Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Agilní vývoj softwarové aplikace

Petr Konopa

© 2017 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Petr Konopa

Systémové inženýrství

Název práce

Agilní vývoj softwarové aplikace

Název anglicky

Agile development of software application

Cíle práce

Cílem práce je analýza, návrh a implementace softwarové aplikace na základě vybrané agilní metodiky. Dále bude provedena rešerše a porovnání tradičních a agilních metodik.

Metodika

Práce se bude sestávat z těchto částí:

1. Rešerše metodik
2. Výběr agilní metodiky
3. Plánování vývoje na základě zvolené agilní metodiky
4. Samotný vývoj softwaru a jeho implementace

Doporučený rozsah práce

30-40 stran

Klíčová slova

Tradiční metody, agilní metodiky, SCRUM, C#, framework .NET, vývoj softwaru

---

Doporučené zdroje informací

EELES, Peter a Peter CRIPPS. Architektura softwaru. Brno: Computer Press, 2011. ISBN 978-80-251-3036-0.

KADLEC, Václav. Agilní programování: metodiky efektivního vývoje softwaru. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

ROBINSON, Simon. C#: programujeme profesionálně. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.

SNELL, Mike. Microsoft Visual Studio 2015 unleashed. 3rd edition. Indianapolis, IN: Sams, 2015. ISBN 978-067-2337-369.

ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace. Praha: BEN – technická literatura, 1999. ISBN 80-860-5658-9.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 02. 2017

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci " Agilní vývoj softwarové aplikace " 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 15.03.2017 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Marku Píckovi, Ph.D. za rady, připomínky a metodické vedení bakalářské práce. Dále bych chtěl poděkovat své rodině, která mne po celou dobu studia nesmírně podporovala.

Agilní vývoj softwarové aplikace

Souhrn

Bakalářská práce se zabývá tématem návrhu a vývoje softwarové aplikace užitím agilní metodiky. Za tímto účelem byla pro získání rozsáhlejších informací k dané problematice provedena rešeršní část. Výsledkem této části je obecná analýza metodik a jejich rozdělení do dvou základních skupin – Tradiční a agilní metodiky.

Druhou polovinu práce tvoří vlastní praktická část. V jejím úvodu dochází k porovnání dříve popsaných metodik a samotnému výběru metodiky Scrum pro návrh a vývoj softwarové aplikace Rainbow Galaxy. Na základě zvolené agilní metodiky je sestaven realizační tým a naplánován vývoj.

Implementace vyvíjeného softwaru je zaznamenána a popsána dle Scrum metodiky z průběhu vývoje prvních třech iterací (sprintů - 0,1,2).

Klíčová slova

Tradiční metody, agilní metodiky, SCRUM, C#, framework .NET, vývoj softwaru

Agile development of software application

Summary

The bachelor thesis deals with the design and development of software applications using agile methodologies. For this purpose, to obtain extensive information about the issue there is conducted a research part. The result of this part is the general analysis of methodologies and their division into two groups - traditional and agile methodologies.

The second half of the work creates an own practical part. In its introduction there is a comparison of previously described methodologies and the selection of the Scrum methodology itself for design and development of a software application Rainbow Galaxy. Based on the selected agile methodology there is formed a team and planned a development.

Implementation of the developed software is recorded and documented according to Scrum methodology during the development of the first three iterations (Sprints – 0,1,2).

Keywords

Traditional methods, agile methodology, SCRUM, C#, framework, .NET, software development

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Softwarové inženýrství je oproti ostatním inženýrským odvětvím vcelku mladý obor. I když první zmínky zaměřující se na vývoj softwarových aplikací lze najít v polovině 60. let, softwarové inženýrství bylo oborem jako takovým uznáno teprve roku 1997 ve Spojených státech amerických. Jedná se o období, kdy svět zažil velký rozvoj logických obvodů a počítačů. S tím se zároveň zvýšil počet vyvíjených softwarových aplikací. Ty však měly nízkou kvalitu provedení, nebylo snadné je udržovat, či inovovat a velmi často docházelo k prodlužování času projektů z důvodu špatné produktivity práce programátorů. To vše vedlo k potřebě definování důležitých kroků (metodik), jejichž dodržování by dopomohlo k zefektivnění práce, tvorbě „bezchybné“ aplikace a snížení nákladů na vyvíjený software. Nejvíce se k tomu blíží agilní vývoj softwaru, který vychází z Manifestu agilního vývoje softwaru. Ten byl sepsán roku 2001, kdy se sešla skupina nejvýznamnějších představitelů, kteří definovali základních 12 principů agilního vývoje.

V počátcích této bakalářské práce, jež je zaměřena na vývoj softwarové aplikace, nahlédneme více do historie softwarového inženýrství a jednotlivě si popíšeme základní metodiky tradičního a agilního vývoje. Poté si metodiky porovnáme a vybereme jednu z nich, jejíž princip bude užit na vytvoření plánu pro vývoj softwaru. Dále pomocí tohoto plánu bude uskutečněn samotný vývoj aplikace Rainbow Galaxy.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této bakalářské práce je vývoj a implementace softwaru užitím agilní metodiky. Za tímto účelem bude provedena studie tradičních a agilních metodik, následovaná jejich porovnáním a výběrem jedné, která se aplikuje na samotný vývoj softwaru.

2.2 Metodika

K vytvoření bakalářské práce bylo nutné provést rešerši zvolené tématiky. Za tímto účelem je provedena studie knih a dostupných pramenů na téma tradiční a agilní metodiky. Na základě toho byla sepsána úvodní část práce tvořící rešerši. Dále práce navazovala na porovnání metodik, jejímž účelem bylo vyselektovat jednu konkrétní metodiku a tu následně aplikovat na vývoj softwaru. Další krok práce tedy navazuje na návrh projektu a jeho následnou implementaci pro vývoj softwarové aplikace Rainbow Galaxy.

3 Teoretická východiska

3.1 Softwarové inženýrství

Definice dle Fritz Bauera: „Softwarové inženýrství je zavedení a používání řádných inženýrských principů tak, abychom dosáhli ekonomické tvorby softwaru, který je spolehlivý a pracuje na dostupných výpočetních prostředcích“ [2]. Ekonomickou tvorbu softwaru (dále jen SW), jak jí definuje Kadlec, lze rozdělit na pět základních tříd, jenž je potřeba dodržet pro úspěšnou a ekonomickou tvorbu SW.

Ekonomická tvorba SW [2]:

1. Vhodné sestavení vývojového týmu
2. Výběr správného vývojového nástroje
3. Uvážit, zda část SW vyvinout či koupit
4. Nalézt společnou řeč se zadavatelem
5. Zvážit řešení budoucí údržby a rozšiřování programu

Vytvořený SW by měl účinně pracovat na dostupných technologických zařízeních a být spolehlivý.

3.1.1 Historie

Softwarové inženýrství je při porovnání s jinými inženýrskými odvětvími nováčkem, jehož mnohé části nejsou exaktně stanoveny a stále se v průběhu času dotváří. Vývoj lze rozdělit do těchto základních etap [2]:

- Období do poloviny 60. let minulého století, kdy se vytvářeli jednoúčelové programy, většinou neudržovatelné a neměnné, které byly uloženy v trvalé paměti. V tomto období nebyla použita žádná metodika.
- Na přelomu 60. a 70. let se objevují první náznaky, jež později vedli ke vzniku softwarového inženýrství. Vznikají pojmy „návrh shora dolů“. „modularita“ atd. V tomto období dochází i k softwarové krizi, kdy docházelo k nečekaným prodlužováním a prodražováním projektů.
- Softwarové inženýrství jako obor vzniká v 70. letech. V té době se hardware stává dostupnějším. Díky tomu se rozvíjí tvorba programů. Stále chybí určité zavedené postupy a tvorba SW není nijak řízena. Avšak ke konci

období se začínají využívat dnes známé techniky softwarového inženýrství, např. specifikace, návrh, architektura, testování, zajišťování kvality atd.

- V 80. letech s rozvojem softwaru dochází k rozkvětu softwarového inženýrství. Značný rozmach lze pozorovat u objektově orientovaných přístupů. Snaha ustoupit od jednorázových řešení a vznik a zdokonalování, komponentních technologií a architektur.
- Roku 1997 je s certifikátem v USA softwarové inženýrství uznáno jako obor

3.2 Tradiční metodiky

Představíme si tradiční metodiky životního cyklu při vývoji SW, které udály směr vzniku agilních metodik z důvodu přílišné složitosti a malé flexibility. Nestačily brát v úvahu požadavky nově příchozích projektů. Tradiční metodiky přestaly být vhodné pro malé projekty a týmy. Jako hlavní důvod byla brána přílišná dokumentace, revize a proces schvalování.

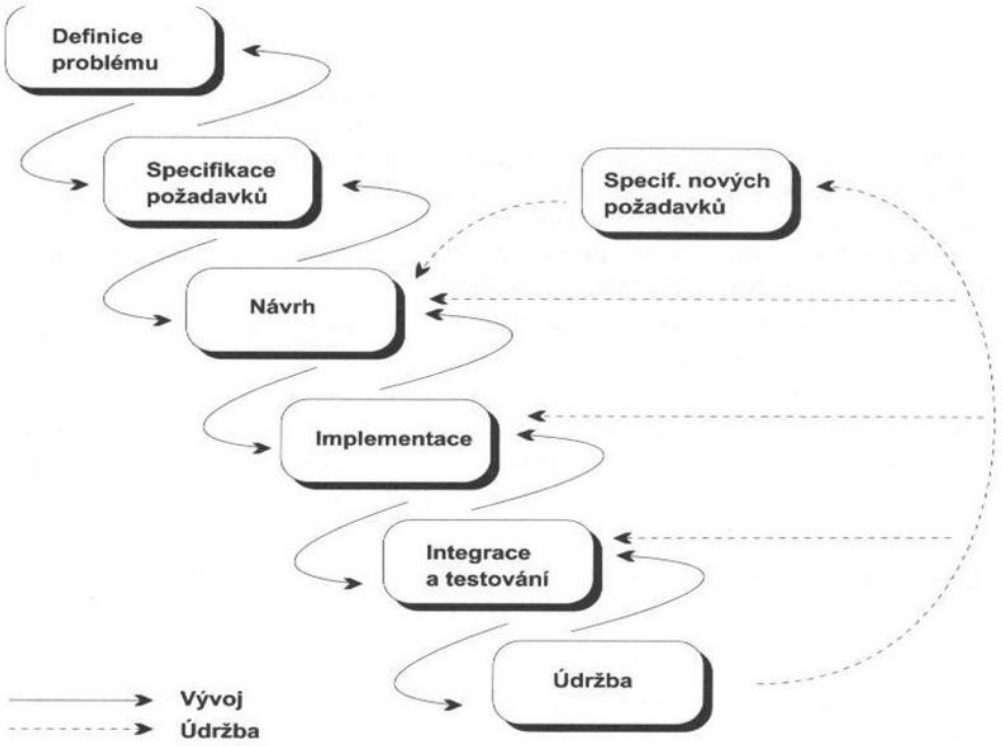
Z tradičních metodik se podíváme na vodopádový model, spirálový model a Rational Unified Process

3.2.1 Vodopádový model

Vodopádový model je nejstarší model životního cyklu SW. „Hledíme-li na něj obecně jako na posloupnost kroků a fází vedoucích k vytvoření kvalitního SW, můžeme považovat životní model za nejobecnější a zahrnout jej do srovnání“ [2]. Základním znakem modelu jsou sekvenčně seřazené fáze s žádnou nebo minimální iterací. K přechodu z jedné fáze na druhou dochází teprve tehdy, je-li po skončení jednotlivého kroku vždy vše řádně ukončeno schvalovacím procesem. Z toho vyplývá, že cyklus může vždy jen o jeden krok vpřed. Pokud v dalším fázi zjistíme chybu, je možné se vrátit vždy jen o krok zpět. Po opravení chyb je potřebné opět provést schvalovací proces. Důležité je zmínit, že zadavatel projektu nemá možnost v průběhu vytváření SW upřesnit požadavky. Úpravy již hotového SW se provádí až po skončení všech kroků projektu. Posloupnost základních kroků a fází je zobrazeno na obrázku 1.

Výhodou modelu je jeho jednoduchost (přímočarost). Je ideální pro řízení (pro přechod na další fázi musí dojít ke schválení předchozího kroku). Disciplína procesu vývoje (systematický postup mezi fázemi).

Avšak jednoduchost lze brát i jako nevýhodu, kdy nelze tímto modelem zvládnout rozsáhlejší a komplexnější projekty. Další negativum je výše zmíněná neschopnost zadavatele dodatečně přidávat a měnit požadavky. Zároveň může dojít k situaci, kdy analytik špatně pochopí zákazníka, či mu zákazník špatně formuluje přání a už nevytváří požadovaný produkt.



