Univerzita Pardubice

Fakulta ekonomicko-správní

Využití agilních metod, SCRUM, v projektovém řízení

Bc. Květoslava Bartůňková

Diplomová práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2010/2011**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Květoslava BARTŮŇKOVÁ**
Osobní číslo:	**E09824**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Využití agilních metod, SCRUM, v projektovém řízení.**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

V práci budou zpracována následující problematika: 1) Tradiční metodiky tvorby IS 2) Agilní metodiky tvorby IS 3) Popis vybraných agilních metodik 4) Agilní projektový management 5) SCRUM 6) Vývoj vlastní aplikace Scrum board.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Agile Alliance [online]. 2010 [cit. 2010-06-14]. Dostupné z WWW: <http://www.agilealliance.org>.**
2. **BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů . Praha : Grada, 2005. 163 s. ISBN 80-247-1075-7.**
3. **KADLEC, Václav. Agilní programování : Metodiky efektivního vývoje softwaru. Brno : Computer Press, 2004. 278 s. ISBN 80-251-0342-0.**
4. **Mountain goat software [online]. 1998-2010 [cit. 2010-06-14]. Dostupné z WWW: <http://www.mountaingoatsoftware.com/>.**
5. **SCHWABER, Ken. Agile Project Management With Scrum. United States : Microsoft Press, 2004. 163 s. ISBN 978073561993.**

Vedoucí diplomové práce: **Ing. Pavel Jirava, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **4. října 2010**

Termín odevzdání diplomové práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 4. října 2010

PROHLÁŠENÍ AUTORA

Prohlašuji:
Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 15. 4. 2011

Bartůňková Květoslava

Poděkování

Tímto bych chtěla poděkovat vedoucímu mé práce, panu Ing. Pavlu Jiravovi, Ph.D., za pomoc, ochotu a cenné rady poskytnuté během zpracování diplomové práce.

Dále bych ráda poděkovala mé rodině, která mě podporovala v průběhu celého studia.

SOUHRN

Diplomová práce se zaměřuje na popis a využití tradičních, ale především agilních postupů v projektovém řízení, přičemž se jedná pouze o oblast vývoje informačního systému. Největší důraz je kladen na agilní metodiku SCRUM, dle které byla vyvinuta aplikace SCRUM Board. Ta nahrazuje fyzickou tabuli, využívanou v této metodice, za její elektronickou verzi.

KLÍČOVÁ SLOVA

tradiční metodiky, agilní metodiky, projektové řízení, SCRUM, SCRUM Board

TITLE

Agile methods, SCRUM, in Project Management

ABSTRACT

This thesis focuses on description and the use of traditional but mainly agile methods in project management while it is only about the devolopment of an information system. The biggest attention is on SCRUM. In this way the aplication SCRUM Board was created. This aplication replaced a physical board which is used in this method in its software version.

KEYWORDS

traditional methods, agile methods, project management, SCRUM, SCRUM Board

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM GRAFŮ

SEZNAM TABULEK

Úvod

Při vývoji informačního systému je postupováno dle metodik, které definují jednotlivé fáze průběhu projektu. Do roku 2000 byly používány především tradičními metodiky, avšak od roku 2001 se především v zahraničí, ale v posledních letech i v České republice, postupně přechází k vývoji softwaru pomocí agilního přístupu. V této práci se v rámci projektového řízení vždy jedná o takové projekty, které vyvíjí informační systém nebo nějaký druh softwaru.

V první části práce jsou popsány tradiční přístupy, jako je vodopádový a spirálový model životního cyklu. Dnes jsou tyto modely využívány spíše k ukázce, jak se při tvorbě postupovat nemá, avšak v minulosti byly tyto postupy hodně využívány a i dnes se lze setkat s projekty, které staví právě na některém tomto přístupu. Spirálový model je jakési rozšíření vodopádového modelu, ale i ten již není dostačující. Proto se vývoj ubírá k agilnímu programování.

Použití agilního přístupu je popsáno v druhé kapitole této práce. Je zde popsán především vznik a cíle agilních metodik a také jejich využívání v ČR. Nejdůležitější částí je však popis rozdílu od tradičního přístupu. Rozdíl spočívá v tom, že velký důraz je kladen především na čas, po který je projekt nutné dokončit a také na zdroje, které je nutné dodržet. Většinou jsou vytvořeny rychle a v průběhu práce lze měnit požadavky dle potřeb zákazníka. Kdežto dle tradičních modelů je nutné dodržet především specifikované požadavky definované zákazníkem nebo zadavatelem projektu.

V další části práce se nachází konkrétní popis vybraných agilních metodik. Byly vybrány především ty, které jsou často využívané a vykazují dobré výsledky. Patří sem metodiky: Rational Unified Process, Unified Software Development Process, Extrémní programování, Lean Development, Feature Driven Development a dále ostatní menší, které jsou využívány méně. Každá metodika má svá specifika, proto záleží na druhu vyvíjeného softwaru a především na samotné organizaci, který přístup přijme jako nejlepší. Nejpoužívanější metodikou se stala v poslední době metodika SCRUM, která klade důraz především na zvýšení efektivity vývoje informačního systému. Tento přístup je detailně popsán v kapitole pět této práce a byla mu věnována největší pozornost.

Dle SCRUM vznikla aplikace SCRUM Board, její vývoj je popsán v poslední části práce. Jedná se především o názorné demonstrování použití daného přístupu k vývoji softwaru. Je nutné definovat požadavky, role i funkce uživatelů a programátorů a především

náležitosti a cíle celé aplikace. Dále je třeba provést ošetření bezpečnosti a otestovat všechny funkce vyvinutého nástroje tak, aby výsledek odpovídal fyzické tabuli, která je v metodice SCRUM využívaná.

Cílem práce je tedy rozlišit tradiční a agilní přístup, definovat přínosy, výhody a nevýhody obou a zdůvodnit, proč je v dnešní době dobré přecházet právě k agilním metodikám vývoje softwaru. Dále je cílem popsat používané agilní metodiky, především SCRUM a názorně vytvořit aplikaci, která demonstruje použití tohoto přístupu.

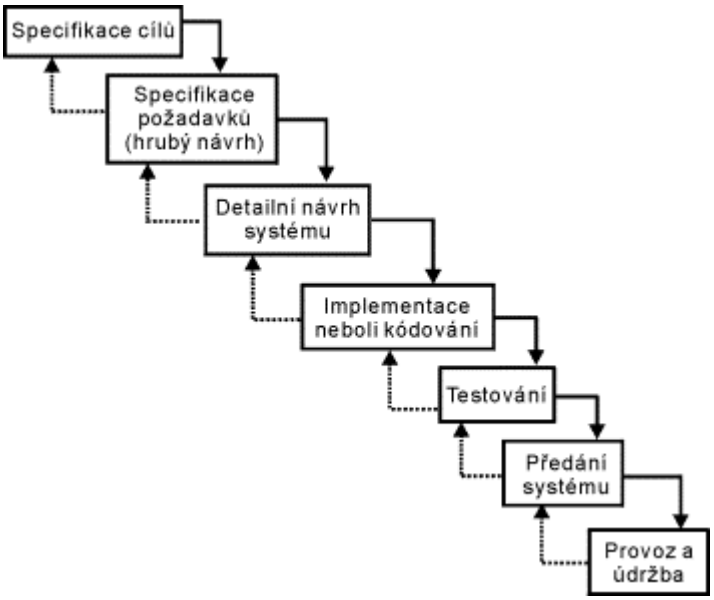
1. Tradiční metodiky

Tato práce je zaměřena na popis rozdílu mezi tradičními a agilními metodikami, především na popis metodiky SCRUM a vývoj aplikace (softwaru), která nahrazuje fyzickou tabuli s rozpisem průběhu projektu, proces projektového řízení tedy bude brán v úvahu vždy ve spojení s vývojem softwaru. Projektové řízení se stalo samostatným oborem. Cílem tohoto oboru je systematický výzkum úspěšných i neúspěšných projektů. V projektovém řízení je nutností dodržovat určité normy a postupy a na účastníky projektového řízení jsou také kladeny určité požadavky. Z celého procesu vzejde metodika, která shrnuje všechna doporučení vzniklá z pozorování projektů. V této práci se jedná o projekty, které se zabývají tvorbou informačního systému, čili programátor pracuje na základě těchto doporučení na vývoji softwaru. Metodiky lze rozdělit na tradiční (rigorózní) a agilní. U tradičních metodik je kladen silný důraz na úvodní fáze – monitorování zákazníka, monitorování systému, návrh, apod. Vývoj zde obvykle trvá delší dobu a je méně efektivní. V této části práce se nachází stručný popis vybraných tradičních metodik, které jsou využívány při vývoji informačního systému – vodopádový model životního cyklu a spirálový model životního cyklu. [9][5]

1.1 Vodopádový model životního cyklu

Jedná se o tradiční a často používaný model životního cyklu vývoje informačního systému. V dnešní době je však tento model využíván především k ukázce, jak vývoj probíhat nemá. Je vyhovující pouze pro jednodušší projekty, výhodou je jednoduchost a snadné použití. [9]

Na obrázku 1 je zobrazena struktura modelu. Jedná se o jednu z možných alternativ zobrazení tohoto modelu.

