoje webových stránek. Datoví vědci potřebují pracovat s různými druhy a formáty dat, a proto je pro ně

Python vhodnou volbou. To, že je Python velice oblíbený, dokazují také data portálu Stackoverflow, která ukazují množství hledání řešení problémů v různých programovacích jazycích. (YEPIS, 2023)

Dalším plusem jazyka Python, díky čemuž se těší takové popularitě mezi vývojáři, je dostupnost knihoven a frameworků třetích stran. Vývojáři jazyka Python vytvořili řadu knihoven a frameworků nejen pro práci s daty. Nejznámější knihovny pro datovou vědu jsou dle zdrojů NumPy, Pandas a Matplotlib. Díky těmto knihovnám je práce s daty v jazyce Python jednoduchá. Tyto knihovny poskytují nástroje pro manipulaci s daty, jejich analýzu a vizualizaci**.** (Python dokumentace)

Přestože existují i jiné jazyky, které jsou v této oblasti vývoje oblíbené, tak je Python díky své popularitě a komunitě vývojářů dále rozšiřován a drží se vysoko v průzkumech oblíbenosti.

3.8 Popis EKG signálu

Elektrokardiogram neboli EKG, patří mezi základní diagnostické metody v oblasti nejen kardiologických vyšetření, ale také se s ním lze setkat během preventivní prohlídky u praktického lékaře, či během předoperačního vyšetření. Vyšetření vypovídá o stavu kardiovaskulárního systému sledovaného pacienta v době náběru vyšetření.

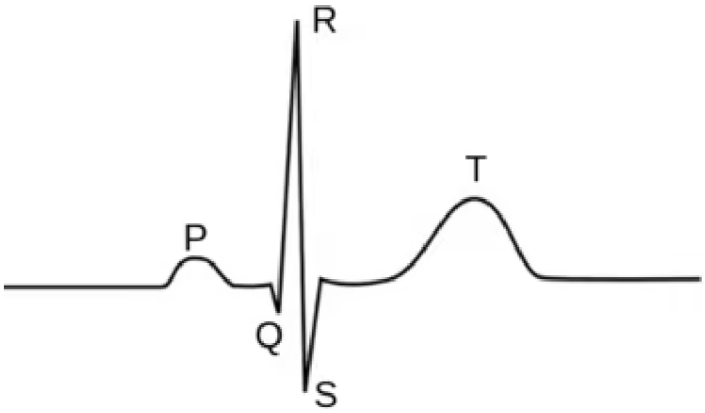
EKG vyšetření se zaznamenává na elektrokardiograf. Křivka EKG je záznam průběhu změn povrchového napětí na kůži pacienta generovaného srdeční aktivitou, který lze sledovat pomocí elektrod připevněných na kůži pacienta. Protože srdce je sval, tak je třeba pro spolehlivou a přehlednou práci s EKG signálem nutné aplikovat na signál řadu filtrů signálu pro odfiltrování jiné svalové aktivity, případně dalšího rušení. (Hampton, 2007 - 1-3)

Hodnocení EKG probíhá čtením z EKG křivky, která je vyobrazením cesty elektrického impulsu srdcem podle sousledu činnosti částí srdce. Hodnocení probíhá na ose X podle vzdálenosti jednotlivých událostí, které se uvádí v milisekundách a sleduje se vzdálenost mezi jednotlivými částmi každého srdečního cyklu, které jsou vyjádřeny

amplitudou na ose Y. Hodnoty odečítané na ose Y, znázorňují sílu naměřeného signálu a podle těchto hodnot odečítaných v čase lze hodnotit stav sledovaného pacienta. (Hampton, 2013 - 10-12)

Pro normální signál zdravého jedince bez abnormalit se jedná o následující fáze:

1. Kontrakce Síní - Vlna P,
2. Depolarizace Komor - QRS komplex,
3. Repolarizace komor - Vlna T.



Obrázek 4 Srdeční cyklus (SUNIT, 2019)

4 Vlastní práce

Praktická část této bakalářské práce je věnována zpracování uživatelských požadavků pro vývojový nástroj, kdy výsledkem této syntézy je návrh řešení spolu s přikládaným algoritmem řešení problému, v konkrétním bodě vývoje nových funkcí zpracování EKG signálu takovým způsobem, aby nástroj vyhovoval uživatelům. Pro zpracování této části práce jsou vybrány metody softwarového inženýrství. K implementaci, díky menšímu rozsahu projektu, budou využity prvky metody Waterfall, které byly popsány v předešlé kapitole této bakalářské práce.