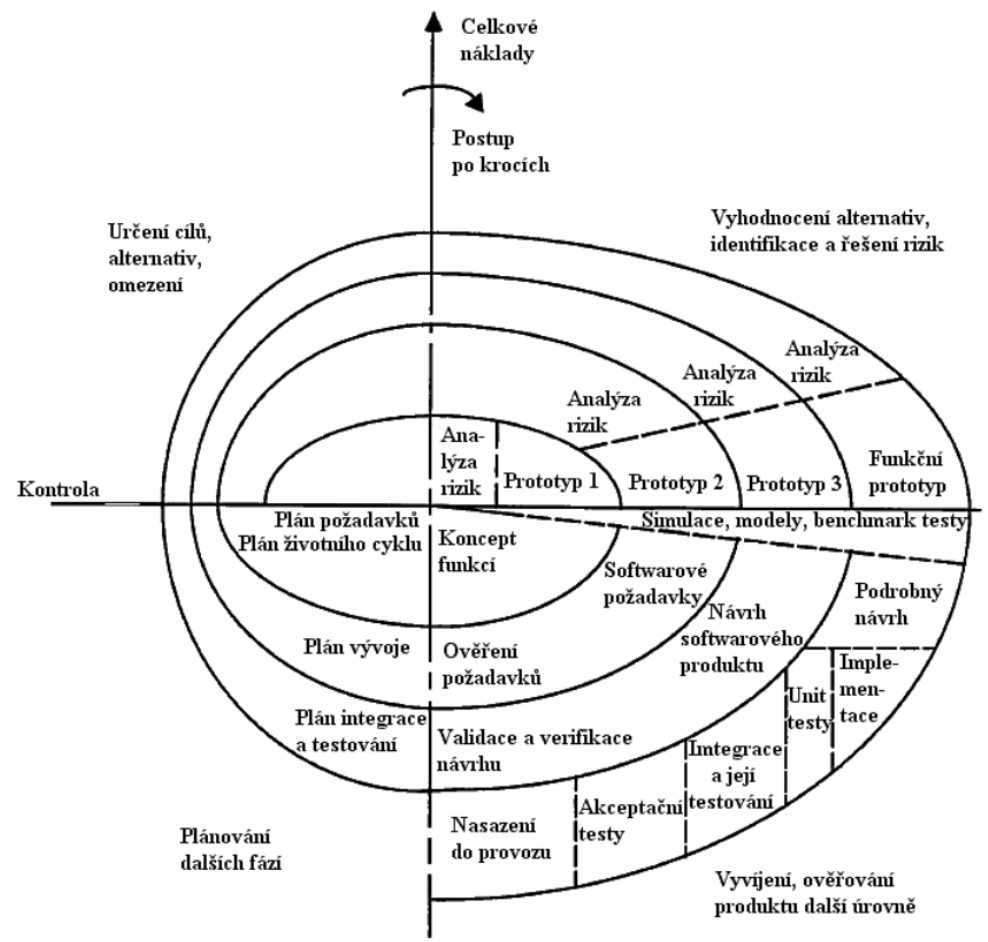
Obrázek 1: Schéma vodopádového životního cyklu [14]

3.2.2 Spirálový model

Vodopádový model přestal být dostačující, a tak v roce 1985 přichází Barry Boehm se spirálovým modelem, který však z vodopádového modelu vychází. Novější model přináší výhodu podrobné analýzy rizik a iterativní přístup a spadá do skupiny tzv. riziky řízených přístupů. Vývoj je postaven na analýze všech rizik a problémů. Od výsledku analýz se odvíjí další postupy při vývoji.

Kdykoli se průběh cyklu dostane do druhého kvadrantu, stanovují se cíle, alternativy, omezující podmínky apod. dalšího postupu. V prvním kvadrantu se provádí analýza rizik pro další iteraci. Ve čtvrtém kvadrantu dochází k realizaci a vytvoření prototypu pro daný cyklus. V posledním třetím kvadrantu dochází k plánování dalšího

postupu. Na obrázku č. 2 je základní popis jednotlivých kvadrantů a níže popis cyklů spirálového modelu.



Obrázek 2: Schéma spirálového životního cyklu [13]

Následující informace jsou čerpány z [2]. Popis cyklů spirálového modelu (obrázek 2):

1. V prvním cyklu je nutné nalézt globální rizika, jež by mohly ohrozit vývoj SW. Po splnění je dalším krokem zpracování základního konceptu vývoje a rozhodnutí o konkrétních použitých metodách.
2. V průběhu druhého cyklu se pak konstruuje a ověřuje specifikace požadavků na systém.
3. Třetí cyklus je zaměřen na vytvoření a ověření detailního designu.
4. Čtvrtý cyklus se týká implementace, testování a integrace.

Spirálový model není tak výhodný pro menší skupiny. Je pro ně komplikovanější a dochází ke ztrátě efektivity byrokratickou zátěží. Nejvýhodnější je tehdy, kdy zadavatel projektu je i jeho dodavatel z důvodu časté iterace. Dochází k menší časové ztrátě.

3.2.3 Rational Unified Process

Rational Unified Process (dále jen RUP) je komplexní metodika vyvinuta a dodávána firmou Rational Software, která ji dodává formou frameworku v podobě webových stránek. Po pořízení dostanete metodické pokyny, instrukce, šablony, příklady a procedurální aspekty a fáze vývoje SW. Základní charakteristikou této metody je, že je výrazně objektově orientována a náleží do skupiny tzv. přístupů řízených případy použití.

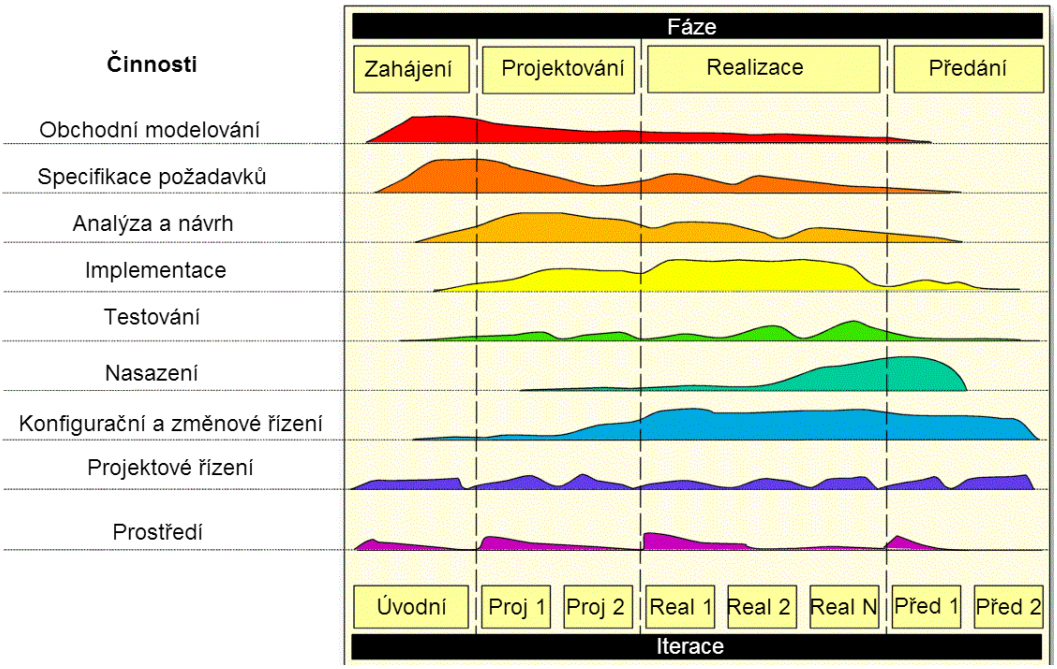
Vývoj v tomto prostředí probíhá v iteracích, kde se každá z iterací člení na čtyři fáze (obrázek 3):

- Zahájení – vývojáři formulují účel, rozsah projektu a obchodní kontext
- Projektování – definování základů architektury pomocí analýzy potřeb projektu, zákazníka
- Realizace – vývoj a tvorba zdrojových kódů a designu aplikace
- Předání – dochází k předání do dalšího vývojového cyklu nebo k zákazníkovi

Základem metodiky RUP při vývoji SW je používání šesti základních praktik [2]:

1. *Iterativní vývoj SW* - Zahájení, projektování, realizace, předání
2. *Správa a řízení požadavků* - Žadatel komunikuje a průběžně upravuje, zpřesňuje své požadavky na produkt
3. *Použití komponentové architektury* - Komponenta je zapouzdřená část systému s přesně definovaným rozhraním a s definovaným chováním. Dochází k úspoře zdrojů při znovupoužití hotové komponenty.
4. *Vizuální modelování* - Model je zjednodušení reality a je vytvářen za účelem dokonalejšího porozumění systému. RUP používá standardní modelovací jazyk UML, který zjednodušuje komunikaci v týmu, zlepšení konzistence projektu.
5. *Průběžné zajišťování a ověřování kvality* – Je levnější a snazší opravit nesprávné rozhraní mezi moduly v době návrhu, než v době implementace

6. *Řízení změn* - Neřízené změny, které mohou nastat při vývoji nebo po předání, mohou vést k chaosu. Tím může dojít k narušení, či rozvrácení vývojového procesu.



Obrázek 3: Vývojový cyklus v metodice RUP [12]

3.3 Agilní metodiky

„Agilní metodiky pro řízení vývoje software jsou takové metodiky, které využívají agilní přístup, tedy pružně reagují na změnu, průběžně rozvrhují práci v průběhu vývoje a ověřují výstupy s uživateli. Proces vývoje je díky agilnímu přístupu postavený na týmové spolupráci, otevřené komunikaci týmu, zapojení zákazníka a celkové flexibilitě a otevřenosti změnám“ [7]. To vše pro dodání fungující aplikace.

3.3.1 Manifest agilního vývoje softwaru

V roce 2001 se sešli nejvýznamnější představitelé nových přístupů k tvorbě SW (nejznámější: Alistar Cockburn, Kent Beck, Martin Fowler a další), kteří se shodli na základních tezích agilního vývoje, z nichž vzešel Manifest agilního vývoje.

„Objevujeme lepší způsoby vývoje software tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:

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

Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více“ [11].

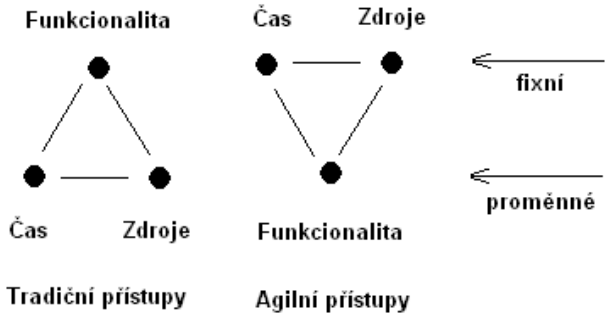
Dále definovali dvanáct principů agilního vývoje SW [11]:

1. Hlavním bodem je vyhovět zákazníkovi a dodávat hodnotný SW průběžně a včas.
2. Změny, které vedou ke zvýšení konkurenceschopnosti zákazníka jsou agilními procesy podporovány. Proto jsou úpravy v požadavcích vítány v jakékoli části vývoje.
3. Fungující SW dodávat v co nejkratších časových úsecích (týdny, měsíce). Kdy dáváme přednost nejmenší periodě.
4. Dodržení každodenní spolupráce lidí z byznysu a vývoje do ukončení daného projektu.
5. Vytvářet projekty pro motivované zaměstnance. Udržet jejich motivaci podporou jejich potřeb, vytvářením příjemného prostředí a důvěrou, že splní zadanou práci.
6. Sdělování informací vývojovému týmu je nejúčinnější a nejefektnější při osobním kontaktu.
7. Fungující SW je hlavním měřítkem pokroku a úspěchu.
8. Podporování udržitelného rozvoje pomocí agilních procesů. Všichni lidé zapojení do projektu by měli trvale držet stejné tempo.
9. Trvalou pozornost k technické dokonalosti a dobrý design nám zlepšuje agilitu.
10. Klíčovým prvkem je jednoduchost, kde je zapotřebí maximalizovat množství práce, jež není potřebná.
11. U samo-organizujících se týmů vznikají nejlepší nápady.
12. Tým v pravidelných intervalech přemýšlí, jak zvýšit efektivnost a provede změny, k tomu potřebné.

Ze čtyř výše uvedených základních bodů a dvanácti principů vychází veškeré agilní metodiky.

3.3.2 Principy agilních metodik

Na obrázku 4 si porovnáme a řekneme základní rozdíly mezi tradičními a agilními programovacími přístupy.



Obrázek 4: Rozdíl mezi tradičními a agilními přístupy [15]

Na začátku tradičního vývojového procesu jsou stanoveny požadavky, které se nemění. Jako měnitelný je tu čas a zdroj. Víme, co bude program umět, ale už s jistotou nevíme za kolik to bude a jak dlouho bude vývoj trvat.