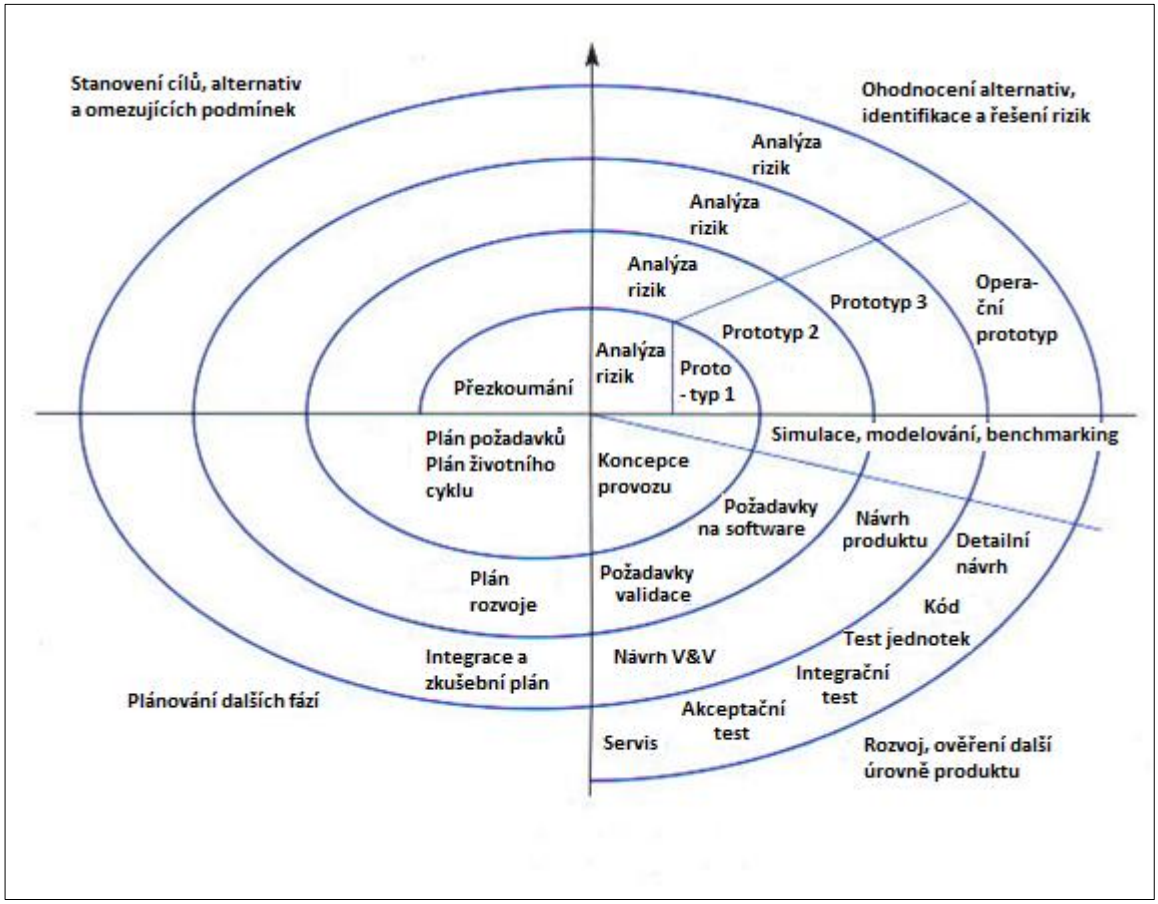


Obrázek 1 - Vodopádový model životního cyklu (zdroj [6])

Principem modelu je, že následující fáze začne až po skončení fáze předchozí. Nevýhodou je, že i při zjištění nedostatků se lze vrátit pouze do předchozí úrovně. Zákazník může do vývoje zasahovat pouze na začátku a na konci, což je velice nedostačující. U většiny případů není možné definovat požadavky už na začátku vývoje. Jakmile je systém předán zákazníkovi, nastává nejdelší fáze celého vývoje – údržba. Toto všechno jsou důvody, proč není tento model již tolik využíván. Největším problémem je však to, že problémy jsou zjištěny až po naprogramování celého systému, následují změny návrhu, opětovné přeprogramování a tím je celý projekt zpožděn. Nejedná se přímo o metodiku, popisuje spíše které kroky je nutné při vývoji dodržet, oproti metodice, která většinou obsahuje kromě kroků i další informace co v kterém okamžiku přesně dělat. V minulosti byl však tento model hojně využíván, a proto byl zařazen mezi tradiční přístupy vývoje softwaru. [9][5]

1.2 Spirálový model životního cyklu

Rozdíl oproti vodopádovému životnímu cyklu je v zavedení iterací a analýzy rizik. Tento model definoval v roce 1985 Barry Boehm. Na obrázku 2 jsou zobrazeny čtyři iterace – určení cílů, alternativ a omezení, dále iterace hodnocení alternativ, identifikace a řešení rizik, třetí iterací je vývoj a ověření další úrovně produktu, a poslední iterací je plánování další fáze. V první iteraci se definují rizika a definují se možnosti řešení. Druhá iterace řeší definici požadavků a během třetí iterace se vytváří detailní návrh řešení. Poslední iterace zahrnuje implementaci navrženého řešení a testování. [5]



Obrázek 2 - Spirálový model životního cyklu (zdroj: autor - upraveno dle [11])

Vývoj probíhá od středu spirály ven. Levý horní kvadrant znázorňuje stanovení cílů, alternativ, omezujících podmínek, apod. Z levého horního kvadrantu se průběh posouvá do pravého horního kvadrantu, kde je třeba provést analýzu rizik. Následuje spodní pravý kvadrant, ve kterém probíhá samotná realizace, která zahrnuje implementaci, vývoj, specifikaci požadavků, testování, omezování apod. Poslední kvadrant zajišťuje následné plánování dalšího postupu. [9]

Časový průběh spirály lze rozdělit do čtyř cyklů. Spirálový model klade důraz především na rizika. Stanovení a definování rizik, která by mohla ovlivnit správný chod systému je úkolem prvního cyklu spirály. V tomto cyklu je také zpracována základní osnova vývoje a rozhoduje se zde o použití konkrétních metod. Úkolem druhého cyklu je specifikovat požadavky na systém. Vytvoření a ověření detailního designu patří do třetího cyklu průběhu spirály a poslední cyklus – čtvrtý zabezpečuje implementaci, testování a integraci. [9]

Největší výhodou spirálového modelu je nezávislost na konkrétní metodice. Možnost neúspěchu je zde ošetřena analýzou rizik, což je také silnou výhodou modelu. Mezi další klady tohoto modelu patří komplexnost a vhodnost pro složité projekty. Na druhou stranu model má i své nevýhody. Lidé, kteří stanovují rizika, musejí být experty ve svém oboru, jinak může celý projekt skončit krachem. Model je komplikovaný a změny požadavků lze provádět pouze po dokončení jednoho cyklu. [9]

2. Agilní metodiky

Dle [9] je jediným cílem těchto metodik co nejdříve a co nejlevněji dodat zákazníkovi produkt, který splňuje všechny požadavky, je kvalitní a umožňuje provádět i následnou údržbu a podporu. Vytvářet rychleji a efektivněji je hlavním požadavkem dnešní doby.

2.1 Vznik agilních metodik

V roce 2001 podepsali představitelé nového přístupu *Manifest agilního vývoje softwaru*. Vytvořili tak *Alianci pro agilní vývoj softwaru*. V manifestu jsou obsaženy čtyři stanovení takové, že tvrzení na levé straně mají větší váhu než tvrzení na pravé straně. Manifest dle [5] zní:

„Odkryli jsme lepší způsob, jak vyvíjet software, využíváme jich a chceme pomoci ostatním, aby jej také používali. Preferujeme:

- Individualitu a spolupráci před procesy a nástroji,
- fungující software před podrobnou dokumentací,
- spolupráci se zákazníkem před sjednáváním smluv,
- reakci na změnu před plněním plánu.“

Pod tímto manifestem je podepsáno sedmnáct nezávislých odborníků z oblasti softwarového inženýrství.

2.2 Cíl agilních metodik

Především u projektů informačních systémů či internetových projektů je největším problémem rychlost vývoje. Nelze, aby společnost vyvíjela např. webovou aplikaci dva roky, protože konkurence mezitím spustí dvě totožné služby a pro společnost bude velice složité, nebo i nemožné, získat tyto zákazníky na svou stranu. [9]

Proto začali být po roce 2000 prosazovány metodiky s co nejrychlejším vývojem softwaru. Pojem agilní metodika je skupina metodik, kde prověření správnosti navrženého systému probíhá tak, že systém je co nejrychleji vyvinut, předložen zákazníkovi a posléze je upravován na základě zpětné vazby. V posledních letech lze hovořit o velkém pokroku ve vývoji softwaru. Firmy se snažily zavádět nové technologie, přesto podle průzkumu dle [12] je zrušeno 19 % informačních projektů pro jejich neúspěch a u 46 % projektů softwaru nebyly dodrženy stanovené náklady, čas apod. Firmy tedy většinou zavádějí ISO nebo Agilní

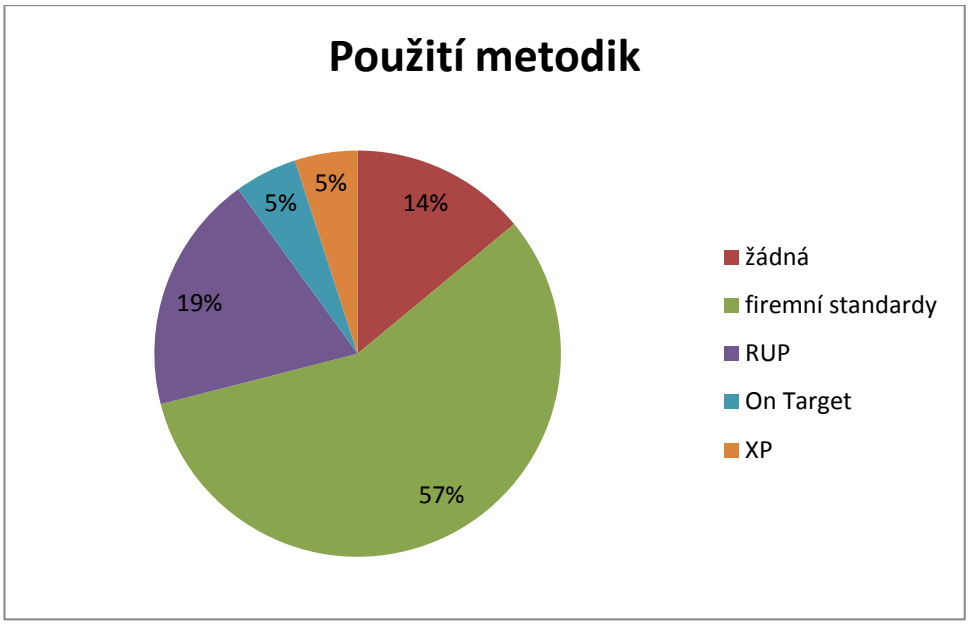
metodiky. Před zavedením agilních metodik si většinou společnosti pokládají následující otázky. Jak zajistit, aby zaměstnanci fungovali jako tým? Jak zajistit soutěž mezi týmy ale zároveň i spolupráci? Jak zajistit včasné dokončení projektu? Agilní metodiky patří mezi metody jak řešit tyto otázky. Často se stává, že pracovník nadhodnotí úkol i na dvojnásobek, aby si zajistil klidný průběh vývoje, nebo naopak zapomene na důležité části a projekt není dokončen včas. Agilní metodiky také definují práci s lidmi na projektu, správnou motivaci a plánování projektu. Také pomáhají se zapojením zákazníka do celého procesu. [9][29][31]