4.1 Popis softwarových nástrojů pro práci s EKG signálem

Lékařský personál používá k hodnocení EKG záznamů kromě papírových výtisků specializovaný software. Tento software může sloužit jako úložiště zaznamenaných vyšetření na EKG přístrojích ve zdravotnickém zařízení, a plnit tak úlohu informačního systému. V praxi ovšem zdravotnická zařízení využívají v provozu pouze jeden centrální informační systém, který sdružuje další specializované softwarové produkty, případně v různé hloubce napojení i přímo lékařská zařízení. V takto napojených systémech bývá krom přenosu dat o pacientech řízen i přístup zaměstnanců a další provozní skutečnosti. Dále například napojení na systémy zdravotních pojišťoven pro uskutečnění plateb za provedenou péči, nebo obědové menu místní jídelny. Specializovaný software má několik standardizovaných možností jakým způsobem se na centrální informační systém napojit a přenášet obousměrně data.

Jedním takovým specializovaným softwarem je i software určený pro nabírání EKG a následnou práci s EKG daty a jejich archivaci. Pro práci s EKG signálem slouží specializované moduly, které zobrazují ve standardizované mřížce a velikosti EKG signál, a dávají lékařům možnosti pro práci s EKG signálem. Jednou z těchto možností je např. úprava velikosti mřížky a celkového zobrazení, dále například změna pojmenování svodového systému. Měření určitých segmentů zobrazených signálů srdečního cyklu v průběhu času udávaného v milimetrech na sekundu záznamu, nebo síly naměřeného potenciálu v amplitudě udávané v milivoltech. Dále tento druh softwaru nabízí více či méně komplexní automatickou analýzu rozměření EKG signálu. Takový druh automatické

diagnostiky je určitý druh nápovědy pro lékaře, slovo nápověda není zvoleno náhodou, ale je to z důvodu legislativního pohledu, kdy každý závěr po hodnocení EKG signálu, a hlavně jeho interpretaci, musí vytvořit lékařský personál dle svého úsudku. Automatické rozměření se provádí analytickými či statistickými metodami a bývá většinou zobrazeno formou tabulky a váže se k danému vyšetření a pacientovi.

4.2 Popis sledovaných ukazatelů v praktické části práce

V praktické části práce bude zpracováno několik ukazatelů, které nyní základně přiblížím. Jedná se o parametry, které určují například počátek a konec určité epizody v čase průběhu signálu. Například v případě pohledu na vlny signálu se jedná o údaj náklonu osy této vlny.

4.3 Uživatelské požadavky pro softwarový nástroj

Vývojový tým pracující na vývoji automatické diagnostiky EKG signálu za použití algoritmizace, či strojového učení, zadal požadavek pro jinou část vývojového oddělení zabývající se správou a zpracováním dat na podpůrný nástroj pro ulehčení práce s daty během vývoje nové funkce. Jedná se tedy o požadavky na interní nástroj, který je zamýšleno použít pouze pro jeden druh úkonu v jednom daném projektu.

K analýze uživatelských požadavků jsem se rozhodl využít metod softwarového inženýrství. V tomto případě uživatelské požadavky vzešly z brainstormingu nad nastoleným problémem ve skupině vývojového týmu. Problém byl představen jako nutnost sumarizace obsahu dat v určité množině souborů ve formátu XML (dále jako datasada), dostatečně přehlednou a jednoduchou formou. Zároveň velmi důležitým parametrem, již od začátku nastaveným, byla nutnost minimalizace použití zdrojů k vytvoření řešení pro popsaný problém.

Během řešení hloubky přístupu z hlediska uživatele bylo zjevné, že nástroj nemusí splňovat kritéria aplikace s uživatelským rozhraním, a může jít o nástroj v podobě algoritmu,

zpracovaného ve funkcích určitého bloku kódu, s použitím již existujících metod a balíčků třetích stran pro zjednodušení a zefektivnění procesu implementace.