U agilního přístupu jsou zadavatelem zadány jako neměnné zdroje a čas. V průběhu prací se se zákazníkem jedná a přehodnocují se priority, proto je zde funkcionalita proměnná. Pro tento přístup se stanový nejdelší možný čas vývoje a nejvyšší náklady.

Základní principy agilních metodik [2]:

- *Vývoj s krátkými iteracemi* - zákazník je průběžně seznámen s aktuálním vývojem
- *Důraz na komunikaci v týmu* - snaha integrovat a zařídit fungující komunikaci v týmu
- *Stálá komunikace se zadavatelem* - v ideálním případě je zadavatel člen vývojového týmu a neustále s ním komunikuje, podílí se na návrhu a testech
- *Průběžné automatizované testování* - testy by měli být automatizované a v důsledku častých změn by mělo docházet k průběžnému testování správnosti systému.

Díky agilnímu manifestu docházelo ve vcelku krátké době ke zvyšování počtu agilních metodik. Přehled nejvýznamnějších metodik: Adaptivní vývoj SW, Vlastnostmi řízený vývoj, Extrémní programování, Lean Development, SCRUM Development Process, Crystal metodiky, Dynamic System Development Method a Testy řízený vývoj.

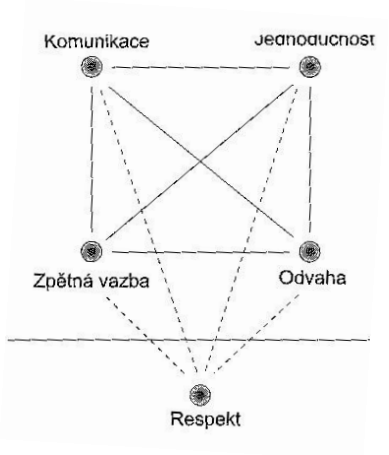
3.3.3 Extrémní programování

Extrémní programování (dále jen XP) používá běžné principy a postupy, avšak jejich užívání dotahuje do extrému. Například, osvědčí-li se jednoduchost pro funkčnost, nechá se systém v nejjednodušší podobě. Osvědčí-li se kontroly zdrojového textu, kód se bude revidovat neustále atd. Tím dochází ke zlepšení produktivity celého životního cyklu a ke zlepšení reakce na změny a snížení projektových rizik. „Jediným cílem XP je vyvíjet SW plnící bezvadně své funkce ve stanovených termínech“ [6].

Při tvorbě zadání u XP jsou tradiční proměnné mírně upraveny a rozšířeny o čtvrtou proměnnou na níž je kladen největší důraz. Zde si volí manažeři, zákazníci, zadavatelé hodnoty libovolných proměnných. Hodnotu poslední čtvrté proměnné si zvolí vývojářský tým. Pokud zadavatelé budou chtít určit všechny čtyři hodnoty, tak Kent Beck [6] uvádí, že výsledkem bude nekvalitní a zpožděný SW. Proměnné XP jsou:

- Kvalita
- Čas
- Náklady
- Šíře zadání

XP se řídí a staví na čtyřech základních hodnotách, jež jsou: komunikace, jednoduchost, zpětná vazba, odvaha a podprahová hodnota respekt, které jsou navzájem propojené, jak můžeme vidět na obrázku 5.



Obrázek 5: Vztah základních hodnot v XP [2]

K chybám a zpoždění projektu dochází při špatné komunikaci. K té dochází především ze strachu z nadřízeného, či v pozapomnění sdělit někomu jinému důležitou informaci.

Další hodnotou je jednoduchost, kdy je snazší udělat jednoduchou věc co nejdříve. Než se zdržovat se složitější, která by se v budoucnu mohla nakonec stejně změnit.

U zpětné vazby je důležité zjištění aktuálního stavu vyvíjeného SW, o požadavcích a potřebách zákazníka.

V neposlední řadě je odvaha, kdy je důležité učinit zásadní rozhodnutí. Např. zjistíte, že není možné pokračovat dál a musíte předělat větší část zdrojového kódu.

Poslední a podprahovou hodnotou je respekt, kdy je důležité komunikovat s ostatními a zjistit, zda jim nelze nějak pomoct. Pokud si členové týmu nebudou pomáhat, bude XP nepoužitelné.

Základní hodnoty z minulého odstavce tvoří základní rámec XP, avšak jsou příliš obecné a nemohou stačit k vytvoření užitelné metodiky. Tady Kent Beck [2][6] definuje 12 konkrétních postupů k tvorbě kvalitního SW produktu. Postupy se navzájem vyžadují a podporují. Při nedodržení těchto bodů se nejedná o metodiku XP.

1. *Plánovací hra* - Spočívá v tvorbě karet zadání, kde si zadavatel určí požadovanou funkčnost. Díky tomu vytvoří vývojový tým hrubý plán, který se dále konzultuje se zákazníkem. Dochází k analýze potřeb zákazníka a vytváří se mu představa, co je možné splnit.
2. *Malé verze* - Jsou kompletní, funkční a dávají uživateli určitý užitek. Uvolňují se v krátkých časových rozmezích.
3. *Metafora* - Slouží ke snadnější komunikaci mezi týmem a zákazníkem. Vývoj je veden pomocí jednoduchého příběhu, jež vytváří snáze zapamatovatelné pojmy a názvy.
4. *Jednoduchý návrh* - Nejjednodušší návrh funkčního systému, který je v danou chvíli možné vytvořit.
5. *Testování* - Dochází k průběžnému a automatizovanému testování, do které jsou zapojeni i zákazníci. Ti testují funkcionalitu. Pokud dojde při testování k chybám, nejdříve se opraví. Poté se pokračuje vývojem dalších funkcí systému
6. *Refaktorizace* - Refaktorizací rozumíme restruktalizaci systému bez změn jeho chování. Jedná se tedy o zjednodušení a zpřehlednění kódu. V tomto kroku dochází k přidání nových funkcí. Při správné refaktorizaci lze dosáhnout odstranění duplicit, zdokonalení komunikace, vylepšení návrhu, zjednodušení systému atd.

7. *Párové programování* - Dvojice programátorů sdílí jeden počítač. Jeden se stará o implementaci metody. Druhý kontroluje a dohlíží, zda je daná implementace v souladu s globální strategií.
8. *Společenské vlastnictví* - Důležitý bod XP, kde za celý systém přijímají odpovědnost všichni členové týmu. V systému může kdokoli a kdekoli měnit kód. Tím dochází ke zrychlení projektu, avšak s tím přichází i potřeba znatelné koordinovanosti pomocí konfiguračního managementu.
9. *Nepřetržitá integrace* - Pro tuto integraci je dobré mít k dispozici alespoň jeden počítač k tomu vyhrazený. Systém by měl být integrován a testován nejméně jednou denně.
10. *Čtyřicetihodinový pracovní týden* - Pokud chceme, aby byl projekt správně řízen, je zapotřebí dbát na spokojenost lidských zdrojů, nesmíme je přetěžovat, aby nedělaly zbytečné chyby z únavy a nesoustředěnosti.
11. *Zákazník na pracovišti* - Pro fungující spolupráci je zapotřebí součinnost stávajícího zákazníka, aby dohlížel na vývoj systému. Podílel se na řešení otázek, spolupracoval na testech a řešil možné spory.
12. *Standardy pro psaní zdrojového textu* - Při vývoji pomocí XP se nevytváří téměř žádná dokumentace. Samotný zdrojový kód je zdrojem všech informací. Proto je zapotřebí, aby všichni členové týmu dodržovali pravidla pro psaní kódu. To znamená, že by se informace měli udržovat v dobře strukturovaném, čitelném a komentovaném textu.

Co se týče velikosti týmu a projektu je tato metodika vhodná pro menší skupiny lidí a menší projekty, kde je potřeba rychlý vývoj. Jelikož je žádoucí vyvíjet SW v co nejjednodušší a nejkratší formě, určitě shledáme výhodou, že lidi jsou rádi, když vidí blízký a konkrétní cíl, když SW funguje. Určitě nesmíme opomenout i snahu o spokojenost programátorů a nepřekračování délky pracovní doby (viz. bod 10.). Avšak nejjednodušší část se může stát zároveň nejsložitější. Lidé v dnešní době hledají jen složitosti, a tudíž mohou mít problém s vývojem SW nejlehčím možným způsobem. Při ostychu může být obtížné se zeptat kolegy o radu a pokračovat dále. U XP je důležité si zvyknout spolupracovat se skupinou lidí, odvrátit se od individualismu [2].

3.3.4 Lean Development

Autorem Lean Development je Robert Charette [8], jež vychází z principů, které byly používány v 80. letech pro zefektivnění výroby automobilů v Japonsku, kde se výrobní proces nazýval Lean Manufacturing. Základní vlastností metodiky je odstranění všeho nedůležitého.

Definování metodiky dle Kadlece [2]: „Lean Development vede k systematickému přístupu a k identifikování a eliminaci možných zdrojů plýtvání v průběhu celého vývojového procesu, který se pokouší dodat zákazníkovi perfektní produkt a splnit tak jeho požadavky.“ Základem metodiky je dovést vývojový proces do optimální, efektivní a kvalitativní podoby. Lean Development je více než ostatní metodiky zaměřen na strategii. Základním cílem metodiky je vyvíjet SW, za třetinu času, vystačit v projektu s třetinou rozpočtu, snížit tvorbu chyb o třetinu běžného množství. Není to metodika popisující způsob vývoje a přesné řízení týmu, nýbrž nabízí vícero pravidel a principů, kde při jejich dodržení zajistí efektivní a ekonomický výsledek.

Metodika stojí na deseti základních pravidlech [2][8]:

1. *Odstranění zbytečností* - Odstranit vše, co konečnému produktu nepřináší žádnou hodnotu.
2. *Minimalizování zásob* - Není nutné vyvíjet a vytvářet artefakty (dokumenty, prototypy, modely), které nejsou momentálně potřebné nebo se v budoucnu nepřiblíží k zákazníkovi. Je důležité udržet počet artefaktů na minimálním počtu, které jsou nutné pro udržitelnou hodnotu systému.
3. *Maximalizování toku (zkrácení vývojového času)* - Pokud je potřeba maximalizovat tok, rozčlení se proces do menších částí a provede se iterativní proces vývoje. Tento proces přichází i s výhodou, díky níž může zákazník průběžně sledovat prototypy a doplňovat požadavky za chodu.
4. *Vývoj tažený poptávkou* - Principem je provést všechna rozhodnutí co nejpozději, a to z důvodu, že zadavatelé mnohdy nejsou schopni definovat současné potřeby, natož budoucí. „Všechna klíčová rozhodnutí, která učiníme, by měla být provedena v nejzazším možném okamžiku, kdy už je nevyhnutelné rozhodnout se o dalším směřování projektu. Čím dříve učiníme rozhodnutí, tím se zvyšuje pravděpodobnost, že jej budeme muset později měnit“ [2].

5. *Rozhodovací pravomoc pro pracovníky* - „Vývojáři musí chápat, jak jejich práce přispívá celkovému cíli, musí vědět, co mají vykonat a do kdy, a musí mít možnost rozhodovat“ [8].
6. *Uspokojovat požadavky zákazníků* - Neúspěšnost projektu často stojí na neúplnosti, či nesprávnosti zadaných požadavků. Lepší způsob, než přimění uživatele podepsat rozsáhlou specifikaci projektu je vytvořit s ním partnerství. Zvyšuje se pravděpodobnost, že při kontaktu se zákazníkem dokáže tým do vývoje lépe zahrnout zákazníkovy požadavky.
7. *Zavést zpětnou vazbu* - Pokud není možné v úvodu projektu definovat všechny požadavky, tak zavedením zpětné vazby do vývojového procesu budeme tyto požadavky postupně doplňovat. V průběhu vývoje bude docházet ke změnám a k možným chybám. Z preventivního hlediska se nejdříve před zavedením do projektu píší testy pro dané komponenty.
8. *Odstranění lokálních optimalizací* - Je zbytečné marnit čas a energii na jednotlivé komponenty, pokud jejich optimalizace nepřinese celkové zvýšení efektivity systému.
9. *Partnerství s dodavateli* - Pro zákazníka je nejdůležitější výsledná hodnota. Pro zlepšení kvality a urychlení práce může být výhodné nakoupit např. hotové knihovny.
10. *Zavedení kultury pro neustálé zlepšování* - Tvorba lepších pracovních podmínek a motivování zaměstnanců k maximálním výkonům.