2.3 Stav používání agilních metodik v ČR

Ve světě se agilní metodiky prosazují stále více. V ČR se teprve začínají dostávat do podvědomí firem a odborníků na informační a komunikační technologie. Mezi agilní metodiky patří [9]:

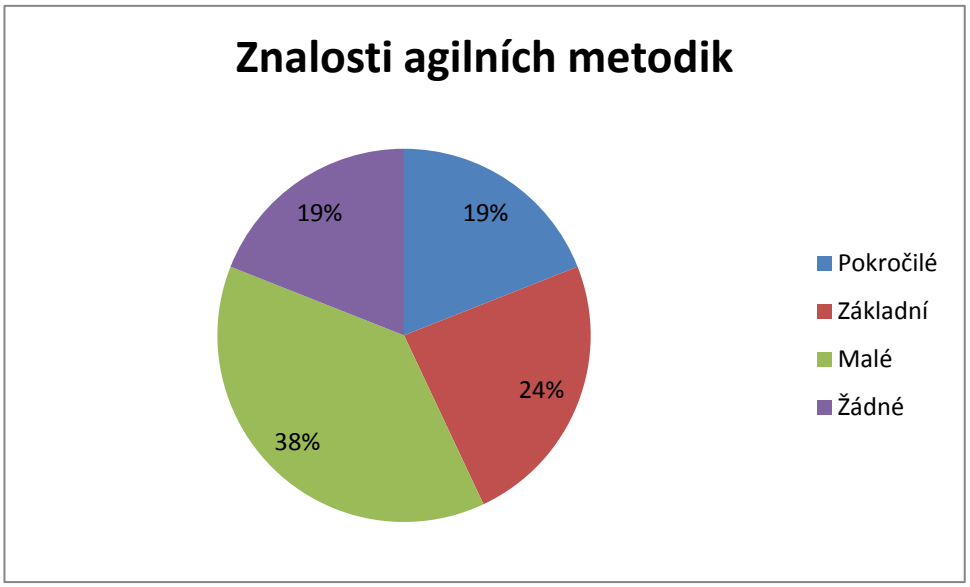
- Dynamic Systems Development Method (DSDM),
- Adaptive Software Development (ASD),
- Feature–Driven Development (FDD),
- Extrémní programování (Extreme Programming, XP),
- Lean Development,
- SCRUM,
- Crystal metodiky,
- Agilní modelování (Agile Modeling).

V roce 2006 proběhl na VŠE Praha Průzkum používání agilních metodik v ČR. Graf 1 zobrazuje, že agilní metodiky v té době byly využívány velice málo. Od té doby se stav zlepšil, avšak stále se ČR nemůže rovnat s využíváním agilních metodik ve světě. Metodiky, které se nazývají On Target, jsou cílované, tzn. zaměřené na cíl projektu. [9]



Graf 1 - Použití metodik ve firmách v ČR (zdroj: autor - upraveno dle [6])

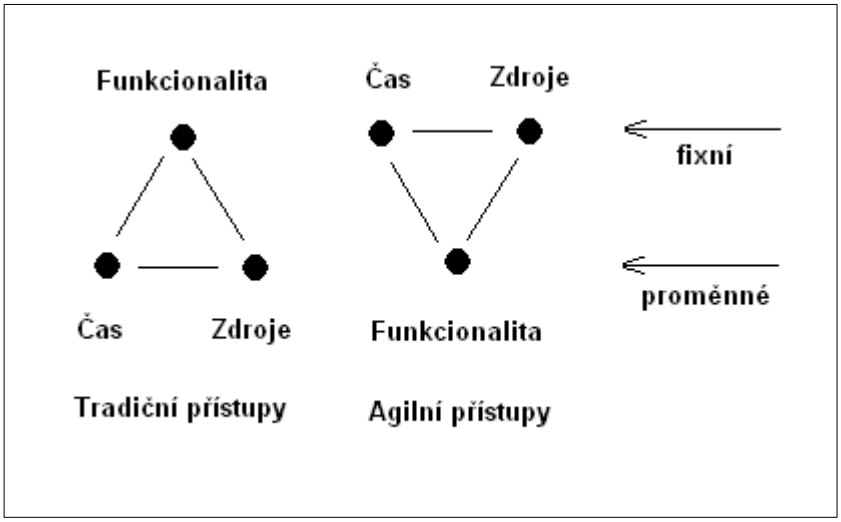
Příčinou malého využívání agilních metodik může být nízký rozsah znalostí o této tématice. K průzkumu byli vybíráni pracovníci s vysokoškolským vzděláním v oblasti informatiky nebo s dlouhodobou odbornou praxí. Průzkum ukázal, že pouze 43% dokáže vysvětlit pojem agilní metodika – graf 2. Stav se opět od roku 2006 zlepšuje. Důvodem může být fakt, že na vysokých školách se stále více vyučuje o agilních metodikách. [6]



Graf 2 - Znalosti agilních metodik (zdroj: autor - upraveno dle [6])

2.4 Princip agilních metodik

Tradiční metodiky vycházejí z definování požadavků na systém. Tyto požadavky byly definovány na počátku vývoje a jsou fixní (neměnné), proměnné veličiny jsou zde čas a zdroje. Lze říci, co bude přesně systém umět, avšak náklady a čas nejsou odhadnuty. Na druhou stranu agilní metodiky využívají jako proměnnou veličinu požadavky, kdežto čas a náklady jsou pevně stanoveny. Rozdíl je zobrazen na obrázku 3. Mezi základní principy pro dodržování dané posloupnosti agilních metodik patří plán sestavený tak, aby nové funkce mohly být do vývoje dodávány často, mnohdy i denně. Dalším důležitým principem je správná a přímá komunikace v týmu pracovníků. Jedná se o časté schůzky, které dříve odhalí existující problémy. Nepřetržitá komunikace se zákazníkem je neméně důležitým principem správného fungování agilních metodik. Uživatel by měl komunikovat s týmem, podílet se jak na návrhu, tak na testování, což je posledním důležitým principem. Testování musí probíhat opakovaně a průběžně, automatizovaně před implementací testovaných částí. [9]



Obrázek 3 - Rozdíl mezi tradičními a agilními programovacími přístupy (zdroj: [16])

3. Popis vybraných agilních metodik

Tyto metodiky vykazují velice dobré uplatnění na projektech, které je třeba vytvořit rychle a je u nich patrná častá změna požadavků. Většina těchto metodik byla vybrána dle průzkumu jejich používání ve světě.[1] Každá z metodik má své výhody, nevýhody a použití. V poslední době se začíná do popředí dostávat metodika SCRUM, která bude popsána v kapitole páté podrobněji.

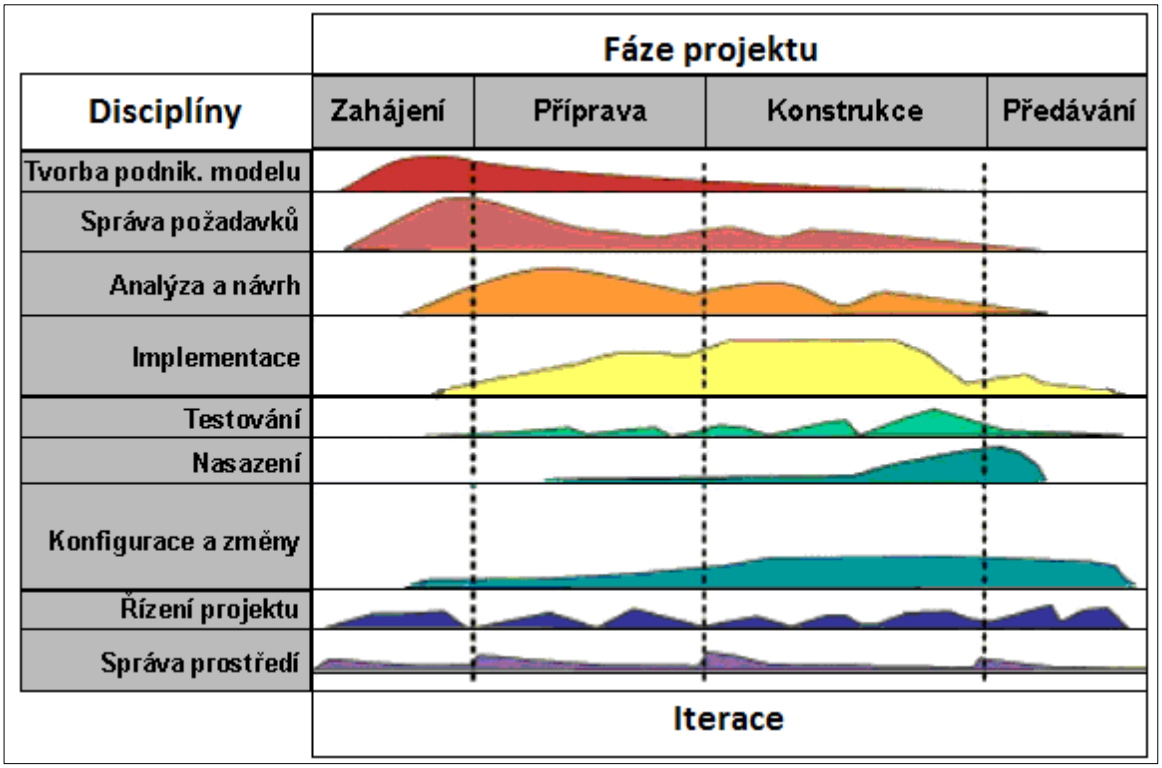
3.1 Metodika Rational Unified Process

Jedná se o standard mezi metodikami. Zpočátku byla zařazována mezi tradiční, avšak postupem času byla doplňována o praktiky agilního přístupu. Pomocí RUP lze vytvářet postupy pro všechny typy projektů až po lehkou, agilní metodiku. Proto je zařazena mezi agilní. [5]