Dále bylo během brainstormingu řešeno, zda logovat činnost takového algoritmu a do jaké hloubky. Výsledkem debaty byla shoda nad řešením splňujícím alespoň základní pravidla pro splnění bezpečnosti práce s daty a zabránění například špatné interpretaci. Logování algoritmu by mělo pokrývat informaci o nezpracovaném záznamu a informaci takovou uložit ve snadno čitelném a jasně identifikovatelném formátu v textovém souboru na stejné místo jako výsledný soubor Microsoft Excel s výsledky algoritmu.

4.3.1 Seznam konečných uživatelských požadavků v bodech:

1. Potřeba rychlého zjištění obsahu dat v datasetu ve sledovaných ukazatelích.
2. Základní statistické zpracování zjištěných hodnot z EKG dat.
3. Přehled statistiky dat - pokud jsou známa.
4. Výsledky zobrazit formou reportu v souboru Microsoft Excel.
5. Vstupními parametry by z důvodu jednoduchosti použití měly být pouze cesta k adresáři s daty a cesta uložení výsledného reportu.
6. Není nutné měnit druh použité metodiky zpracování.
7. Logování neúspěšně zpracovaných souborů.
8. Zpracovat věk pacientů a určit:
    - průměrný věk pacientů,
    - nejmladší pacient,
    - nejstarší pacient.
9. Z XML souboru číst a pracovat zejména s daty z elementu “globalResults“, seznam dat:

CalculatedFrom_05ms	QRS_Duration_05ms
CalculatedTo_05ms	QT_Duration_05ms
HR_bpm	P_Axis_deg
RR_AvgDistance_05ms	QRS_Axis_deg
P_Duration_05ms	T_Axis_deg
PQ_Duration_05ms	QTc_Duration_05ms

Tabulka 2 Seznam sledovaných parametrů. Vlastní tvorba.

Tabulka č. 2 zobrazuje výčet sledovaných parametrů určených ke zpracování.

Potřeba rychlého zjištění obsahu dat v datasetu ve sledovaných ukazatelích:

Zjišťováním potřeb bylo určeno, které konkrétní hodnoty je v datech nutné sledovat. Jedná se o obecné ukazatele určující základní parametry EKG signálu.

4.4 Návrh algoritmu na základě analýzy uživatelských požadavků

Uživatelské požadavky jsou znázorněny v diagramech, kde se řeší vztah uživatele k navrhovanému algoritmickému řešení, a řeší se také vztah jednotlivých komponent navrhovaného algoritmu. Během analýzy uživatelských požadavků nevznikl požadavek na vytváření databázové struktury pro ukládání mezivýsledků zpracovávaných dat navrhovaným algoritmem.

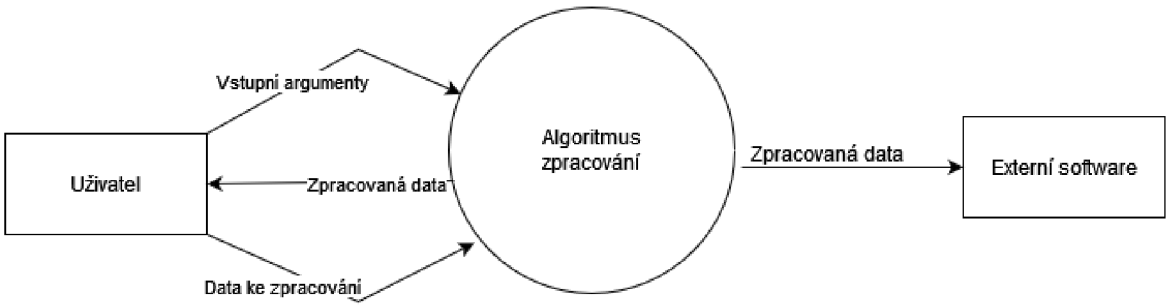
Pro zpracování diagramů a pro návrh řešení je nutné vydefinovat datové typy jednotlivých uvažovaných aspektů XML souboru určených ke zpracování.