Z těchto uvedených pravidel vznikla pozměněná metodika Lean Development, jež definuje sedm principů. Jsou i tací, kteří zastávají pouze 7 principů. Jedním z nich je i Marry Poppendick [9]. Metodika Lean Development neudává komplexní popis vývoje SW. Ukazuje jen, čeho se vyvarovat a jakých postupů se při vývoji v deseti krocích držet. Splněním těchto kroků zaručuje rychlý a efektivní vývojový proces, kdy dojde k uspokojení zákazníkových požadavků.

3.3.5 SCRUM

První zmínka o agilní metodice s názvem Scrum se datuje od roku 1986. Pro tuto metodiku je důležitá komunikace se zákazníkem a maximální spolupráce mezi členy v týmu. Nejvhodnější je především pro malé týmy o pár členech, avšak je možné zapojit

více týmů. Úkolem lidí v týmu je každodenní přezkoumání realizovaných požadavků a zadání dalších kroků. Scrum se dokáže vyrovnat s měnícími se požadavky.

Vlastník produktu (Product Owner) zahájí vytvoření produktu ve Scrum. Sestaví seznam požadavků (Product backlog), který je seřazen podle důležitosti. Vývoj běží v iteracích nazývaných Sprinty. Vývojový tým si musí naplánovat množství práce, které během každého Sprintu musí stihnout. Tato práce je obsahem Sprint backlogu, která udává detailní popisek úkolů z Product backlogu v nejbližším období. „Součástí Sprintu jsou povinné schůzky. Plánovací schůzka (Sprint Planning), Denní schůzka (Daily Scrum), Vyhodnocení sprintu (Sprint Review) a Retrospektiva sprintu (Sprint Retropective), na kterých probíhají odhady práce, plánování činností, zpětná kontrola vykonané práce, přezkoumaní a vyhodnocení přírůstku – dokončení práce, úprava backlogu, návrhy na zlepšení. Pokud jsou stanovené cíle splněny, tak je produkt hotový a dokončen“ [10].

Scrum praktiky lze rozdělit na čtyři hlavní body: role, artefakty, činnosti a pravidla viz. Obrázek 6.

1. Role

Ve Scrumu mají lidé určitou roli, která má svou funkci. Metodika určuje tři základní role: Vlastník produktu, Vývojový tým a Scrum master, kteří dohromady tvoří Scrum tým. Tato skupina by měla být zcela soběstačná, sama si vede práci a organizuje se.

a) Vlastník produktu

Je člověk, který má na srdci zájmy zákazníka a zodpovídá za zákazníkovi investice. Má za úkol zadat projekt, rozhodnout o rozsahu, rozpočtu a termínech projektu. Seznamuje vývojový tým s dalším postupem, určuje požadavky, hledá řešení a stanovuje prioritní vlastnosti projektu. Vlastník produktu spolupracuje se všemi účastníky projektu. Hlavním úkolem je ověřování jejich plnění.

b) Scrum master

Má zodpovědnost za dodržování pravidel Scrumu. Jeho rolí je napomáhat v komunikaci mezi vlastníkem produktu a vývojovým týmem. Zároveň se snaží udržovat jejich činnost v mezích Scrumu. Má za úkol pomáhat týmu dodržet cíle, vyřešit problémy, inspirovat tým k lepším výsledkům a vést k samostatnosti. Usnadňuje práci týmu a je prostředníkem, který se snaží zavést hodnoty, principy a zvyklosti Scrumu.

c) Vývojový tým

Tým, ve kterém jsou obsažené všechny nutné profese. Členové tohoto týmu jsou profesionálové, kteří jsou schopni vyrobit přírůstek produktu během jednoho sprintu. Každý vývojový tým se samostatně rozhoduje a spravuje, nikdo jiný ho neorganizuje při jeho práci. Má za úkol předvádět výsledky své práce vlastníkovi produktu, předkládat návrhy na zlepšení, testovat a vyvíjet produkt. Během jednoho sprintu se tým musí věnovat jedině zadané práci a složení vývojového týmu se během toho nemění.

2. Artefakty

„Artefakty označují entity, které ve Scrum vznikají, mění se nebo se používají a které jsou užitečné pro zajištění transparentnosti a kontroly. Paří k nim Product backlog, Sprint backlog a přírůstek“ [10].

a) Product backlog

Je to seznam důležitý pro následující vývoj produktu, ve kterém jsou zahrnuty vlastnosti, funkce, požadavky, rozšíření, technické zlepšení, opravy a další práce. Položky na tomto seznamu jsou sepsány ve formě uživatelských příběhů (User stories) a mají určenou důležitost, popis a odhad. Vše se musí dokončit během jednoho Sprintu, a proto jsou rozděleny. Důležitost položek v seznamu se odráží od míry rizika, naléhavosti zpracování a od přínosu.

b) Sprint backlog

Popisuje práci vývojového týmu, která je potřebná pro splnění Sprintu a vytvoření přírůstku a která přinese uživateli očekávanou hodnotu. Vývojový tým smí dělat změny ve Sprint backlogu, protože denně sleduje položky, aktualizuje a upravuje seznam podle probíhajícího vývoje. Úkolem Sprint backlog je pomáhat vývojovému týmu zorientovat se v kolech a aktualizovat odhad času práce, kterou má za úkol vykonat.

c) Přírůstek

„Je to souhrn položek z Product backlogu, které byly dokončeny během minulých a současných Sprintů. Důležité je, aby byl Scrum týmem odhlasován jako „hotovo“ a byl po dokončení použitelný pro nasazení. To, zda je práce ve sprintu dokončena v odpovídající kvalitě si posuzuje každý tým sám. Přírůstky se doplňují, testují a sleduje se jejich kompatibilita“ [10].

3. Činnosti ve Scrumu

Scrum musí dodržovat předepsané činnosti, aby tím byla zajištěna transparentnost procesů a jejich kontrola, sledování nákladů. Díky předepsaným činnostem ve Scrumu, které jsou pravidelné a dodržují časový limity, nedochází k prodlevám a ztrátám času v průběhu plánování.

a) Sprint

Sprintem se rozumí iterace ve Scrumu, která je časově limitována a výsledkem je přírůstek produktu. Důležité je, aby všechny sprinty trvaly stejnou dobu. Sprinty na sebe vzájemně navazují, mají určený začátek i konec. Nový sprint vznikne po ukončení předcházejícího sprintu. „Sprint je složen z plánovací schůzky, denních schůzek, vlastních vývojových prací, vyhodnocení sprintu a retrospektivy. Obsahem sprintu je návrh cílového produktu, plán k realizaci cíle, samotná práce a výsledný produkt“ [10]. Z hlediska ovlivnění cíle sprintu nelze po jeho startu provádět změny. Pokud dochází k časové ztrátě vývoje, musí se kontaktovat zadavatel a konzultovat, které položky Product backlogu se ze současného sprintu mohou odstranit. V opačném případě se přidají nové položky.

b) Plánovací schůzka

V průběhu Sprintu se plánují činnosti, na kterých se podílí Scrum tým. Plánovací schůzka má časový limit, během níž se domlouvá budoucí přírůstek ve Sprintu. Dochází k vytvoření plánu práce, která je pro daný sprint nezbytná

c) Denní schůzka

Vývojový tým se každý den schází, aby projednal plán na dalších 24 hodin. Úkolem této schůzky je kontrola vykonané práce od poslední Denní schůzky. Po revizi minulého dne se odhadne a naplánují práce na den další.

d) Vyhodnocení sprintu

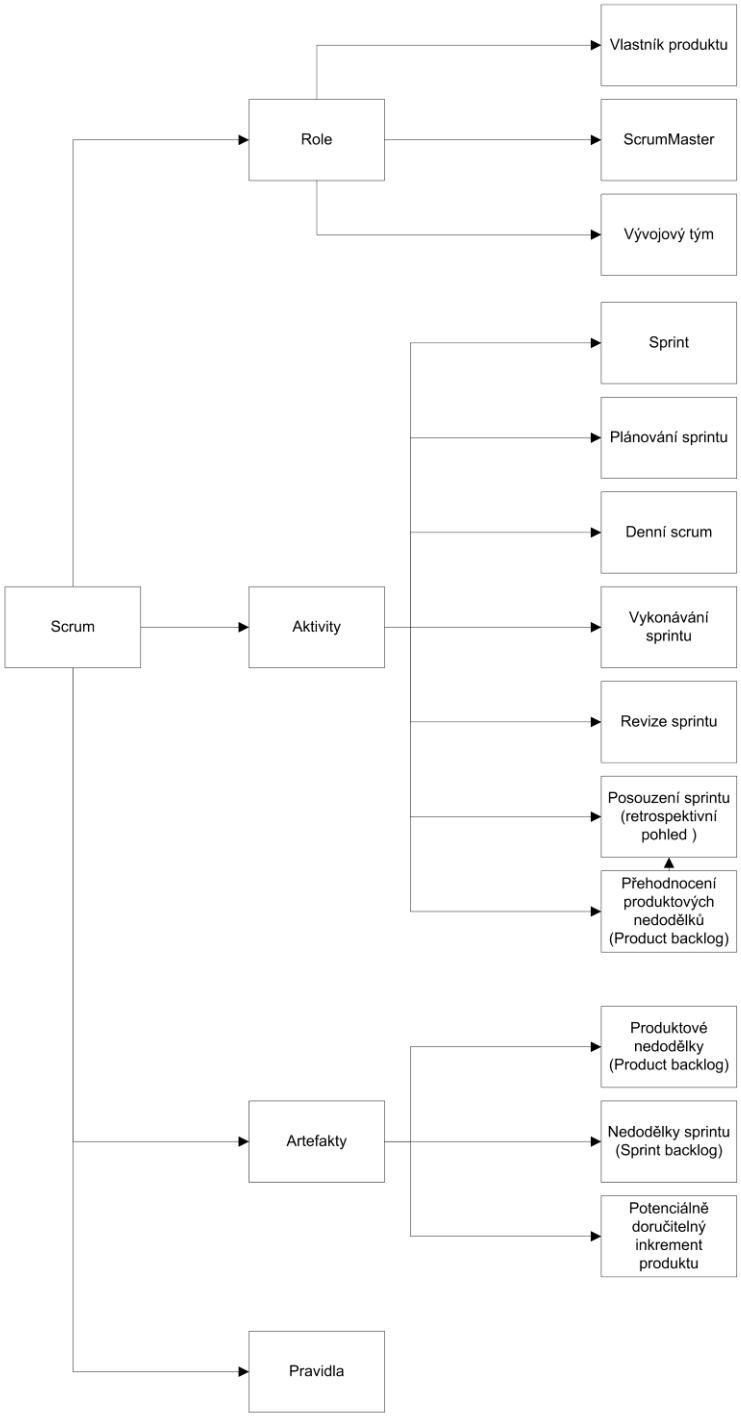
Je to neformální schůzka Scrum týmu s ostatními stakeholdery, kde je cílem přezkoumat přírůstek a v případě nutnosti poupravit Product backlog.

e) Retrospektiva sprintu

Zajímá se o použité procesy, nástroje a lidské zdroje. Plánuje kroky, které pomohou zlepšovat činnost týmu v následující iteraci. Kontroluje zpětně práci Srcum týmu.

Scrum zajišťuje efektivní využití času a peněz. Velké projekty jsou rozděleny do jednodušších, snadno zvládnutelných sprintů. Funguje dobře pro rychle se rozvíjející projekty. Důvodem je možnost flexibilně reagovat na změny při průběhu práce.

Díky denním schůzkám (Scrum meetings) dochází ke zjištění stavu projektu a následnému rozhodování, jak dále pokračovat, popřípadě změnění směřování projektu pro zefektivnění procesu. Scrum metodika je spíše jen souhrn vzorů, jak postupovat. Neudává specifika, díky nimž by se dal software vyvíjet.



Obrázek 6: Praktiky Scrum metodiky

4 Praktická část

V úvodu této kapitoly dochází k porovnání jednotlivých metodik a následnému zvolení jedné z nich, jež se bude aplikovat na vývoj softwarové aplikace. Vlastní práce se zaměřuje na vývoj vizualizačního softwaru pro společnost zabývající se skladováním ovoce a zeleniny v České republice i zahraničí. Hlavním účelem vyvíjeného softwaru je, aby farmář, zemědělské družstvo, distribuční společnost, byla schopna prokázat, za jakých podmínek byly jejich plodiny a produkty skladovány. Toto osvědčení je již dnes standartní podmínkou velkých obchodních řetězců při odkupu skladovaných zemědělských plodin. Všechna data jsou načtena z paměti programovatelného logického automatu (PLC - Programmable Logic Controller), který řídí celkový proces vytváření vhodných podmínek atmosféry ve skladu. Tento software tedy vyžaduje adekvátně naprogramované PLC, aby bylo možné vybraná data propojit. Z tohoto důvodu je do vývoje SW nutné zapojit i samotné vývojáře aplikačního softwaru daného PLC. Dále požadavky vlastníka produktu vycházejí ze zkušeností z předešlé aplikace, z toho důvodu Product owner vycházel při psaní backlogu z předešlé aplikace, kterou má Rainbow Galaxy nahradit. Hlavním požadavkem při vývoji bylo zachovat vizualizační prostředí původní aplikace z důvodu snadnějšího přechodu stávajících koncových zákazníků.