3. 1.1 Vznik a popis metodiky RUP

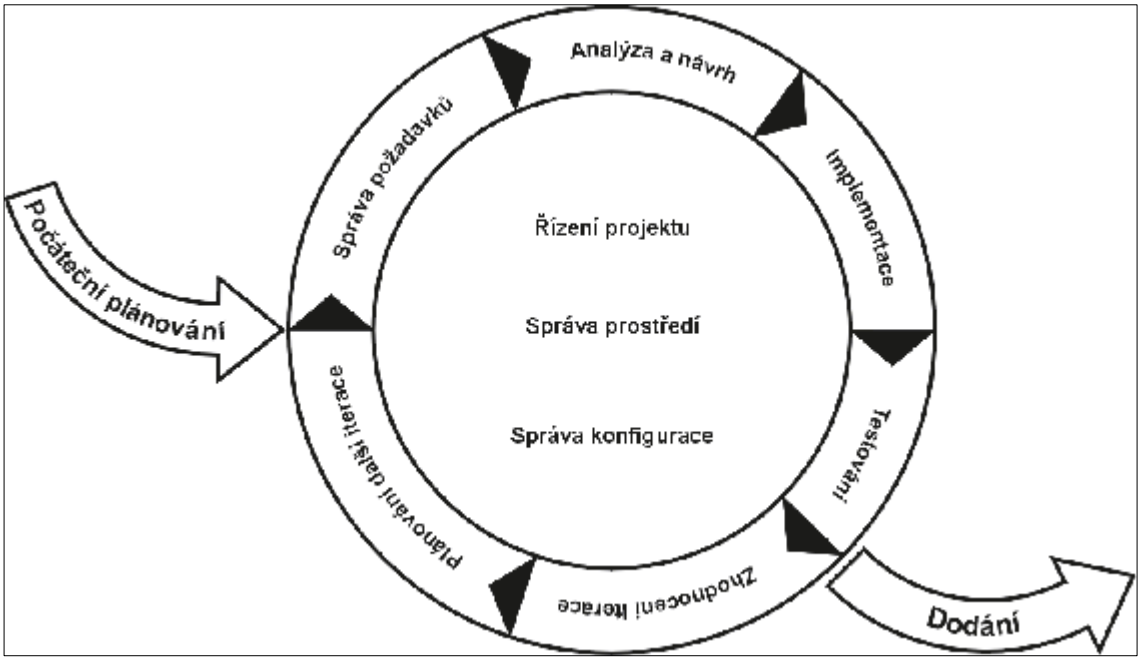
RUP byla vyvinuta společností *Rational Software*. Jedná se o velmi robustní a propracovanou metodiku. Sama společnost mluví o svém produktu jako o instruktorovi, který nabízí metodické pokyny, instrukce, šablony, příklady vývoje, dokumenty, modely, manuály, programový kód apod. Jedná se o objektově orientovanou metodiku, základním elementem je případ použití. Vývoj probíhá v iteracích a každou iteraci lze rozdělit na čtyři fáze – zahájení, příprava, konstrukce, předávání. Jeden vývojový cyklus je zobrazen na obrázku 4. Definice účelu, rozsahu projektu a obchodního kontextu probíhá ve fázi zahájení. Následuje fáze přípravy, kdy vývojáři analyzují potřeby projektu a zákazníka. Ve fázi konstrukce probíhá vývoj designu a tvoří se zdrojové kódy. V poslední fázi předávání proběhne předání projektu zákazníkovi nebo do dalšího vývojového cyklu. Počet cyklů je závislý na rozsahu projektu. [9]

[1] Scott Ambler provedl Průzkum využívání agilních metod v roce 2006 ve Spojených státech amerických.



Obrázek 4 - Jeden vývojový cyklus metodiky RUP (zdroj: autor – upraveno dle [14])

Metodika RUP klade velký důraz na definici, vyhledávání, analýzu a řízení rizik. Na základě této analýzy vzniká plán iterací. Iterativní vývoj je zobrazen na obrázku 5. Při každé činnosti vznikají tzv. meziprodukty, což jsou dokumenty a formální náležitosti. Patří sem např. obecný popis, UML reprezentace, role pracovníků, časová náročnost a doporučené rozvržení práce. Kromě těchto meziproduktů zde existují tzv. mezníky. Tyto mezníky značí ukončení jednotlivých fází cyklu. Jedná se o body, v nichž se vyhodnocuje dosažený pokrok, jsou zkontrolovány správnosti směřování projektu a především se zde rozhoduje o dalším postupu. [9]



Obrázek 5 - Iterativní vývoj (zdroj: [26])

3. 1.2 Výhody a nevýhody

Obecnost, šíře a mohutnost patří mezi nejsilnější stránky RUP. Na základě této výhody je metodika použitelná pro širokou škálu projektů. Rizika v průběhu projektu lze snadno a rychle rozpoznat a to především díky iterativnímu přístupu. Cyklický průběh zajišťuje údržbu systému, podporu a aktualizaci. Systém je tak celistvě zabezpečen. Další velkou výhodou je objektově orientovaný přístup v průběhu vývoje systému. Celý systém lze provázat s notací UML, což zajišťuje intuitivní ovládání a realističtější pohled na celý vývojový cyklus. Tato notace zajišťuje provázanost zápisu modelů a vztahů. RUP patří mezi známé a často využívané metodiky. Pokud tedy společnost využívá při tvorbě projektu tuto metodiku, získává tím konkurenční výhodu. Ostatní společnosti jsou si vědomy, že využití RUP zajišťuje vysoké procento správnosti a podrobnosti. [9][13]

Výhody však mohou v některých případech přejít na nevýhody. Například šíře, obecnost a mohutnost může způsobit problémy u menších týmů a rozsáhlých projektů. Vzhledem k tomu, že tým se zabývá dlouho zkoumáním metodiky, může celý proces ztrácet na efektivitě. V neposlední řadě je velkou nevýhodou vysoká cenu pořízení produktu. V době psaní této práce je dle [13] cena za jednu licenci stanovena na něco málo přes 1.000 USD. [9][13]

3.2 Metodika Unified Software Development Process

Jedná se o metodiku, která je velice podobná předešlé RUP. Také řeší problém otázek kdo, co, kdy a jak a je propojena s jazykem UML. Vývoj projektu řídí případy užití a rizika a probíhá stejně jako u předešlé iterativním způsobem, platí však, že délka jedné iterace by neměla být delší než dva až tři měsíce. Důležitý je zde textový popis použití od uživatelů a identifikace rizik. Tento agilní přístup obsahuje pět pracovních postupů. Prvním krokem je stanovení požadavků od zákazníka, následuje analýza, návrh a implementace, kde je sestavován programový kód systému. Poslední z postupů je testování, kde je nutné porovnat očekávané funkce od výstupu. Důležitým rozdílem mezi metodikou RUP a metodikou Unified Software Development Process je, že pořízení druhé jmenované je bezplatné.[9]

3.3 Extrémní programování

Tato metodika vychází z tvrzení, že zdrojový kód je jediným správným zdrojem informací. Extrémní programování je určeno pro menší až střední týmy o počtu 2 – 10 programátorů. [17]

3. 3.1 Vývoj a charakteristiky extrémního programování

V roce 1999 představil základní charakteristiky této metodiky Kent Beck. Jedná se o metodiku, která umožňuje flexibilnější a přizpůsobivější proces vývoje. Základními charakteristikami je vysoká účinnost, pružnost a předvídatelnost systému. Jde o nepříliš rizikový a zábavný vývoj informačního systému. Extrémní programování rozšiřuje obrázek 3 o čtvrtou proměnnou – šíře zadání. Na tuto proměnnou se klade největší důraz. To znamená, že ke kvalitě (funkcionalitě), času a nákladům (zdrojům) přibude šíře zadání. Hodnoty prvních tří proměnných definují zákazníci nebo manažeři, výslednou proměnnou - šíři zadání stanovuje vývojový tým. Pokud by i čtvrtou proměnnou definovali zákazníci či manažeři, prudce by klesla kvalita celého systému. Pokud vyžadují jinou hodnotu šíře zadání, je nutné, aby změnili základní tři proměnné. Vývojový tým tedy stanoví proměnou tak, aby zachoval požadovanou kvalitu za stanovené náklady ve stanoveném čase. Zákazníci nebo manažeři tedy musí stanovit priority jednotlivých třech proměnných a vývojový tým se soustředí nejdříve na dodávku funkcí s vyšší prioritou. [17]

Mezi základní a nejdůležitější pojmy v této metodice patří dle [9]:

- Komunikace,
- jednoduchost,

- zpětná vazba,
- odvaha,
- respekt.

Pokud vzniknou nějaké problémy nebo zpoždění, bývá velká pravděpodobnost, že dochází ke špatné komunikaci v týmu odborníků. Programátoři často nemají tušení, že je informace důležitá. Když však nastane problém, jsou vyvedeni z omylu. Extrémní programování definuje celou řadu postupů, které pomáhají při udržování komunikačních kanálů, ale také definuje tzv. „kouče“, což je zvláštní osoba z projektového týmu, která kontroluje výpadky komunikace. [17]

Pro definování jednoduchosti v extrémním programování je nutné uvést otázku: „Jak vypadá nejjednodušší věc, která ještě může fungovat?“ Existuje zde určitá míra rizika, avšak je lepší projektovat jednodušší a ekonomičtější systémy. Důležité je v tomto případě nemyslet na budoucí vývoj. [9]

Zpětná vazba je úzce spojována s komunikací. Správná komunikace sama o sobě zajišťuje informace pro zpětnou vazbu. Jedná se o informace o aktuálním stavu vývoje systému, o situaci ve vývojovém týmu, o požadavcích a potřebách zákazníka a o dalších skutečnostech, které jsou důležité při vývoji systému. Základem je správné testování, které ukáže, zda je systém v pořádku nebo zda je nutné provést opravy či změny. [17]