CalculatedFrom_05ms - **Float**	QRS_Duration_05ms - **Float**
CalculatedTo_05ms - **Float**	QT_Duration_05ms - **Float**
HR_bpm - **Integer**	P_Axis_deg - **Float**
RR_AvgDistance_05ms - **Float**	QRS_Axis_deg - **Float**
P_Duration_05ms - **Float**	T_Axis_deg - **Float**
PQ_Duration_05ms - **Float**	QTc_Duration_05ms - **Float**

Tabulka 3 Datové typy. Vlastní tvorba.

Tabulka č. 3 zobrazuje předpokládané datové typy sledovaných hodnot.

4.4.1 Kontextový diagram DFD



Obrázek 5 Kontextový diagram – vztahy entit. Vlastní tvorba.

4.4.2 Konkrétní procesy v systému

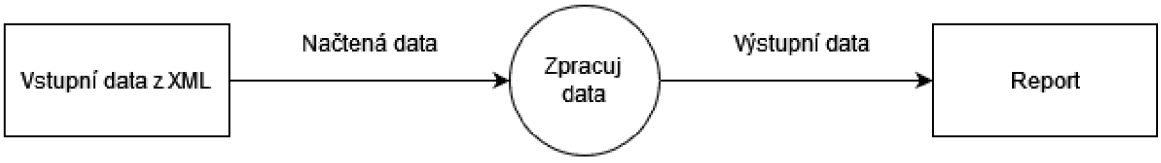
4.4.2.1 Vztah uživatele k algoritmu

Diagram označuje vztah uživatele k algoritmu, kdy uživatel na vstupu dává data ke zpracování a na výstupu odebírá zpracovaná data.

Obrázek 6 Diagram vztahu uživatele k algoritmu. Vlastní tvorba.

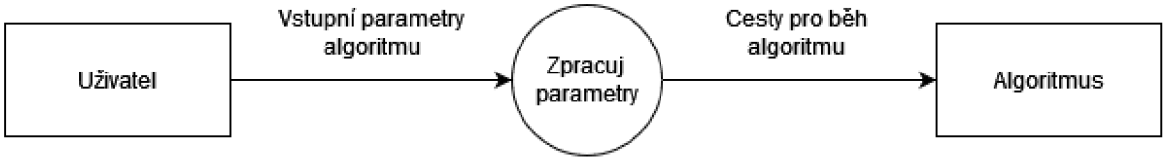
4.4.2.2 Vztah mezi komponentami algoritmu

Diagram cesty zpracování dat ze souboru formátu XML. Diagram popisuje vztah mezi funkcemi navrhovaného algoritmu.



Obrázek 7 Kontextový diagram – vztah mezi komponentami algoritmu. Vlastní tvorba.

4.4.2.3 **Parametry spuštění algoritmu uživatelem**



Obrázek 8 Kontextový diagram – parametry spuštění algoritmu uživatelem. Vlastní tvorba.

4.5 Implementace

Pro implementaci byl po zhodnocení všech požadovaných kritérií a nároků na algoritmus zvolen programovací jazyk Python, konkrétně verze 3.10. Tato volba byla učiněna na základě vhodnosti použití jazyka Python k práci s daty i s velkým počtem souborů, kdy toto východisko bylo převzato z vědomostí získaných vypracováním literární rešerše.

Samotná implementace probíhala na základě vydefinovaných požadavků a těchto zpracovaných do diagramů vztahů zamýšlených komponent vydefinovaného algoritmu.

K implementaci algoritmu byly vybrány následující balíčky třetích stran:

- Argparse – slouží k parsování argumentů z příkazu po spuštění.
- Pathlib - nabízí jednodušší a intuitivnější způsob práce s cestami k souborům a adresářům.
- Pandas - Balíček pro práci s daty včetně jejich načítání, zpracování a ukládání.
- Os – Tento balíček slouží pro předávání informací s operačním systémem.
- Logging – Umožňuje snadné zapisování aktivity do souboru - logování.
- xml.etree.ElementTree – Baliček pro práci s XML soubory. Umožňuje parsování, úpravu dat, ukládání a mazání.