Firma tvořící tento software nevyužívá agilních metodik k vývoji SW, avšak pro důvody této bakalářské práce se rozhodli jí vyzkoušet. Hlavním důvodem tohoto počínání bylo zjistit, zda aplikace agilní metodiky přinese firmě větší efektivitu při tvorbě SW.

4.1 Porovnání metod a následný výběr

4.1.1 Porovnání tradičních a agilních metod

Moderní životní cyklus vývoje softwaru metodiky lze rozdělit do dvou typů – Tradiční proces a agilní proces. Tyto procesy budou v krátkém popisu rozebrány níže. Přehledné porovnání tradičních a agilních metod dle jednotlivých kritérií je zobrazeno v Tabulce 1 na následující stránce.

	Rigorózní metodiky	Agilní metodiky
Nástroje metodiky	Procesy se zaměřují na formálnost a dokumento-vatelnost, lidé jsou sekun-dární faktor	Praktiky vychází ze znalostí jednotlivců, lidé jsou klíčovým faktorem úspěchu
Podrobnost metodiky	Podrobný popis činností a procesů	Pouze základní struktura
Kvalita	Zaměření na kvalitu procesů, které povedou ke kvalitnímu výsledku	Zaměření na priority zákazníka
Předvídatelnost	Sběr požadavků a plánování předem	Přírůstkové shromažďování požadavků, plánování po iteracích
Změny	Snaha změny minimalizovat a řízení změn	Snaha změny umožnit s ohledem na nové události
Participace zákazníka na projektu	Jen v počátcích a koncových fázích	Po celou dobu projektu
Specializace lidí	Vyžaduje specializaci pro týmové role	Sdílení znalostí a spolupráce v týmu
Dokumentace	Rozsáhlá dokumentace	Minimální dokumentace, podstatné je pochopení
Způsob vývoje	Vodopádový, iterativní s dlouhými iteracemi	Přírůstkový vývoj s velmi krátkými iteracemi
Forma komunikace	Převážně písemná	Ostatní

Tabulka 1: Porovnání rigorózních a agilních metodik [11]

Tradiční metodiky, které lze označovat i jako rigorózní, jsou charakterizovány sekvenční sérií kroků, jako jsou požadavky, definice, plánování, budování, testování a nasazení. Za prvé jsou v plném rozsahu zdokumentovány požadavky klienta. Následuje vizualizace, pomocí které se vytvoří obecná architektura softwaru. Poté začíná skutečné kódování, na něž dále navazují různé typy testování a výsledné konečné nasazení.

Základní myšlenkou tradičních metodik je detailní popis celého projektu, ještě před tím, než samotný projekt začne. Práce samotná je pak plánována na základě detailního popisu z úvodu projektu napříč celou vývojovou fází. Lidský faktor je zde brán jako snáze nahraditelný a se zákazníkem se dále v průběhu vývoje nepočítá. Se zákazníkem je vytvořen pouze smluvní vztah. Tento způsob vývoje SW je velmi neflexibilní a neodpovídá dnešním vývojovým trendům, kdy je kladen důraz na flexibilitu řešení a možnost zapracování změn v průběhu vývoje samotného.

Agilní metodiky berou v potaz především důraz na přizpůsobitelnost nových řešení a začlenění těchto možností do projektu v průběhu jeho vytváření. Jak již bylo zmíněno v kapitole 3.3.2., u agilní metodiky je nejdůležitější pevně definovaný čas, jehož délka se určuje na začátku vývoje. Kdežto pro rigorózní modely je prioritní funkcionalita. Tudíž agilní metody z časového hlediska redukují funkcionalitu jen na nejnutnější funkce a implementují je do zdrojového kódu. Dále tyto metodiky předpokládají, že dojde při vývoji ke změnám funkcionality, proto je kladen důraz na přizpůsobivost. Pro kontrolu změněných a upravených částí zdrojového kódu dochází k častému testování, aby se zajistila a zachovala kompatibilita softwaru. Dalším hlavním bodem těchto metod je neustálá komunikace se zákazníkem. Principy agilních metod vychází z Manifestu agilního vývoje softwaru (3.3.1.)

Přehledné porovnání tradičních a agilních metod dle jednotlivých kritérií je zobrazeno v tabulce 1.

4.1.2 Porovnání SCRUM a Lean Development

Scrum je vývojový softwarový rámec se zaměřením na schopné jedince, kdežto Lean Development pomáhá optimalizovat vývojový proces odstraňováním všeho, co zásadně neomezuje chod softwaru. Obě metodiky slibují zefektivnění času a snížení spotřeby peněz. U Scrumu však lze doba vývoje časově vyjádřit. Lean Development spíše jen udává, jak za pomoci deseti základních pravidel této metodiky dospět k dané časové a peněžní úspoře. Zatímco Scrum a Lean se soustřeďují na různé aspekty vývoje, obě metody se primárně zaměřují na vytváření určitých bodů s principem základních hodnot, které sdílejí. Lean a Scrum se však mohou navzájem vcelku dobře doplňovat. V případech kde Scrum nepředepisuje konkrétní procesy, může být rozšířen o proces, či principy z Lean Development.

4.1.3 Porovnání SCRUM a Extrémní programování

Scrum je metodika, která se zabývá spíše produktivitou, zatímco extrémní programování je zaměřené spíše na samotný proces vývoje. Nespornou výhodou XP programování, které je v mnoha společnostech využíváno jsou inženýrské praktiky, například párové programování, testově řízený vývoj. Tyto činnosti zrychlují proces vývoje a omezují potřebu znovu procházení kódu. XP týmy pracují na požadavcích zákazníka v přesně definovaných prioritách v daném pořadí. Kdežto Scrum tým nemusí

dodržet plnění požadavků v daném sledu, ale může je v rámci jednoho sprintu plnit v libovolné posloupnosti. Z mého pohledu je lepší začít se Scrum metodikou, která je více formálně ukotvená, kdežto XP vyžaduje užší vztah týmu, aby byl schopný uvést věci do extrému, které jsou základním kamenem extrémního programování. Společnou záležitostí těchto metodik je snaha o dodržování pevně stanovené doby (40 hodin týdně), aby nedocházelo k vyčerpání lidského organismu a tím neklesala rychlost průběhu vývoje daného softwaru. XP metodologie je více benevolentní ke změnám konkrétních vlastností během vývojové iterace.

4.1.4 Výběr metodiky

Z výčtu a porovnání metodik byla vybrána metodika Scrum. Především pro její nesporné výhody, které umožňují sledování aktuálního pracovního postupu v projektu a dodávání mezi-iteračních výsledků zákazníkovi. Dále tím, že je přípustné během sprintu vytvářet software v jakémkoli sledu definovaných požadavků běžícího sprintu. Scrum master po konzultaci s vlastníkem produktu umožňuje přidání dalších User stories z backlogu do sprintu v případě, že již byly vykonány všechny úkony ze sprint backlogu. Či v opačném případě může Scrum Master po konzultaci se zákazníkem pár požadavků slevit, aby byl dodán funkční SW z dané iterace sprintu včas. Jedním z hlavních důvodů výběru této metodiky byl zpětnovazebně řízený vývoj Scrumu, který je založený na zpětné vazbě vlastníka produktu (Product owner) po každé iteraci vývojového cyklu (sprintu).

4.2 Návrh projektu

Vývoj softwarové aplikace s pracovním označením Rainbow Galaxy byl v době psaní bakalářské práce v počáteční 1.etapě projektu. První fáze je zaměřena na zobrazení měřených a řídicích dat v čase pro distributorské i diagnostické účely.

První etapu projektu lze logickou posloupností rozdělit na část organizační v samotném úvodu projektu, dále vývojovou za použití vybrané metodiky Scrum, zakončenou dodáním konečného produktu na trh v požadovaném čase.

Ve druhé etapě projektu bude SW rozšířen o možnost samotného nastavení celkového řídicího procesu skladování potravin pro různé druhy ovoce a zeleniny s pestrou škálou konfigurací parametrů a skladovacích programů.

4.2.1 Organizace projektu

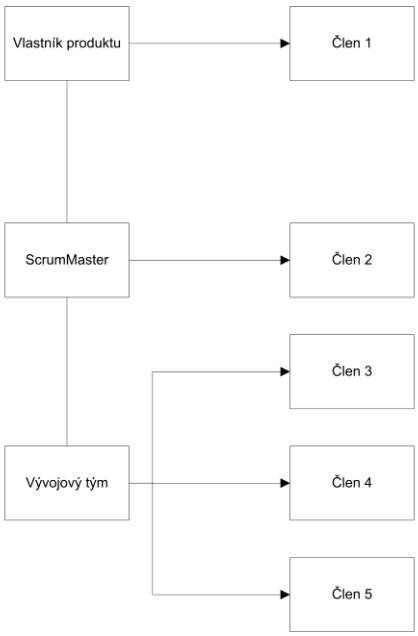
Pro potřeby psaní této bakalářské práce, souhlasila realizační společnost vyvíjeného SW, s implementováním metodiky Scrum do vývojového procesu a zapojení autora této práce do vývojového týmu.

Dle dané metodiky je naprosto klíčové, aby roli Scrum Master zastával pouze jeden člen, který však není zároveň členem vývojového týmu. To umožňuje Scrum Masterovi být nezávislým prostředníkem mezi vývojovým týmem a vlastníkem produktu. Scrum Master zajišťuje, aby vývojový tým měl vše potřebné k plnění jejich úkolů, zajišťuje pohodu na pracovišti a tvůrčí prostředí. Naopak také naslouchá vlastníkovi produktu při vznášení požadavků na vyvíjení daného SW. Scrum Master pak vede celý proces vývoje a udržuje v chodu dodržování pravidel iteračních cyklů jednotlivých sprintů a chování jednotlivých účastníku Scrum týmu v souladu s danými Scrum pravidly.

User stories potřebné pro zadání na straně aplikace PLC zabývající se samotným programováním řízení daných programů skladování potravin a jejich požadavky jsou majetkem zadávající společnosti a po vzájemné dohodě nebudou zveřejněny.

4.2.2 Rizika

Celkem bylo do Scrum týmu zahrnuto 5 členů - 1 vlastník produktu, 1 Scrum Master a 3 členové vývojového týmu (viz. Obrázek číslo 7).



Obrázek 7: Scrum tým

Při plánování bylo zapotřebí zvážit několik rizik, které by mohly narušit průběh vývoje samotné aplikace:

- vyvíjená aplikace RainbowGalaxy je nadstavba řídicího systému Climatix. Z toho důvodu je zapotřebí vyvíjet i řídicí aplikaci samotného PLC dle standardu IEC 61131-3, aby bylo možné dané požadavky uspokojit a obě technologie úspěšně propojit. Toto riziko bylo ohodnoceno v důsledku složení členů týmu jako malé.

- externí zařízení přistuje k řídicímu systému Climatix skrze komunikační rozhraní Rainbow přes Ethernet (případně USB či procesní sběrnici). Přístup k archivním datům není zcela zdokumentován a žádná ze zúčastněných stran nemá s tímto rozhraním předchozí zkušenosti. Toto riziko bylo ohodnoceno jako vysoké.

- z důvodu malé velikosti vývojového týmu ve velmi široké technické oblasti, může jakákoliv nepřítomnost jediného člena, způsobit ochromení a zablokování zbytku týmu (např. viz popis rizika prvního bodu - Rainbow Galaxy je nadstavba řídicího systému, za účelem naprogramování vyšší vrstvy musí být odpovídající data dostupná i v nižší vrstvě). Toto riziko bylo ohodnoceno jako velmi vysoké. Z důvodu snížení tohoto rizika byla do plánu projektu vložena časová rezerva trvající jeden iterační cyklus v posledním plánovaném sprintu.

4.2.3 Komunikace členů

Komunikace členů probíhá na denní bázi na základě Scrum pravidel (výjimečně pomocí vzdálené komunikace komunikačními nástroji), při kterých se řeší plánování jednotlivých úloh, zhodnocuje se jejich postup a předvádí se již dosažené dílčí výsledky. Na konci samotných sprintů se pak zpětně hodnotí průběh daných iteračních cyklů.

4.2.4 Plán projektu

Z důvodu obtížnějších možností testování celkového SW na straně PLC bylo nutné stanovit iteraci delší. Plán projektu je vyobrazen v Tabulce číslo 2.