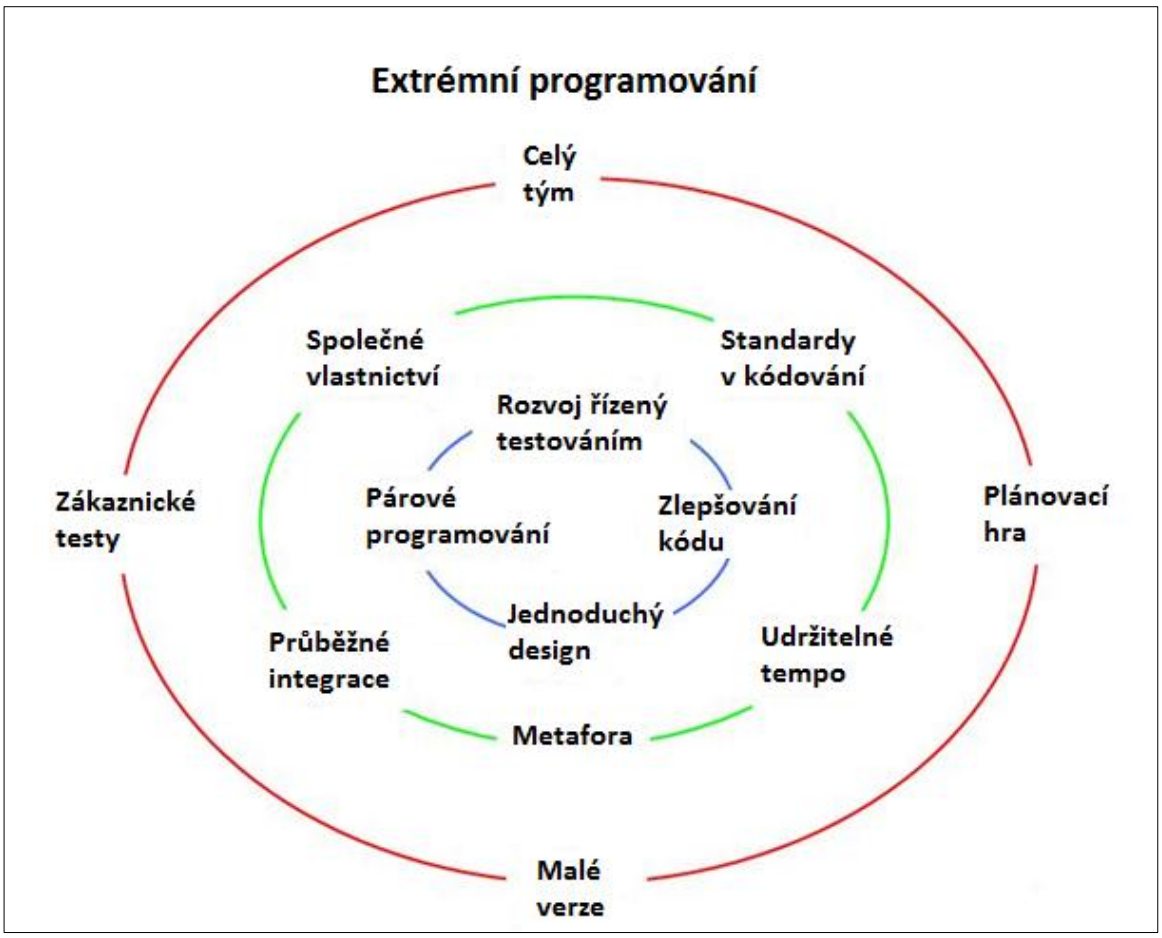
Další hodnotou extrémního programování je odvaha, bez které se nedá extrémní programování využívat. Chybu, kterou tým nalezne ve vývoji, je třeba opravit a to za jakoukoli cenu. Může se tedy stát, že bude nutné přepracovat i celý zdrojový kód z důvodu této chyby. Odvaha je tedy velice důležitou součástí celého vývoje, pokud se vývoj nedaří, je třeba znovu a znovu zkoušet nové postupy a možnosti. [9]

Pod těmito čtyřmi hodnotami se nachází hodnota poslední, hlubší – respekt. Jednotlivci v týmu musejí spolupracovat s ostatními členy týmu, zajímat se o to, na čem který člen právě pracuje a snažit se mu jeho práci co nejvíce usnadnit, nebo mu pomoci. Pokud se v týmu každý bude starat pouze o svoji část vývoje, pravděpodobně dojde k chybám a k pádu celého systému. [9]

3. 3.2 Postupy extrémního programování

Na obrázku 6 lze sledovat jednotlivé postupy, jak probíhá extrémní programování. Každý, kdo jakoukoliv měrou přispívá do projektu, je součástí týmu. Zákazník je zúčastněný tak, že se nachází přímo v místě, kde je tým a každý den spolupracuje s tvůrci projektu.

Stanovuje priority, požadavky a dá se říci, že řídí projekt. Do týmu patří programátoři, analytici, testeři, existuje zde trenér, který pomáhá udržet tým na správné cestě. V týmu se nachází také manažer, který kontroluje zdroje, koordinuje činnosti při vývoji a komunikuje s externím prostředím. [14]



Obrázek 6 - Postupy a cykly extrémního programování (zdroj: autor - upraveno dle [19])

Dále je zde dvanáct postupů, které musejí být dodrženy. V první řadě jde o plánovací hru, kde je vytvořen plán. Tento plán říká, čeho bude dosaženo do data spuštění a také co se bude dělat dál. Dalším postupem je tzv. malá verze, kdy při každé iteraci je uvolněna nová verze projektu, což napomáhá zákazníkovi při pohledu na tvorbu systému, zda se tým ubírá správným směrem. Přítomnost zákazníka v týmu je stěžejní v extrémním programování. Zákazník testuje, zda požadované funkce fungují a zda jsou funkce správně implementovány. [14]

Důležitými postupy v extrémním programování je společné vlastnictví, standard psaní zdrojového kódu, nepřetržitá integrace, metafora a čtyřicetihodinový pracovní týden. Společné vlastnictví definuje, že za celý systém přijímá zodpovědnost celý tým. Jelikož každý

z programátorů může pracovat na jakékoliv části vývoje, opravovat nedostatky či chyby, je nutné, aby byly přijaty určité standardy při psaní zdrojového kódu. Aby každý věděl, co od systému očekávat, jak se má chovat a jaké jsou cíle, je zde metafora, což je vize, jak program funguje. Důležité je také pracovníky nepřetěžovat, pokud např. programátor pracuje hodně s přesčasy, jeho produktivita klesá, je unavený a častěji dělá chyby. Po dokončení nějakého úkolu probíhá ihned integrace, testování a sestavování, a to i několikrát denně. [14][18]

Posledními důležitými postupy jsou refaktorizace, párové programování, jednoduchý design a testování. Při refaktorizaci se odstraňují duplicity v systému, tím se zvyšuje rychlost vývoje a zvyšuje se i rychlost, jakou jde projekt kupředu. U počítače vždy sedí dva programátoři, tím se zajistí, že celý kód je znovu přezkoumán, zkontrolován a zlepšuje se i vzhled celého systému. Programátoři se snaží o co nejjednodušší design, který je zachován po celou dobu vývoje. Jediným způsobem, jak zabránit vzniku chyb nebo jak odstranit nedostatky, je po každé iteraci provést testování. Netestují pouze programátoři, ale i zákazníci, kteří testují správnost požadovaných funkcí. [9][14]

3. 3.3 Výhody a nevýhody extrémního programování

Tato metodika zdůrazňuje především spokojenost zákazníka. Programátoři mohou reagovat na změny, které si vyžaduje zákazník a to i v konečných fázích životního cyklu vývoje produktu. Jedná se zde o spolupráci. Vývojáři, zákazníci i manažeři jsou si rovni. Extrémní programování se vyznačuje svojí jednoduchostí, rychlostí a efektivností. Na druhou stranu právě jednoduchost může být brána jako nevýhoda. Ne každý vývojář dokáže přiznat, že něco neví, neumí nebo nechápe. Spousta lidí nedokáže dělat v páru, nedokáže dobře a jasně komunikovat. V systému našich vzdělávacích zařízení je člověk veden spíše k individualismu, což je v této metodice nutné eliminovat a spolupracovat v týmu. [33][19]

3.4 Lean Development

Autorem této metodiky je Robert Charette. Pochází z principů Lean Manufaturing a Total Quality Management. Tyto principy propagovala při výrobě aut automobilová společnost Toyota. V naší zemi principy této metodiky využívala firma Baťa. Cílem je vytvoření softwaru s třetinovou lidskou prací, s třetinovým časem, s třetinou investic a třetinovým úsilím přizpůsobit se novým podmínkám. Lean Development dodržuje deset pravidel [5]:

1) Odstranění zbytečného (to co konečnému systému nepřináší žádnou hodnotu),
2) minimalizace zásob,
3) minimalizace času vývoje,
4) vývoj tažený poptávkou (požadavky uživatele jsou důležitým ukazatelem),
5) rozhodnutí provádějí pracovníci na nejnižší úrovni managementu,
6) uspokojit požadavky zákazníků nejen teď, ale i v budoucnosti,
7) zavedení zpětné vazby,
8) neprovádět optimalizaci v době změn,
9) partnerství s dodavateli,
10) budování kultury pro neustálé zlepšování.