Pro otestování funkcí algoritmu byly použity Unit testy, které jsou součástí samostatného souboru a pokrývají kód implementovaného algoritmu. Testy byly tvořeny metodou „Arange, Act, Assert“, tedy v prvním kroku dochází k přípravě očekávaných dat pro

porovnání, poté se použitím testovacích dat zavolá testovaná funkce algoritmu a v poslední fázi dochází k porovnání výsledků vrácených funkcí s očekávaným výsledkem připraveným v první fázi testu.

Seznam unit testů:

- test_convert_to_int_column
- test_convert_to_float_column
- test_convert_to_numeric_mixed_column
- test_count_xml_files
- test_results_to_xlsx
- test_parse_xml_file

K implementaci testovacích metod byly vybrány tyto balíčky:

- Xlrd – Balíček pro práci s Microsoft Excel.
- Tempfile – Umožňuje snadnou práci s dočasnými soubory.
- Numpy – Vědecký balíček, knihovna nabízející širokou škálu možnosti práce s matematickými funkcemi.

4.6 Uživatelský test algoritmu

Po fázi implementace je nutné zpracovaný algoritmus podrobit uživatelskému testování pro zaručení správnosti fungování dle návrhu. V případě takového typu algoritmu je třeba brát zřetel na vyhodnocení výsledků testování.

4.6.1 Testovací procedury

Vydefinované testovací procedury pro dosažení požadované kvality:

- Ověření správného vytváření obsahu.
- Ověření správné implementace algoritmu sumarizace informací o datasadě.
- Ověření zpracování parametru vstupní cesty k souborům a výstupní cesty pro Excel.

4.6.1.1 **Ověření správného vytváření obsahu**

Výstupem běhu algoritmu je soubor, který je ve formátu xlsx a obsahuje dvě záložky „Describe“ a „Data Types“. Jedná se tedy o správné výstupní tabulky. Během nad známou, neboli testovací datasadou, bylo ověřeno, že jsou tabulky správně plněny výstupními daty.

	CalculatedFrom_05ms	CalculatedTo_05ms	HR_bpm	RR_AvgDistance_05ms	P_Duration_05ms	PQ_Duration_05ms
count	75	75	75	75	51	51
mean	1386.666667	11840	89.66666667	972.8533333	213.1764706	317.8823529
std	4079.922274	9539.448678	50.18515268	1053.487627	78.38487249	61.1115855
min	0	0	9	256	20	216
25%	0	0	55	556	180	280
50%	0	16000	78	769	200	316
75%	0	16000	108	1091	252	340
max	16000	32000	234	6968	396	484

QRS_Duration_05ms	QT_Duration_05ms	P_Axis_deg	QRS_Axis_deg	T_Axis_deg	QTc_Duration_05ms
75	69	51	75	69	69
204.3733333	775.9710145	17.88235294	48.90666667	7.405797101	848.5797101
68.50108962	114.8705116	66.36946498	54.76390934	73.4663426	150.8232707
124	492	-125	-128	-167	276
162	688	-46	40	-55	810
184	778	48	55	35	848
212	846	65	81	48	922
436	1056	106	171	146	1254

Tabulka 4 Ukázka hodnot tabulky ze záložky "Describe". Vlastní tvorba.

Popis tabulky č. 4: Tabulka je rozdělena do dvou řádku z důvodu čitelnosti. Na zachycené podobě tabulky je možné odečítat výstupní hodnoty algoritmu pro vyčleněnou testovací datasadu.

CalculatedFrom_05ms	float64
CalculatedTo_05ms	float64
HR_bpm	int32
RR_AvgDistance_05ms	float64
P_Duration_05ms	float64
PQ_Duration_05ms	float64
QRS_Duration_05ms	float64
QT_Duration_05ms	float64
P_Axis_deg	float64
QRS_Axis_deg	float64
T_Axis_deg	float64
QTc_Duration_05ms	float64

Tabulka 5 Ukázka hodnot tabulky ze záložky "Data types". Vlastní tvorba.

Popis tabulky č. 5: Na zachycené podobě tabulky je možné odečítat výstupní hodnoty algoritmu pro vyčleněnou testovací datasadu.