I. etapa vývoje Rainbow Galaxy	Začátek	Konec
Nastavení spolupráce zúčastněných stran projektu	1.11.2016	28.11.2016
Organizace projektu - zpracování požadavků na software v podobě Uživatelských příběhů a vytvoření seznamu požadavků dle priorit	5.12.2016	6.1.2017
Sprint 0 - návrh architektury a řešení	9.1.2017	20.1.2017
Sprint 1	23.1.2017	10.2.2017
Sprint 2	13.2.2017	3.3.2017
Sprint 3	6.3.2017	24.3.2017
Sprint 4	27.3.2017	14.4.2017
Sprint 5	17.4.2017	5.5.2017
Časová rezerva	8.5.2017	26.5.2017
Field test, zaškolení zákazníka	29.5.2017	30.6.2017
Pilotní provoz fáze projektu 1	1.7.2017	31.12.2017

Tabulka 2: Plán projektu

4.2.5 Požadavky produktu Rainbow Galaxy (Produkt backlog)

Vlastník produktu realizoval požadavky na vytvářený software v 1. etapě vývoje SW v podobě uživatelských příběhů v tabulce číslo 3 níže. Tyto požadavky zahrnují pouze informace, které jsou vyvíjené na straně SW Rainbow Galaxy a bylo je možné zveřejnit. Položky jsou seřazené od nejvyšší priority č.1 po nejnižší. Tento seznam nemusí být konečný a může se mezi jednotlivými sprinty postupně vyvíjet a doplňovat o nové požadavky vlastníka produktu s přeskupením definovaných priorit. Pokud je však již sprint naplánován a probíhá, není dovoleno vlastníkovi produktu změnit položky sprintu vývojového týmu při jeho běhu.

	Rainbow Galaxy fáze 1 - Grafy
Priorita	Product backlog
1	Osa x bude mít zobrazena čas ve formátu den. měsíc. rok s popiskem Datum a čas
2	Při zvětšování /zmenšování okna samotné aplikace se velikost grafů mění proporcionálně
3	S osou x je možné hýbat a posouvat jí směrem v čase do prava, či do leva s myší podržením pravého tlačítko myši na dané ose tak, aby se všechny měřené charakteristiky posouvaly souběžně
4	Vybraná část na ose x lze zoomovat pro přiblížení otočným kolečkem myši
5	Na vybraném grafu je po kliknutí levým tlačítkem myši na aktuální

	charakteristiku možné zobrazovat aktuální data v ose x a ose y
6	Uživatel může zvětšit zobrazenou část grafu na maximum podle vybraného výřezu
7	Uživatel se může ze zvětšeného výřezu grafu vrátit na původní velikost
8	Osa y měřených teplotních vstupních veličin bude popsána popiskem Teplota a jednotkou [°C]
9	Osa y měřených vlhkostí bude popsána popiskem Relativní vlhkost a jednotkou [%]
10	Graf zobrazující charakteristiku teploty čidla venkovního vzduchu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla venkovního vzduchu
11	Graf zobrazující charakteristiku nastavitelné žádané materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla venkovního vzduchu
12	Graf umožňuje zobrazit charakteristiku minimální materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené minimální materiálové teploty všech aktivních teplotních čidel
13	Graf umožňuje zobrazit charakteristiku průměrné materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené průměrné materiálové teploty všech aktivních čidel
14	Graf umožňuje zobrazit charakteristiku maximální materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené maximální materiálové teploty všech aktivních čidel
15	Graf umožňuje zobrazit charakteristiku venkovní relativní vlhkosti venkovního vzduchu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu venkovní relativní vlhkosti
16	Graf umožňuje zobrazit charakteristiku průměrných teplot čidla č.1 a č.2 kanálových teplot. Aplikační SW musí mít zaregistrovanou archivaci datového bodu průměrné hodnoty teploty čidla č. 1 a č.2 kanálové teploty
17	Graf musí zobrazovat charakteristiku řízení výstupu klapky servopohonu v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení výstupu klapky servopohonu
18	Graf musí zobrazovat charakteristiku řízení výstupu ventilátoru v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení ventilátoru
19	Osa y řízení servopohonu klapky bude mít namísto % jednotek z maximálního otevření zobrazena obrázek klapky
20	Osa y řízení ventilátoru bude mít namísto % jednotek zobrazen z maximálních otáček ventilátoru obrázek ventilátoru
21	Osa y stavového alarmového hlášení bude mít namísto popisu osy y zobrazen obrázek alarmu
22	Osa y řízení chladícího agregátu bude mít namísto % jednotek výkonu z maximálního výkonu chlazení zobrazen obrázek sněhové vločky
23	Osa y řízení odtávání chladícího agregátu bude mít namísto informaci o zapnutí obrázek odtávání
24	Osa y zvlhčovače bude mít namísto popisu vypnuto / zapnuto zobrazen obrázek kapiček zvlhčovače
25	Osa y řízení reverzní ventilace bude mít namísto % otáček ventilátoru zobrazen

	obrázek ventilátoru
26	SW bude mít záložku pro konfiguraci připojení k dané IP adrese řídicího systému
27	V záložce konfigurace připojení bude tlačítko načtení archivních dat z PLC do SQLite databáze
28	Graf musí zobrazovat charakteristiku průběhu alarmových hlášení ze skladu v čase na společném zobrazení. Aplikační SW musí mít zaregistrovanou archivaci datového bodu alarmových hlášení. Aplikační SW musí umět zpracovat jednotlivé alarmy a uložit je do datového bodu tak, aby byla možná jeho následná archivace v rámci PLC v čase
29	Graf musí zobrazovat charakteristiku řízení chladícího agregátu v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení chladícího agregátu v čase, pokud je v konfiguraci skladu
30	Graf musí zobrazovat charakteristiku řízení odtávání výparníku chladicího agregátu v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení odtávání výparníku chladícího agregátu v čase, pokud je v konfiguraci skladu
31	Graf zobrazující charakteristiku řízení agregátu zvlhčovače vzduchu ve skladu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu řízení zvlhčení vzduchu v čase
32	Graf zobrazující charakteristiku řízení agregátu reverzního ventilátoru ve skladu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu řízení reverzního ventilátoru v čase
33	Graf může být zobrazen v časech od - do, dle daného výběru
34	Uložená data z PLC v PC je možné poslat elektronickou poštou
35	Data pro zobrazení grafů bude možné volit dle jednotlivých sekcí skladů
36	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.1
37	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.2
38	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.3
39	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.4
40	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.5
41	Vstupní veličiny mohou být do grafu vstupních veličin definovány na základě výběru dostupných měřených veličin pro pozici č.6
42	Pro vybranou vstupní veličinu na pozici č.1 může být vybrána barva dané charakteristiky
43	Pro vybranou vstupní veličinu na pozici č.2 může být vybrána barva dané charakteristiky
44	Pro vybranou vstupní veličinu na pozici č.3 může být vybrána barva dané charakteristiky
45	Pro vybranou vstupní veličinu na pozici č.4 může být vybrána barva dané charakteristiky
46	Pro vybranou vstupní veličinu na pozici č.5 může být vybrána barva dané charakteristiky

47	Pro vybranou vstupní veličinu na pozici č.6 může být vybrána barva dané charakteristiky
48	Graf zobrazující charakteristiku žádané teploty materiálu ve skladu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu žádané materiálové teploty
49	Graf zobrazující charakteristiku teploty čidla č.1 materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla č. 1 materiálové teploty
50	Graf zobrazující charakteristiku teploty čidla č.2 materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla č. 2 materiálové teploty
51	Graf zobrazující charakteristiku teploty čidla č.3 materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla č. 3 materiálové teploty
52	Graf zobrazující charakteristiku teploty čidla č.4 materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla č. 4 materiálové teploty
53	Graf zobrazující charakteristiku teploty čidla č.5 materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla č. 5 materiálové teploty
54	Grafy bude možné vytisknout
55	Grafy bude možné exportovat do PNG formátu
56	Grafy bude možné exportovat do CSV formátu
57	Grafy bude možné exportovat do XLS formátu
58	Grafy bude možné exportovat do PDF formátu
59	Bude zaveden online mód sledování měřených a řídicích dat
60	SW bude nastavitelný v jazykové mutaci do německého jazyka
61	SW bude nastavitelný v jazykové mutaci do anglického jazyka
62	SW bude nastavitelný v jazykové mutaci do ruského jazyka
63	SW bude nastavitelný v jazykové mutaci do polského jazyka
64	SW bude nastavitelný v jazykové mutaci do maďarského jazyka

Tabulka 3: Požadavky vlastníka projektu

4.3 Realizace softwaru

4.3.1 Nultý sprint

Ačkoliv není nultá iterace podmínkou Scrumu, obecně se doporučuje. Tým v této době prodělává případná školení, volí se potřební lidé do týmu a vybírají se vhodné nástroje. Z důvodu limitujících možností lidských zdrojů byl náš tým pevně definovaný již před touto fází a personální zásahy nebyly nutné.

Po prostudování dané tématiky se Scrum tým rozhodl použít vývojové prostředí Visual Studio s použitím jazyka C# .NET s grafickým zobrazením daného SW za pomocí technologie WPF (Windows Presentation Foundation). Na straně PLC pak bylo použito vývojové prostředí Sapro a Scope pro vývoj softwaru řídicího systému.

Pro načtení a uložení dat na straně PC padly návrhy pro použití databází SQLite a Microsoft SQL Server Compact Edition. Bylo však zjištěno, že společnost Microsoft označila svůj produkt Microsoft SQL Server Compact za zastaralý a již od února 2013 jej dále nevyvíjí. MS SQL Server Compact již bude pouze udržována v rámci standardního životního cyklu produktu. Z tohoto důvodu byla vybrána databáze SQLite, která se neustále vyvíjí a používají ji velké světové společnosti typu Google ve svém prohlížeči Chrome, apod (viz. podkapitola 4.3.1.2.).

Pro práci s SQLite databázemi se při jejich ladění a testování osvědčil program LINQPad5. Ten umožňuje např. prohlížení relačních databází různých typů a zobrazování jejich dat. Pro prohlížení SQLite databází je však potřeba doinstalovat SW plugin.

Ze srovnávaných možností volně dostupných SW knihoven pro zobrazení grafů vyšla nejlepší možnou variantou knihovna OxyPlot podporující WPF z důvodu velmi dobré optimalizace zobrazení dat. Ačkoliv dostupná dokumentace je ve velmi špatném stavu. Oproti tomu srovnávaná knihovna LiveChart má mnohem lepší design grafů, perfektní dokumentaci, avšak rychlost zobrazení dat pro větší počet vzorků je v základní verzi neuspokojivý (viz. podkapitola 4.3.1.3.).

4.3.1.1 PLC řady Climatix

PLC řady Climatix je volně programovatelný řídicí systém. Jedná se o OEM produkt společnosti Siemens a je určený výhradně výrobcům zařízení v oblasti HVAC (Heating, Ventilation, Air Conditioning). Z tohoto důvodu lze tento výrobek nalézt na trhu

pod různým označením. Běžný koncový uživatel si tento výrobek nemůže na trhu koupit, protože podléhá přísným obchodním regulacím.

Výrobce si aplikační řídicí SW daného PLC vyvíjí samostatně a implementuje do něj své „know how“, potřebné k úspěšnému řízení svých systémů. Z tohoto důvodu je aplikační SW každé firmy přísně střežené tajemství a vyvíjený SW je její duševní vlastnictví.

4.3.1.2 SQLite

SQLite je vcelku malá knihovna, která implementuje soběstačný, bezserverový, nulově konfigurovaný, transakční SQL databázi, kterou je možné zdarma užívat pro soukromé nebo komerční účely. Databáze nepracuje na principu klient-server (oddělení klienta a serveru přes počítačovou síť, přes kterou spolu komunikují). Databáze je uložena na pevném úložišti používaného zařízení v jednom jediném souboru (.dbm – DataBase Manager) – snadná záloha dat. Zde se data ukládají užitím primárního klíče do stejně rozsáhlých bloků. Užitím hašovacích technik v souboru databáze se při vyhledávání pomocí klíče rychle dosáhneme k datům.

Výhodou je, že formát souboru .dbm je multiplatformní - snadný přenos mezi jinými platformami. SQLite umožňuje, aby z jednoho zdroje mohlo číst více programů. V případě probíhajících zápisů je databáze uzavřena. Nelze do ní paralelně provádět další zápisy a též není možné v daný okamžik z databáze číst.