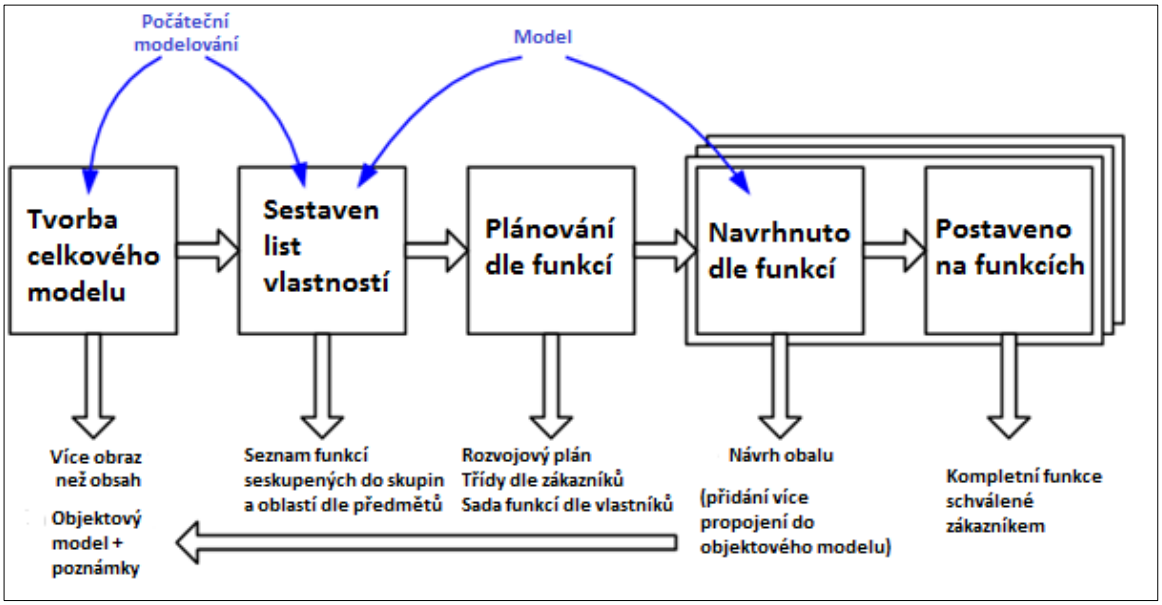
Lean development, na rozdíl od většiny metodik, které se zabývají taktickou úrovní managementu, pracuje na strategické úrovni s vazbou na podnikovou strategii. [5][33]

3.5 Feature Driven Development

Jedná se o agilní metodiku autorů Jeff de Luca a Peter Coad. Metodika obsahuje více formálních požadavků a kroků oproti extrémnímu programování, nachází se zde přesné sledování jednotlivých postupů. Feature Driven Development lze rozdělit na dvě fáze, a to získávání seznamu vlastností (features) pro implementaci a implementaci těchto vlastností jednu po druhé. Vše je založeno na iterativním vývoji, iterace jsou kratší, přibližné trvání jedné iterace je asi dva týdny. Po každé iteraci se předává zákazníkovi fungující meziprodukt. Tím zůstává zákazník klidnější, že vývoj funguje a jde kupředu. Důraz je zde nejvíce kladen na sledování vlastností. Příkladem vlastností může být v aplikaci internetového obchodu např. Přidání zboží do košíku, Výpis zboží v košíku, Spočtení celkové ceny, atd. K výpisu vlastností se využívají UML diagramy. [20]

Na obrázku 7 jsou zobrazeny fáze vývoje softwaru metodikou Feature Driven Development. V první fázi Vytvoření celkového modelu probíhá modelování diagramu tříd v jazyce UML. Následuje fáze Vytvoření seznamu vlastností, kdy je nutné sestavit vyčerpávající seznam, tyto vlastnosti roztřídit do skupin tak, aby spolu vybrané úzce souvisely a logicky do skupiny patřily. Seznam lze do budoucna rozšiřovat a měnit, protože není reálné, aby byly všechny definovány hned zpočátku vývoje projektu. Další fází je Plánování, kde vzniká plán dalšího vývoje, eviduje se také mezní datum ukončení vývoje, což je nejdůležitější mezník celého projektu. Každé skupině vlastností se přiřadí odpovědný

programátor, který zodpovídá za úspěšné dokončení dané třídy do daného data. V této fázi se také nachází plán pořadí implementace jednotlivých vlastností. Čtvrtá fáze je Návrh, kdy vzniká detailní návrh pro implementaci a vzniká plán pro realizaci. Implementace je poslední fáze vývoje. Probíhá zde realizace podle vlastností, testování, příprava integrace a sestavení projektu. Čtvrtá a pátá fáze jsou iterativně opakovány a zabírají přibližně 75% celkového trvání projektu. Z páté fáze se pak lze vrátit do první. [9]



Obrázek 7 - Posloupnost vývojových fází metodiky Feature Driven Development (zdroj: autor – upraveno dle [9])

3.6 Ostatní

Ostatní metodiky nejsou tolik využívány jako předešlé, avšak je třeba je zahrnout do agilního přístupu. Jedná se o Crystal metodiky, Dynamic Software Development Method a Adaptive Software Development.

3. 6.1 Crystal metodiky

Jedná se o seskupení, jejichž hlavním rysem je lehkost produktu a správná komunikace v týmu. Autorem je Alistair Cockburn, který definoval jednotlivé přístupy v této rodině metodik dle barev, a to od nejlehčí *Clear* (průhledná), následuje *Yellow* (žlutá), *Orange* (oranžová), *Red* (červená), *Maroon* (hnědočervená), *Violet* (fialová) atd. Prvky jednotlivých metodik jsou upravovány pro každý projekt. [5]

3. 6.2 Dynamic Software Development

Jsou zde definovány tři fáze vývoje, a sice Funkční model, Design a Stavba systému a nakonec Implementace. Tyto fáze probíhají iterativně a předchází jim Studie proveditelnosti a Obchodní průzkum. Analýza a návrh probíhají pomocí základní techniky prototypování. Tato metodika vznikla v 90. letech ve Velké Británii a patří mezi praktiky rychlého vývoje aplikací. [5]

3. 6.3 Adaptive Software Development

Jedná se o filosofický základ agilního přístupu vývoje softwaru. Říká, že je nutné se změnám přizpůsobit (adaptovat se), ne se jim bránit. Vychází z dynamického životního cyklu *Spekulace – Spolupráce – Učení* (obrázek 8). [5]

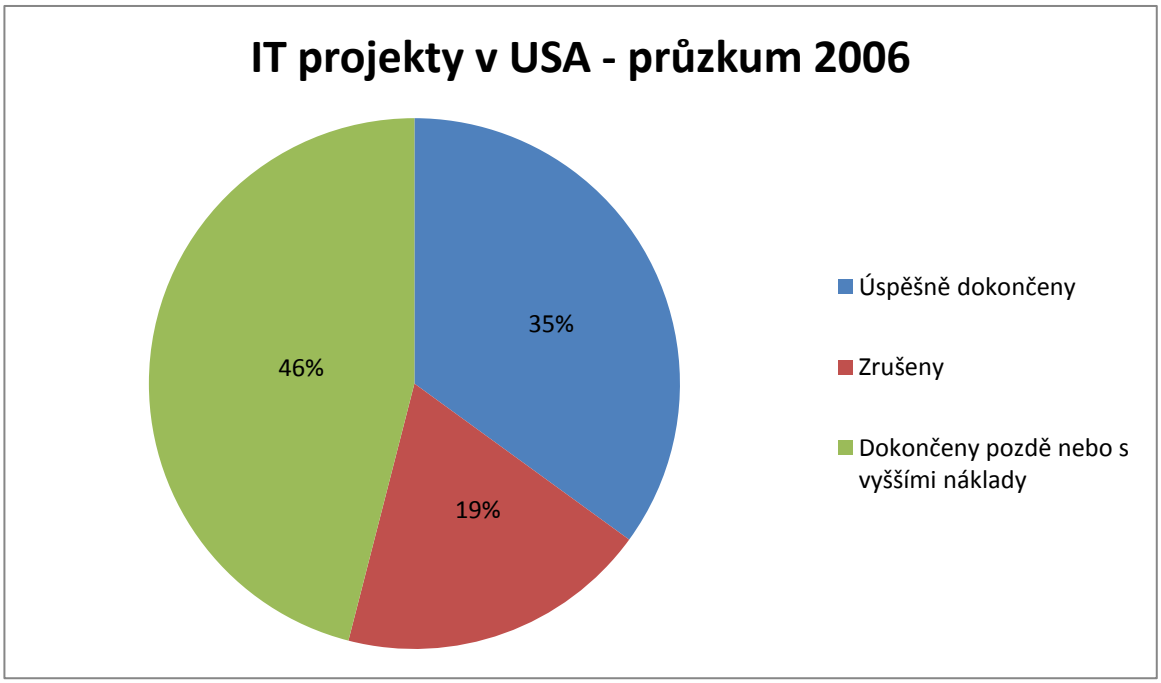
Obrázek 8 - Dynamický životní cyklus adaptivní metodiky (zdroj: autor - upraveno dle [1])

4. Agilní projektový management

Agilní projektový management patří mezi nové obory projektového managementu. Nejvíce se používá v oblasti vývoje softwaru, a to jak ve světě, tak v ČR. Principem je iterační řízení, vše se projektuje po dílčích celcích. Tím se stává celý projekt efektivnějším, jelikož lze včas odhalit chyby a nesrovnalosti a reagovat na ně. Nejznámější metodikou využívanou v agilním projektovém řízení je metodika SCRUM, která bude popsána v další části práce. [36]

4.1 Úspěšnost projektu

Výsledky průzkumu z roku 2006 na grafu 3 dle [12] zobrazují, že celých 19 % projektů zcela selže a u 46 % jsou překročeny náklady nebo čas. Pouze 35 % projektů je úspěšných.



Graf 3 - Průzkum průběhu IT projektů v USA v roce 2006 (zdroj: autor – upraveno dle [12])

Pokud je tradiční projektový management neefektivní, je třeba hledat nové přístupy. Požadavky se totiž mohou měnit a při tradičním přístupu v takovémto případě, lze pouze projekt zrušit a začít znovu. Při agilním projektovém managementu vývojáři a účastníci projektu pracují společně na jednom místě. Tím se programátoři vyvarují nesprávnému pochopení požadavků a zákazníci mohou měnit požadavky přímo ve spolupráci

s vývojovým týmem. Od zákazníků je po každé iteraci získávána zpětná vazba, která má za následek absenci chyb a správnost funkcí. [12]

4.2 Tým a charakteristika agilního přístupu

Tým se většinou skládá z vývojářů, kteří píší kód v párech, z důvodu kontroly kvality. Nachází se zde také koncový uživatel (zákazník), IT architekt, analytik a projektový manažer. Ke všemu se váže minimální dokumentace, protože požadavky a omezení vycházejí ze vzájemné komunikace. Systém se zákazníkovi předkládá postupně po částech, od funkcí s nejvyšší prioritou. [12]

Základem agilního projektového managementu je několik prvků. Jedná se o [12]:

- Vizualizaci,
- umístění týmu na jedno místo,
- testování vývoje,
- adaptivní řízení,
- spolupráci na vývoji,
- správnou identifikaci vlastností,
- upřednostňování vedení a spolupráce před velením a řízením,
- změnu zaměření z nákladů na příjmy,
- poučení, co příště udělat lépe.