4.6.1.2 **Ověření správné implementace algoritmu**

Ověření proběhlo spuštěním Unit testů zapsaných jazykem Python a které testují metody implementovaného algoritmu.

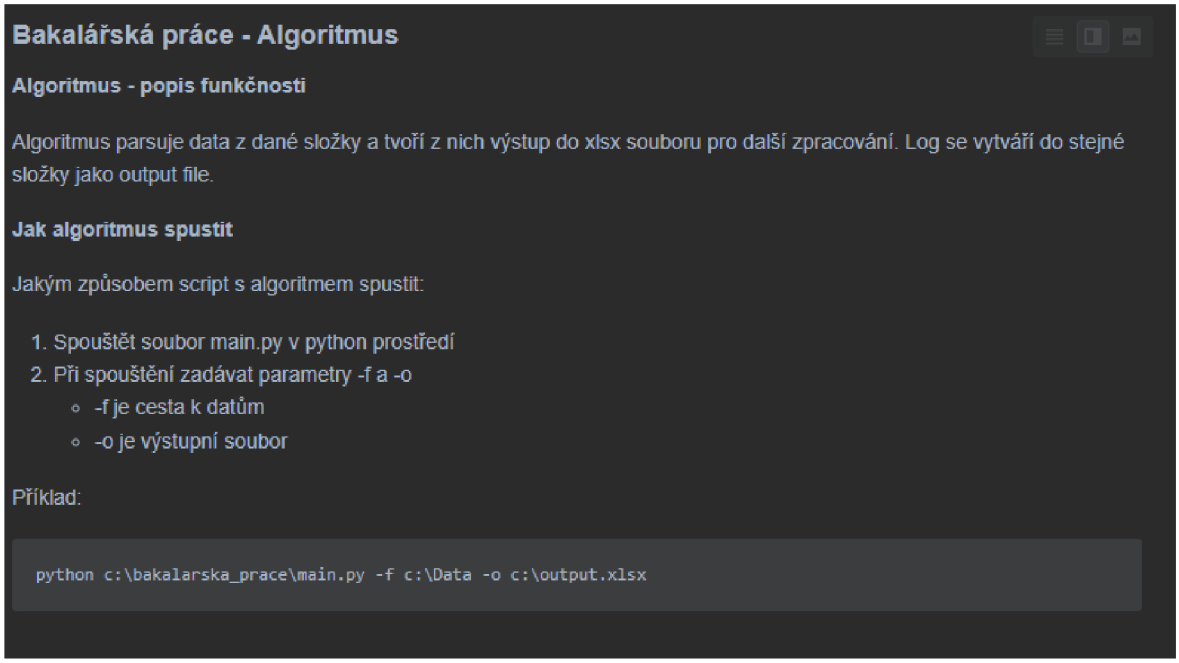
4.6.1.3 **Ověření zpracování parametru vstupní cesty k souborům**

Spuštěním algoritmu s parametry „-f *path* –o *path*“ došlo díky úspěšnému běhu algoritmu k ověření zpracování vstupního parametru pro čtení zdrojových dat, tedy parametru –f a cesty pro výstupní soubor, tedy parametru –o.

4.7 Předání algoritmu vývojovému týmu

Po úspěšném dokončení všech předchozích fází vývoje algoritmu je poslední fází podle modelu Waterfall nasazení. V případě této práce se jedná o předání algoritmu vývojovému týmu, který o něj požádal. Předání znamená předání funkčnosti jako takové spolu s dokumentací během fází vývoje tvořenou. Poté je možné hotový kód zapojit do většího celku a tím je celý proces vývojového cyklu dokončen.

Předávaný kód splňuje standardy programování, a tedy kód je dostatečně okomentován, tak aby bylo jasné, k čemu daný blok kódu slouží a jakou činnost má vykonávat. Dále se jedná o popis celkové funkčnosti algoritmu obsahující nejen požadované chování, ale také způsob obsluhy dodávaného řešení. Tento popis je přiložen jako samostatný soubor k algoritmu s názvem README.md a je zapsán v jazyce Markdown z důvodu formátování. V neposlední řadě se také jedná o samotný kód testovacích procedur, který také jako kód algoritmu musí být dostatečně popsán a mít svou dokumentaci zejména s popisem jednotlivých testovacích metod s popisem co testují.