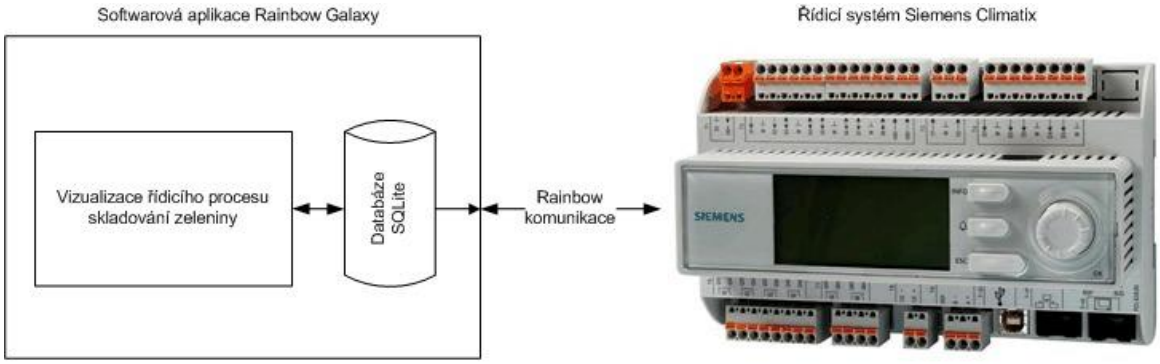
4.3.1.3 OxyPlot

OxyPlot je multiplatformní knihovna pro vykreslování grafů v prostředí .NET. Knihovna obsahuje rozsáhlé množství metod a tříd pro použití v jazyce C#. Umožňuje vizualizaci dat, nejen na platformě WPF (Windows Presentation Foundation), ale i Windows Form, Windows 8, Windows Phone, Windows Phone Silverlight, Windows Forms, Silverlight, GTK#, Xwt, Xamarin.iOS, Xamarin.Android, Xamarin.Forms a Xamarin.Mac. Kód je licencován pod MIT licencí. Což je velmi tolerantní a přátelská licence umožňující komerční a soukromé užívání, vytváření jakýkoliv změn, možnost distribuování zkompilovaného či zdrojového kódu. Avšak je potřeba při užití této knihovny odkázat na autorská práva.

4.3.1.4 Návrh architektury

Na obrázku (viz. Obrázek číslo 8) je zobrazeno základní schéma navržené architektury SW aplikace Rainbow Galaxy. Principiální postup je popsán v základních krocích.

1. Po propojení počítače a řídícího systému PLC Ethernet kabelem budou načtena historická a naměřená data z PLC, pomocí komunikačního protokolu Rainbow.
2. Pro uložení dat byla vybrána databáze SQLite.
3. Dále pak pomocí knihovny OxyPlot dochází k vývoji vizualizace řídicího procesu skladování zeleniny, ovoce a načtení dat z SQLite
4. Výsledné zobrazení SW pomocí WPF (Windows Presentation Foundation), které se zobrazí jako grafy.



Obrázek 8: Schéma architektury

4.4 První sprint

První sprint se zaměřuje na zobrazení vstupních měřených dat řídicího systému Climatix. Toto vyžaduje součinnost celého týmu v archivaci a přípravě měřených vstupních dat.

V počátku tohoto sprintu se sešel celý tým (vlastník produktu, Scrum Master a vývojový tým), aby byl přítomný na události zvané Plánování sprintu. Vlastník produktu všem přítomným popsal User stories z backlogu od nejvyšších priorit po nejnižší, (viz Tabulka číslo 3), kde priorita 1 má v číslování nejvyšší váhu a priorita 64 nejnižší. Vlastník produktu dále prezentoval původní SW aplikaci, jejíž vizuální prostředí má být šablonou pro nově vyvíjený SW Rainbow Galaxy.

Vývojový tým kladl na vlastníka produktu mnoho doplňujících otázek. To umožnilo celkové hlubší pochopení jmenovaných User stories v backlogu.

Během události Plánování sprintu se kolektivně vytvořil Scrum cíl pro danou první iteraci. Dohodnutým cílem bylo vytvořit základní uživatelské prostředí vizualizace grafů pro zobrazení dat (posun charakteristik v čase, zvětšení zobrazení charakteristik, detailnější zobrazení průběhu hodnot v čase, apod.) a zobrazení prvních několika vstupních měřených veličin.

Vývojový tým posléze definoval, které položky z backlogu, dle priority, je možné zavést do Sprintu 1 (viz Tabulka číslo 4).

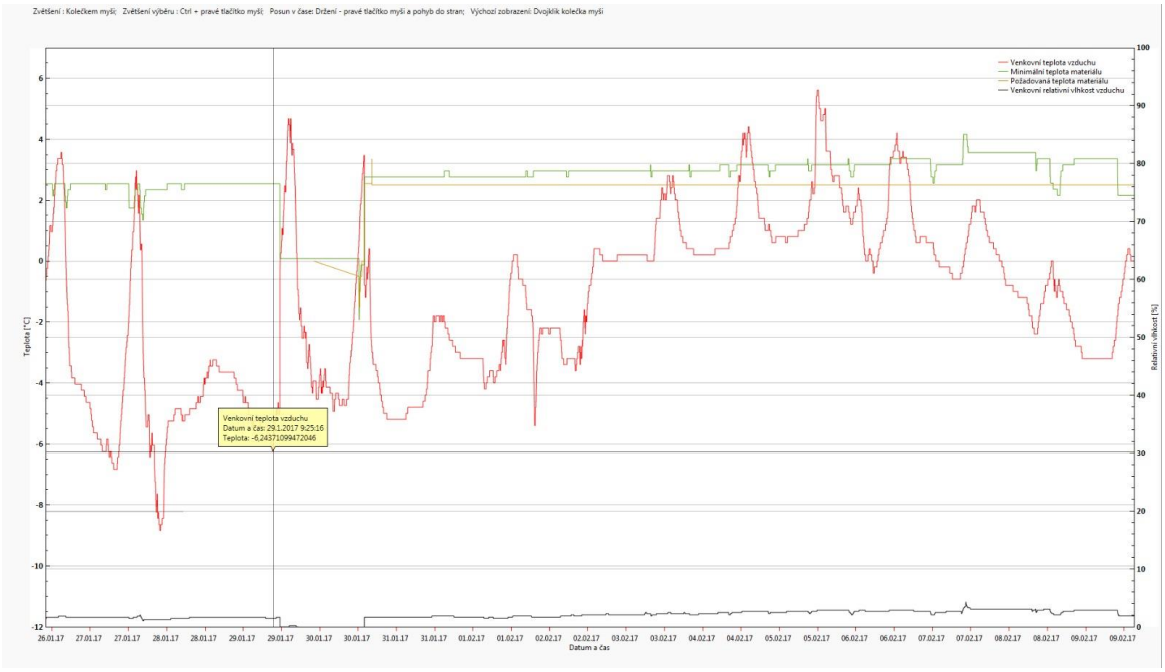
	Rainbow Galaxy fáze I. – Sprint 1
Priorita	Sprint backlog
1	Osa x bude mít zobrazena čas ve formátu den. měsíc. rok popiskem Datum a čas
2	Při zvětšování /zmenšování okna samotné aplikace se velikost grafů mění proporcionálně
3	S osou x je možné hýbat a posouvat jí směrem v čase do prava, či do leva s myší podržením pravého tlačítko myši na dané ose tak, aby se všechny měřené charakteristiky posouvaly souběžně
4	Vybraná část na ose x lze zoomovat pro přiblížení otočným kolečkem myši
5	Na vybraném grafu je po kliknutí levým tlačítkem myši na aktuální charakteristiku možné zobrazovat aktuální data v ose x a ose y
6	Uživatel může zvětšit zobrazenou část grafu na maximum podle vybraného výřezu
7	Uživatel se může ze zvětšeného výřezu grafu vrátit na původní velikost
8	Osa y měřených teplotních vstupních veličin bude popsána popiskem Teplota a jednotkou [°C]
9	Osa y měřených vlhkostí bude popsána popiskem Relativní vlhkost a jednotkou [%]
10	Graf zobrazující charakteristiku teploty čidla venkovního vzduchu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla venkovního vzduchu
11	Graf zobrazující charakteristiku nastavitelné žádané materiálové teploty. Aplikační SW musí mít zaregistrovanou archivaci datového bodu teploty čidla venkovního vzduchu

Tabulka 4: Sprint 1

Během sprintu měl tým denně (krátkou) tzv. Denní Scrum schůzku. Zde se řešilo, co každý jednotlivý člen vývojového týmu dělal v předešlém dni, co bude dělat v den této schůzky a zda nečelí nějakým překážkám při plnění úlohy. Pokud takováto překážka

nastala, odpovědné osoby problém řešily až po skončení schůzky (dle Scrum metodiky), aby nezdržovali zbytek týmu.

Po skončení této iterace proběhla událost nazvaná Posouzení Sprintu 1, na které byl přítomen vlastník produktu, Scrum Master a vývojový tým. Na této události byly prezentovány dosažené výsledky a předvedeno první demo z této iterace. Prezentace samotná byla rozdělena na dvě části. První byla struktura vytvořené SQLite databáze (viz. Příloha 7.2). Druhá samotná vizualizační část aplikace zobrazující archivní měřená data (viz Obrázek číslo 9). Dále bylo posouzeno splnění definovaného cíle této iterace. Až na drobné výhrady byl vlastník produktu spokojen a kolektivně jsme zhodnotili cíl prvního sprintu za vykonaný.



Obrázek 9: Vizuální zobrazení archivních dat - Sprint 1

Následovala událost Retrospektivní zhodnocení sprintu. Úkolem této činnosti je zefektivnění vývojového procesu dle podnětů členů týmu na základě zkušeností z první iterace. Například během realizace docházelo ke zbytečným prostojům z důvodu špatně připraveného rozhraní datových bodů. Když tyto body nejsou správně nadefinovány v aplikačním SW PLC, nelze s nimi pracovat ani v aplikaci Rainbow Galaxy. Navrhovaným řešením pro zlepšení této situace do dalších sprintů bylo domluvení si plánování potřebných datových bodů během denních Scrum schůzek.

4.5 Druhý sprint

Druhý sprint se zaměřuje na zobrazení akčních (výstupních řídicích) veličin. Tato iterace vyžaduje vyšší součinnost týmu než předchozí sprint z důvodu implementace robustních algoritmů pro regulaci daného systému na straně SW PLC.

V tomto sprintu se postupovalo analogicky jako při Sprintu 1. V počátku tohoto sprintu se sešel opět celý tým (vlastník produktu, Scrum Master a vývojový tým) na událost Plánování sprintu. Vlastník produktu popsal znovu User stories z backlogu se současně nejvyšší prioritou. V tomto případě nevyužil svého práva a seznam backlogu ponechal v předchozím stavu bez přesunu priorit či doplnění nové User stories.

Během události Plánování sprintu se kolektivně vytvořil nový Scrum cíl pro druhou iteraci. Dohodnutým cílem bylo především naprogramování řídicího SW aplikace PLC a vizualizace charakteristik řízení jednotlivých akčních členů skladu na straně Rainbow Galaxy. Osy charakteristik řízených agregátů budou doplněné o znázornění jednotlivých ilustračních obrázků (např. ikona vločky pro aktivní chlazení skladu, ikona klapky pro otevírání tunelu přívodního vzduchu apod.).

Vývojový tým posléze definoval, které položky z backlogu, dle priority, je možné zavést do Sprintu 2. V tabulce (viz. Tabulka číslo 5) jsou uvedené pouze User stories, potřebné pro část Rainbow Galaxy, které bylo možné zveřejnit.

Následovaly opět denní Scrum schůzky vedené Scrum Masterem až do završení vývojové fáze sprintu.