Při vizualizaci probíhá např. rozlišení funkcí, které již byly navrhnuty a testovány. Na stěnu se tyto funkce přiřadí pod jednu barvu a funkce, které byly navrhnuty, avšak nebyly zatím testovány, se přidělí k jiné barvě. Uživatelé hned na první pohled vidí, v jakém stavu se v této chvíli projekt nachází. [12]

Každý tým, který přijme agilní metodiky, zajistí rychlejší dodání produktu a vyšší kvalitu, která odpovídá přesným požadavkům zákazníka. Fáze průběhu projektu v agilním přístupu jsou [22]:

1) Nultá iterace,
2) analýza změny,
3) implementace požadované vlastnosti,
4) předvedení klientovi,
5) opět na bod 2), pokud není projekt hotov,
6) údržba a rozvoj, pokud je projekt hotov.

V prvním bodu jde o to, analyzovat počáteční stav a naprogramovat základní aplikaci, která již může být představena zákazníkovi. Ve druhé fázi zákazník poskytne projektovému týmu priority na výsledný systém a projektový tým provede rozbor vývoje. Stanovuje se, které práce mají přednost a které se mohou naopak vyvíjet až později. K získání správného výsledku se využívá funkce prototypování. Vytvoří a upravuje se takový model, který odolává změnám. V této fázi také probíhá tvorba datových modelů, práce s datovými modely a komplexně s logikou celého projektu. Datové modely jsou upravovány, přidávají se nová data a často probíhá i mazání modelů, které nevyhovují nebo vykazují chyby. Existují zde pomocné knihovny, které jsou rovněž upravovány dle přesných požadavků zákazníka. [22]

Následuje fáze implementace, kdy každý účastník týmu vykonává zadané úkoly. Důležitá je zde role manažera, který provádí kontrolu úkolů, s členy týmu řeší změny v dalších iteracích a lze tak konstatovat, že eliminuje všechny problémy, které by mohly bránit vývojářům při práci. Nejedná se však o nadřazenou roli, všichni v týmu, včetně manažera, jsou si rovni, protože se snaží dosáhnout společného cíle, a sice splnit podmínky kladené zákazníkem v dohodnutém čase. [22]

Předání zákazníkovi je již poslední fází celého projektu. Celý tým se řídí dle pravidla, že zákazníkovi se ukazuje pouze dokončená práce. To co není ještě dokončené, je třeba skrýt. Důležité je, aby zákazník shledal změny, které nastaly v již známých aplikacích. Zákazníka je také nutné informovat o nedokončených funkcích a dohodnout se s ním na dalším postupu a vybrat následné implementace. [22]

5. SCRUM

Jde o nejznámější metodiku agilního programování. Vznikla na počátku devadesátých let dvacátého století a postupně se rozšířila do celého světa. Pomyslnými „otci“ této metodiky jsou *Ken Schwaber* a *Jeff Sutherland*. Název vznikl z anglického slova scrum, které je využíváno v ragby. Toto slovo znamená skrumáž, nebo mlýn a je vysvětleno jako soustředění několika hráčů na jednom místě tak, aby dotlačili míč společně na požadovanou pozici. Vývojový tým si také klade za cíl „dotlačit míč“ na požadovanou pozici, a sice dokončit produkt tak, jak vyžaduje zákazník. Místo poskytnutí kompletního a detailního popisu jak projekt vytvořit je vše ponecháno na vývojovém týmu, který dělá to nejlepší, aby byl celý projekt správně vytvořen. Postupuje se pomocí iterací, které jsou nazývány *sprint.* Tyto iterace trvají dva až čtyři týdny. SCRUM se hodí především pro projekty vývoje softwaru, pro projekty, kde jsou naléhavé a často se měnící požadavky od zákazníka na činnosti výsledného systému. Všechny pojmy, které jsou využívány v metodice SCRUM, se nepřekládají do českého jazyka, používají se pojmy jako Product Backlog, Sprint Backlog, Sprint Review, ad. [9][21][23]

5.1 Role

Role účastníků ve SCRUMu se rozděluje na dvě skupiny: Pigs (prasata) a Chickens (kuřata). Toto rozdělení vzniklo díky vtipu o praseti a kuřeti. Kuře a prase si chtějí otevřít restauraci. Kuře ji chce pojmenovat „Ham and Eggs“. Prase vkládá maso, čímž je přímo „zapojeno“, zatímco kuřete se problém „pouze týká“, jelikož vkládá vejce. Ve SCRUMu je tedy rozdělení do skupin Pigs a Chickens takové, že skupina Pigs zastupuje osoby, které přímo souvisí s vývojem a do skupiny Chickens patří osoby, které se přímo vývoje neúčastní, což jsou manažeři a uživatelé produktu. Podrobněji jsou role vysvětleny níže. [21]

Dělení dle [21] je následující. Do skupiny Pigs tedy patří dle terminologie metodiky SCRUM:

- Product Owner,
- ScrumMaster,
- Team.

A do skupiny Chickens jsou zařazeni:

- Managers,
- Stakeholders.

Product Owner zastupuje zájmy všech zúčastněných, zodpovídá za priority, určuje, co bude implementováno v příštím sprintu a také určuje detaily implementace. Sleduje požadavky zákazníka, návratnost investic a spuštění plánů. List požadavků je nazýván Product Backlog a Product Owner hlídá jeho dodržování, především to, aby byla hlavní funkcionalita vyvinuta včas. [21]

Hlavní funkcí *ScrumMastera* je odstínit tým programátorů od okolního světa. Řeší spory, řídí vývojáře, ale také zajišťuje, aby týmu fungovaly počítače, aby měli správný software. Nejvhodnější pro tuto roli se jeví Project Manager (Projektový manažer). Nesmí však být zároveň programátorem, neboť by nemohl ostatní programátory odfiltrovávat od rušivých vlivů, protože by byl sám zranitelný. Proto autor této metodiky zvolil název ScrumMaster, aby nebyla tato funkce přímo spojována s tradičním Projektovým manažerem. ScrumMaster je pouze zprostředkovatel, který nedostává žádná ocenění za splněné projekty, avšak je velice důležitým mezičlánkem mezi uživatelem Product Owner a týmem. [18][21]

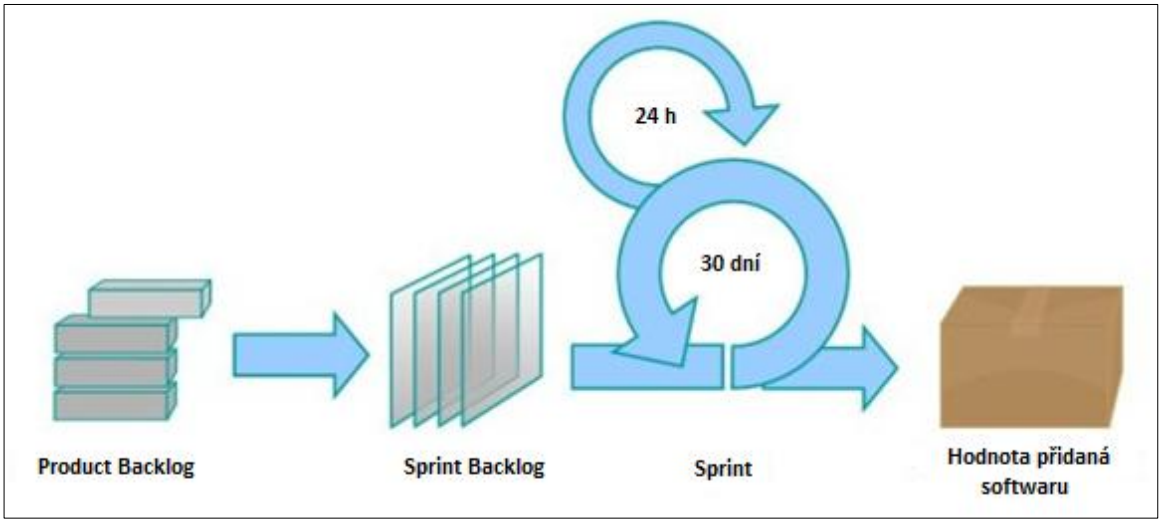
Team (tým) pracuje na rozvoji požadovaných funkcí. Členové týmu jsou kolektivně zodpovědní za úspěch každé části projektu a také za úspěch projektu jako celku. Jedná se o skupinu asi 3 – 6 osob. Členové jsou programátoři, osoby zodpovědné za řízení kvality, dokumentátoři. Pokud je vývojářů více, vzniká ve SCRUMu více týmů, přičemž dělení může být takové, že jeden tým je zodpovědný za jednu skupinu funkcí a druhý za druhou skupinu, nebo může být systém rozdělen na vrstvy a každému týmu může být přidělena jedna vrstva. [21]

Managers (manažeři) pomáhají nastavit prostředí v okolí vývoje. Nejedná se ani o vývojáře, ani Product Owner či ScrumMastery. Pokud se hovoří o *Stakeholders*, jedná se o tým od zákazníka, který testuje funkce a vznáší připomínky zvenčí, během vývoje produktu. Přítomnost zákazníka je velice důležitá, může se účastnit diskuzí, zodpovídá otázky a tím se podílí na vývoji skutečně užitečného produktu. [21]

5.2 Hlavní činnosti v průběhu SCRUMu

Základním prvkem v projektování pomocí SCRUM je vyvíjený produkt. Od týmu se očekává, že na konci každého sprintu předvede část produktu nebo stav systému právě k určenému dni. Dalším důležitým nástrojem zde je Product Backlog (nevyřízený), což je seznam funkcí, které mají být přidány do vyvíjeného systému. Správcem tohoto seznamu je *Product Owner*, který sleduje, zda tým pracuje na funkci s nejvyšší prioritou. Činnosti Product Ownera jsou vysvětleny v další části práce. Nejoblíbenější a také nejcennější

z pohledu funkčnosti je vytvoření Product Backlogu přímo dle krátkého popisu funkcí od uživatele či zákazníka. Na obrázku 9 je zobrazen celý průběh metodiky SCRUM. Na sprint navazuje vznik nové funkcionality, která je dále implementována v rámci sprintu. [23]



Obrázek 9 - Průběh vývoje projektu za použití metodiky SCRUM (zdroj: autor – upraveno dle [27])