Obrázek 9 Ukázka popisu funkčnosti algoritmu ze souboru README.md. Vlastní tvorba.

5 Diskuze

Během řešení dílčího cíle bakalářské práce jsem se zabýval literární rešerší na téma vývoje softwaru, kdy jsem se věnoval tradičním a agilním metodikám řízení projektů. Během studia odborné literatury jsem si udělal poměrně silný obrázek o teoretické stránce obou v práci zmiňovaných implementací dvou hlavních proudů metodik vývoje software, a to Scrum a Waterfall. Když jsem tyto získané zkušenosti porovnal se zkušenostmi z reálného prostředí práce ve firmě zabývající se vývojem softwaru, tak jsem došel k závěru, že je složité tyto teoretické poučky přenášet do praxe. Obzvláště složité jsem vyhodnotil uplatnění čistě agilních metodik ve společnostech, které na to v některých svých částech, například projektech nebo odděleních, nejsou připravené a stále se nedokáží na tuto metodiku adaptovat.

Podle mého názoru se nejedná jen o samotné členy vývojových týmů, ale hlavně o jejich manažery, kteří z různých důvodů takové transformace odmítají. Také jsem si uvědomil složitost práce Product Ownera a Scrum Mastera, kteří mají do velké míry své role velmi podobné a musí pro ně být velice těžké se své role držet a vykonávat činnosti, které mají prioritu. Samozřejmě je také velice důležité umět tyto priority rozpoznat a správně předávat.

Vzhledem k tomu, že mne tato problematika zaujala, tak se domnívám, že to je část mé bakalářské práce, která by si zasloužila v budoucnosti případnou větší pozornost. Myslím si, že je stále mnoho společností, které se snaží díky velikosti projektů přecházet z tradičních metodik na ty agilní, ať už v jakékoli implementaci. Takový přerod řízení projektů by mohlo být velice zajímavé téma ke zpracování například pohledem každé zúčastněné strany na nově adaptovanou metodiku.

6 Závěr

Hlavním cílem bakalářské práce bylo vytvořit softwarový nástroj pro pomoc s vyhodnocováním EKG záznamů pro vývoj automatické diagnostiky EKG signálu s využitím metod softwarového inženýrství. Dílčími cíli práce bylo vytvořit literární rešerši na téma vývoj software, analyzovat uživatelské potřeby a vytvoření modelu softwarového nástroje pro usnadnění vývoje automatické diagnostiky a vyvinout takový algoritmus, který by tento komfort vývojovému týmu dopřál.

V literární rešerši jsem se věnoval studiu odborné literatury na téma vývoje softwaru a samotné problematice EKG. V oblasti problematiky vývoje software jsem se věnoval zejména tradiční metodice Waterfall a inkrementální metodice Scrum. Studoval jsem nejen samotnou metodiku a její pohled na vývoj softwaru, ale také jednotlivé fáze v životním cyklu softwaru takto vyvíjeného. Literární rešerše popisuje oblíbenost programovacích jazyků pro práci s daty. Z důvodu zaměření práce jsem vyhodnotil jako důležité se věnovat také již zmíněné problematice zpracování EKG signálu, jelikož je tato problematika se zadáním práce úzce spjata.

Během návrhu algoritmu pro usnadnění práce s daty pro vývojový tým bylo využito metody brainstorming, pomocí které došlo k vydefinování uživatelských požadavků. Tyto požadavky byly následně analyzovány a sepsány do bodů, aby posloužily pro návrh řešení. Návrh řešení je znázorněn pomocí diagramů a tabulky s výpisem očekávaných datových typů. V návrhu bylo také uvažováno testování algoritmu a výběr již existujících balíčků třetích stran pro snazší a levnější vývoj algoritmu. Dokumentace je tvořena popsáním samotného kódu algoritmu komentáři a krátkým popisem funkčnosti a příkladem použití algoritmu. Tyto informace jsou zapsány v přikládaném souboru README.md. Kvalita a správnost funkčnosti algoritmu je zaručena testovacími metodami, které pokrývají funkčnost samotného algoritmu. Výčet těchto Unit testů je uveden v kapitole Implementace.