	Rainbow Galaxy fáze I. – Sprint 2
Priorita	Sprint backlog
12	Graf umožňuje zobrazit charakteristiku minimální materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené minimální materiálové teploty všech aktivních teplotních čidel
13	Graf umožňuje zobrazit charakteristiku průměrné materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené průměrné materiálové teploty všech aktivních čidel
14	Graf umožňuje zobrazit charakteristiku maximální materiálové teploty všech aktivních materiálových čidel. Aplikační SW musí mít zaregistrovanou archivaci datového bodu ze spočtené maximální materiálové teploty všech aktivních čidel
15	Graf umožňuje zobrazit charakteristiku venkovní relativní vlhkosti venkovního vzduchu. Aplikační SW musí mít zaregistrovanou archivaci datového bodu venkovní relativní vlhkosti
16	Graf umožňuje zobrazit charakteristiku průměrných teplot čidla č.1 a č.2 kanálových teplot. Aplikační SW musí mít zaregistrovanou archivaci datového bodu průměrné hodnoty teploty čidla č. 1 a č.2 kanálové teploty

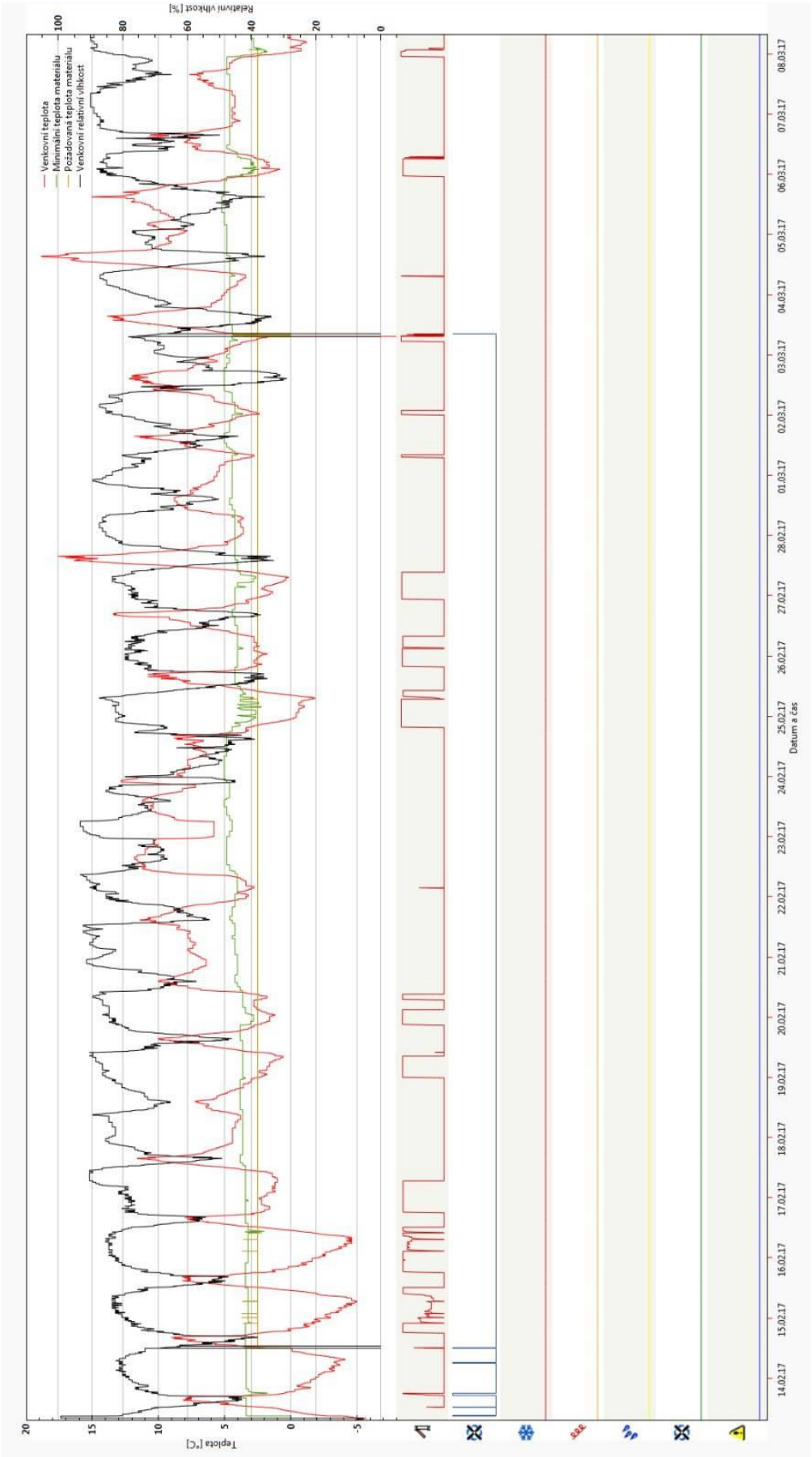
17	Graf musí zobrazovat charakteristiku řízení výstupu klapky servopohonu v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení výstupu klapky servopohonu
18	Graf musí zobrazovat charakteristiku řízení výstupu ventilátoru v čase na společném zobrazení. Aplikační SW PLC musí mít zaregistrovanou archivaci datového bodu řízení ventilátoru
19	Osa y řízení servopohonu klapky bude mít namísto % jednotek z maximálního otevření zobrazena obrázek klapky
20	Osa y řízení ventilátoru bude mít namísto % jednotek zobrazen z maximálních otáček ventilátoru obrázek ventilátoru
21	Osa y stavového alarmového hlášení bude mít namísto popisu osy y zobrazen obrázek alarmu
22	Osa y řízení chladícího agregátu bude mít namísto % jednotek výkonu z maximálního výkonu chlazení zobrazen obrázek sněhové vločky
23	Osa y řízení odtávání chladícího agregátu bude mít namísto informaci o zapnutí obrázek odtávání
24	Osa y zvlhčovače bude mít namísto popisu vypnuto / zapnuto zobrazen obrázek kapiček zvlhčovače
25	Osa y řízení reverzní ventilace bude mít namísto % otáček ventilátoru zobrazen obrázek ventilátoru

Tabulka 5: Sprint 2

Po skončení této iterace opět proběhla událost nazvaná Posouzení Sprintu 2, na které byl přítomen vlastník produktu, Scrum Master a vývojový tým. Při prezentaci iteračního dema byly prezentovány řídicí algoritmy na straně PLC pro ovládání servopohonu klapky a ventilátoru. Na straně Rainbow Galaxy byla předvedena samotná vizualizace archivních dat řízení těchto agregátů (viz. Obrázek číslo 10).

Dále bylo posouzeno splnění definovaného cíle této iterace. Vlastník produktu hodnotil pozitivně plynulé ovládání aplikace Rainbow Galaxy. Na straně SW aplikace PLC pak navrhl několik úprav řídicího SW. Navzdory těmto připomínkám, byl kolektivně cíl sprintu označen za splněný.

Následovala opět událost Retrospektivní zhodnocení sprintu. Jedním z hlavních problémů sprintu byla definována nevhodná organizace výměny projektů, mezi jednotlivými členy týmu. V několika případech se pak stávalo, že člen týmu pracující na straně Rainbow Galaxy pracoval se zastaralou verzí SW na straně PLC. Tím docházelo k nutnosti úprav na straně Rainbow Galaxy, po nahrání aktuální verze projektu do PLC. Jako řešení navrhl Scrum Master zavedení verzovacího systému od příští iterace pro všechny členy vývojového týmu.



Obrázek 10: Vizuální zobrazení archivních dat - Sprint 2

5 Závěr

Práce se zaměřovala na vývoj softwaru užitím agilní metodiky. Z tohoto důvodu bylo zapotřebí obohatit stávající vědomosti o rešerši provedenou na toto téma. Byly zde popsány tradiční a agilní metodiky. Po porovnání těchto metodik byla k realizování praktické části vybrána metodika Scrum, jejíž postupy pomohly vytvořit plán průběhu realizace projektu pro vyvíjenou aplikaci Rainbow Galaxy.

Vizualizační software Rainbow Galaxy je vyvíjen pro společnost, která se zabývá skladováním ovoce, zeleniny (převážně brambory, česnek, cibule, zelí), stromků v České republice i zahraničí. Hlavním účelem vyvíjeného softwaru je, aby farmář, zemědělské družstvo, distribuční společnost, byla schopna prokázat, za jakých podmínek byly jejich plodiny a produkty skladovány. Toto osvědčení je již dnes standartní podmínkou velkých obchodních řetězců při odkupu skladovaných zemědělských plodin.

Vlastník produktu v úvodu projektu definoval backlog (v podobě User stories) na grafickou vizualizaci vyvíjené aplikace na základě vzhledu aktuálně používané SW aplikace, která bude produktem Rainbow Galaxy nahrazena, z důvodu nově použitých řídicích technologií.

Vývoj aplikace s pracovním názvem Rainbow Galaxy proběhl v rámci této bakalářské práce v prvních třech sprintech (nultém, prvním a druhém) celkového plánu.

Cílem nultého sprintu bylo sestavení pracovní skupiny (vývojový tým), výběr a seznámení se s vhodnými nástroji pro tvorbu SW. Dále byl vytvořen návrh architektury, který se skládal ze dvou částí (PLC a Rainbow Galaxy).

Po dohodě s vlastníkem projektu, Scrum masterem a vývojovým týmem bylo cílem první iterace vytvořit základní uživatelské prostředí vizualizace grafů pro zobrazení dat (posun charakteristik v čase, zvětšení zobrazení charakteristik, detailnější zobrazení průběhu hodnot v čase apod.) a zobrazení prvních několika vstupních měřených veličin. V každodenních krátkých Scrum schůzkách se kontroloval postup projektu a organizace týmových činností. Po ukončení prvního sprintu následovala Scrum událost Posouzení sprintu, při které se prezentovaly dosažené výsledky vlastníkovi projektu. Ten k vyvíjené aplikaci měl pouze drobné připomínky, jinak byl spokojen a kolektivně byl cíl prvního sprintu zhodnocen jako vykonaný (zobrazení vizualizace prostředí viz. Obrázek 9; Příloha 7.1).

Dohodnutým cílem pro druhou iteraci bylo především naprogramování řídicího SW aplikace PLC a vizualizace charakteristik řízení jednotlivých akčních členů skladu na straně Rainbow Galaxy. Osy charakteristik řízených agregátů byly doplněné o znázornění jednotlivých ilustračních obrázků (např. ikona vločky pro aktivní chlazení skladu, ikona klapky pro otevírání tunelu přívodního vzduchu apod.). Při prezentaci iteračního dema vlastníkovi produktu byly prezentovány řídicí algoritmy na straně PLC pro ovládání servopohonu klapky a ventilátoru. Na straně Rainbow Galaxy byla předvedena samotná vizualizace archivních dat řízení těchto agregátů (viz. Obrázek číslo 10 a Příloha 7.1). Po prezentaci druhé etapy vlastník produktu hodnotil výsledky druhého sprintu pozitivně, ocenil také plynulé ovládání aplikace Rainbow Galaxy a podobný grafický vzhled v porovnání s předchozí aplikací, užitý na současné verzi. Na straně SW aplikace PLC pak navrhl několik drobných úprav řídicího SW PLC. Cíl sprintu byl označen za splněný a zmíněné připomínky byly zapracovány do další iterace.

Užití praktik metodiky Scrum pro vývoj aplikace byl velice zajímavý, avšak velmi časově náročný projekt, především z důvodu denních schůzek (či komunikace) vývojového týmu. Realizační firma, která souhlasila s použitím vývojové metodiky Scrum pro účely této bakalářské práce, hodnotí použití metodiky pozitivně a rozhodla se ji zavézt i na jejím novém projektu. Jako zákazníka, kupujícího ovoce a zeleninu, mě toto téma velice obohatilo. Získal jsem zajímavý pohled do zákulisí procesu skladování a pochopil jsem, jak dlouhá je cesta mezi tím, než se zelenina vypěstuje, uskladní a dostane na pulty obchodů.

6 Seznam použitých zdrojů

[1] **SNELL, Mike**. Microsoft Visual Studio 2015 unleashed. 3rd editon. Indianapolis, IN: Sams, 2015. ISBN 978-067-2337-369.

[2] **KADLEC, Václav**. Agilní programování: metodiky efektivního vývoje softwaru. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

[3] **ROBINSON, Simon**. C#: programujeme profesionálně. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.

[4] **EELES, Peter a Peter CRIPPS**. Architektura softwaru. Brno: Computer Press, 2011. ISBN 978-80-251-3036-0.

[5] **ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ**. PLC a automatizace. Praha: BEN – technická literatura, 1999. ISBN 80-860-5658-9.

[6] **Beck, Kent**. Extrémní programování. [překl.] Pavel Makovec. Praha : Gradapublishing, 2002. str. 160. ISBN 80-247-0300-9.

[7] **Agilní metodiky řízení vývoje software (Agile software development methodologies)** [online]. c2011-2016 [cit. 2016-11-08]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

[8] **BUCHALCEVOVÁ, Alena**. Metodiky vývoje a údržby informačních systémů. 1. vyd. Praha : Grada, 2005. 163 s. Management v informační společnosti. ISBN 80-247-1075-7.

[9] **Marry, Poppendick**. Lean Software Development. místo neznámé : Addison Wesley, 2003. ISBN 0-321-15078-3.

[10] **PEJCHAL, Jakub**. *Agilní a tradiční metodiky v projektovém řízení*. Brno, 2015.

[11] **Manifest Agilního vývoje software** [online]. c2001 [cit. 2016-12-20]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

[12] **Návrh vývojové metodiky pro webové aplikace** [online]. c2017 [cit. 2017-02-10]. Dostupné z: http://docplayer.cz/3551036-Navrh-vyvojove-metodiky-pro-webove-aplikace.html

[13] **Spirálový model** [online]. [cit. 2017-02-05]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/

[14] **Metodika řízení projektů podle A ...** [online]. c2017 [cit. 2017-01-04]. Dostupné z: http://slideplayer.cz/slide/2283123/

[15] **Programujte agilně, …** [online]. c2017 [cit. 2017-01-08]. Dostupné z: http://www.zive.cz/clanky/programujte-agilne-nic-jineho-vam-nezbyva-a-nebo-ano/sc-3-a-111219/default.aspx

7 Přílohy

7.1 Přiložené CD se softwarovou aplikací

7.2 Navržená SQLite databáze