Obecně lze konstatovat, že došlo k naplnění všech stanovených cílů, které byly zadány pro vypracování této bakalářské práce. Zejména byl navržen a kompletně implementován jednoduchý algoritmus vydefinovaný v průběhu práce splňující zadání.

7 Seznam použitých zdrojů

- MYSLÍN, Josef. *Scrum: průvodce agilním vývojem softwaru*. Brno: Computer Press, 2016. ISBN 978-80-251-4650-7. (Myslín, 2016)
- Kolektiv autorů, *Manifesto for Agile Software Development*, online dostupné z: http://agilemanifesto.org/ (Agile Manifesto, 2001)
- ELLIOTT, Geoffrey. *Global Business Information Technology: an integrated systems approach*, 2004 ISBN 0-321-27012-6 (Elliot, 2004)
- ROYCE, Winston W., *Managing the Development of Large Software Systems* - (Royce, 1970)
- SIMS, Chris a JOHNSON, Hillary Louise, *Scrum: a Breathtakingly Brief and Agile Introduction*, 2012 ISBN-13 978-1937965044 (Sims, Johnson, 2012)
- ŠOCHOVÁ, Zuzana a Eduard KUNCE. *Agilní metody řízení projektů*. 2. vydání. Brno: Computer Press, 2019 ISBN 978-80-251-4961-4. (Šochová, Kunce, 2019)
- MATEROVÁ, H. a VRUBLOVÁ Y., *Informatika a legislativa ve zdravotnictví*. Ostravská univerzita v Ostravě, 2013 ISBN 978-80-7464-434-6. (Materová, Vrublová – 2013)
- Kolektiv autorů - Association for the Advancement of Medical Instrumentation, *Guidance on the use of AGILE practices in the development of medical device software, Association for the Advancement of Medical Instrumentation 2012* ISBN 1–57020–445–4 (Kolektiv autorů - 2012)
- PRESSMAN, Roger S. *Software engineering: a practitioner's approach*. 7th ed. New York: McGraw-Hill Higher Education, 2010. ISBN 978–0–07–337597–7 (Pressman, 2010)
- SOMMERVILLE, Ian. *Software engineering*. 9th edition, Pearson Education 2011. ISBN 978-0-13-703515-1 (Sommerville, 2011)
- TANENBAUM, A. S. a BOS, H. *Modern operating systems*. Prentice Hall Press 2015 ISBN 978-0-13-359162-0. (Tanenbaum a Bos, 2015)
- KANER, Sam a kolektiv autorů. *Facilitator's Guide to Participatory Decision-Making* 2nd edition, 2007. ISBN 978-0-7879-8266-9 (Kaner a kolektiv, 2007)

HAMPTON, John R. *EKG stručně, jasně, přehledně*. Praha: Grada, 2013. ISBN 978-80-247-4246-5 (Hampton, 2013)

- KAGGLE, *State of Data Science and Machine Learning survey,* 2022, online dostupné z: https://www.kaggle.com/kaggle-survey-2022 (Kaggle, 2022)
- PYTHON, Dokumentace programovacího jazyku Python, nedatováno, online dostupné z: https://wiki.python.org/moin/NumericAndScientific/Libraries, https://brochure.getpython.info/ (Python dokumentace)
- YEPIS, Erin, *Comparing tag trends with our Most Loved programming languages,* 2023 online dostupné z : https://stackoverflow.blog/2023/01/26/comparing-tag-trends-with-our-most-loved-programming-languages/ (YEPIS, 2023)
- SUNIT, S. Ekka, *How to read ECG: Electrocardiogram Simplified,* 2019, online dostupné z: https://physiosunit.com/how-read-ecg/ (SUNIT, 2019)

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

9 Přílohy

- Soubory zpracovaného algoritmu v adresáři „Algoritmus“
  - main.py
  - README.md
  - Adresář „tests“ obsahující soubory: tests.py a BP – 3.xml
- V adresáři „Data“ soubory ve formátu xml pro ukázku práce algoritmu, konkrétně: BP – 0.xml, BP – 1.xml, BP – 2.xml, BP – 4.xml, BP – 5.xml, BP – 6.xml.