Hlavní činností v této metodice je *sprint*. SCRUM patří mezi iterativní procesy, projekt je tak rozdělen na několik po sobě jdoucích sprintů. Všechny sprinty jsou časově omezeny, většinou trvají od dvou týdnů do jednoho měsíce. Dle [23] výzkumy dokazují, že nejbějžnější doba trvání jednoho sprintu je dva týdny. Během této doby je možné, aby tým vyvinul část vlastností, které lze dále rozvíjet a testovat. [23]

Každý sprint se skládá ze *Sprint Planning Meeting* (plánovaná setkání týmu), kde je vytvořen *Sprint Backlog*. Jedná se o seznam úkolů, které musejí být provedeny během sprintu. Během těchto setkání probíhá komunikace týmu a Product Ownera. Je rozebrán Product Backlog a jsou vybrány prvky s nejvyšší prioritou, se kterými se bude pracovat v následujícím sprintu. Každý člen týmu se rozhodne, kolik položek je schopen zvládnout a k tomuto číslu se zavazuje. Následně je vytvořen již zmíněný Sprint Backlog. Každý den probíhají setkání členů týmů, kterým se říká Daily Scrum Meeting (každodenní setkání scrumu). Těchto setkání se účastní Product Owner i ScrumMaster. Délka trvání těchto setkání nepřevyšuje patnáct minut. Členové týmu zde diskutují, co vytvořili předchozí den, co bude vytvořeno dnes a jsou tak identifikovány možné překážky dalšího vývoje. Každý programátor tak může sdělit ostatním z týmu, že se například setkal s chybou, a proto se jeho úkol zdržel, nebo naopak vývoj šel rychleji, než očekával, proto může pracovat na dalších svých úkolech. Tato setkání pomáhají týmu při synchronizaci a vzájemné komunikaci. [23]

Na konci každého sprintu probíhá *Sprint Review* (revize). Tým představuje vlastnosti, které byly přidány systému během sprintu, který právě proběhl. Cílem těchto setkání je zpětná vazba od Product Ownera nebo uživatelům či zákazníkům, kteří byli pozváni k Sprint Review. Z tohoto setkání mohou vzejít změny na základě nově objevených skutečností. Ovšem také to může vést k nalezení chyb a k přezkoumání a lze také dopisovat další funkce do Product Backlogu. Na konci každého sprintu také probíhá činnost, která je nazývána *Sprint Retrospective* (pohled zpět). Kromě týmu se účastní také ScrumMaster a Product Owner. Jedná se o setkání, kde probíhá zhodnocení celého minulého sprintu a diskutuje se nad otázkami, co by šlo a bylo dobré zlepšit v následujícím sprintu. [23]

5.3 Workflow (pracovní postup) projektu

Než je zahájen nový projekt, je nutné specifikovat očekávaný výsledek. Definují se vlastnosti a funkce, které má konečný systém obsahovat, z těchto funkcí a vlastností se sepisují tzv. User Stories (případy užití). Celá tato fáze se nazývá Release Planning. Ukázka User Story je zobrazen v tabulce 1. Všechny User Stories se vypíší do Product Backlogu a Product Owner seřadí všechny tyto případy užití dle důležitosti od nejdůležitější po méně důležité. Vznikne tím prioritní řazení, dle kterého jsou případy dále vybírány do jednotlivých sprintů a jsou dle tohoto rozdělení také definovány úkoly. Každému úkolu je přiřazena časová náročnost, která se většinou udává v bodech, jelikož na počátku projektu je rychlost týmu neznámou hodnotou. Proto programátoři během přibližně prvních dvou sprintů odhadují počet úkolů, které mohou stihnout právě v jednom sprintu. [21]

Tabulka 1 - User Story uživatele User (zdroj: autor)

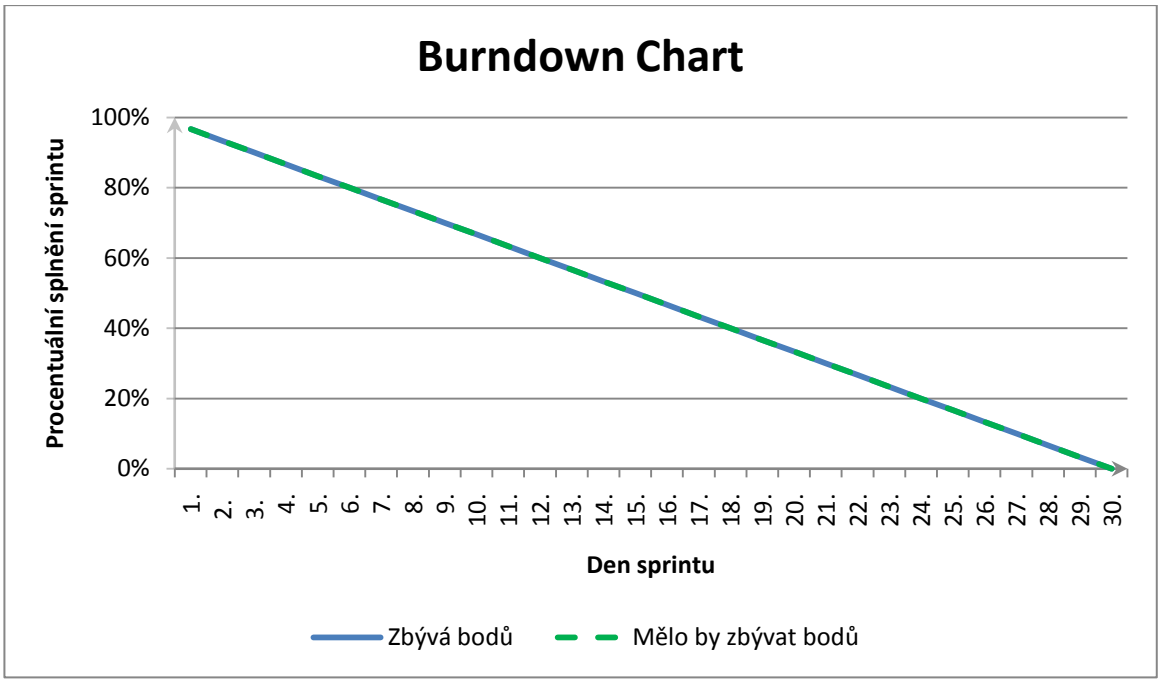
Jako uživatel User
– Mohu zobrazit své úkoly.
– Mohu editovat úkoly.
– Mohu zobrazit své projekty.

Termín dokončení projektu je nazýván jako *Time Box*. Do konce Time Boxu musí proběhnout implementace všech funkcí, který mají být vyvinuty. U agilního přístupu je datum dokončení stanoveno tak, že celý vývoj je dokončen přesně v tento den, v úvahu nepřicházejí žádné rezervy ani možnost rychlejšího provedení práce. Pokud jsou dle priorit vybrány User Stories z Product Backlogu, jsou přesunuty do Sprint Backlogu. Pokud chce zákazník provést

nějakou změnu, může tak učinit pouze mezi iteracemi, tedy sprinty. Pokud je již sprint rozjetý, nelze za žádnou cenu měnit zadání. Toto je klíčová podmínka SCRUMu a pomocí tohoto se programátor soustředí zcela a jedině na dosažení cíle a dodržení termínu. [21]

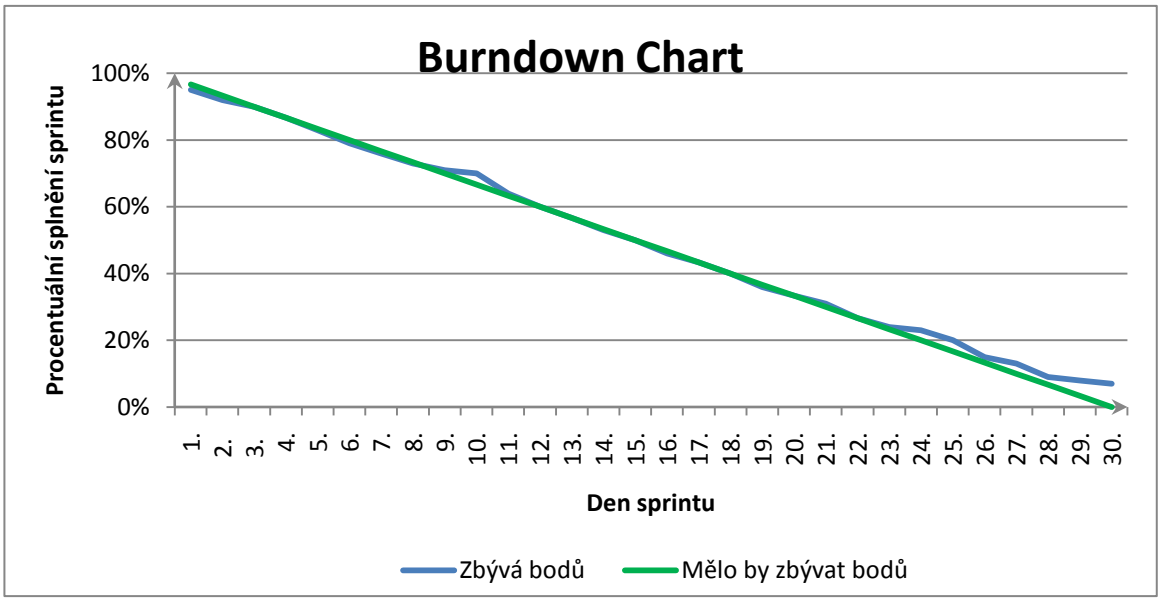
V praxi se lze setkat s tím, že programátor potřebuje pracovat i na jiných projektech, proto je možné rozdělit pracovní dobu mezi projekty. Například pět hodin denně lze věnovat práci na sprintu a tři hodiny na jiných projektech. Je důležité, aby nebylo zasahováno do času, který má programátor k vývoji vyhrazen, protože pak je možné přesně odhadnout čas, za který vývojář vykoná určitý úkol i rychlost vývoje. Z tohoto lze pak soudit, zda úkoly stihne dokončit do konce sprintu. V agilním přístupu je nemožné pracovat přesčas, jak již bylo uvedeno v kapitole 3.3.2. Při denních setkáních Daily Scrum se setkává tým se ScrumMasterem a řeší, co bylo uděláno včera, co bude vytvořeno dnes a kam se bude tým ubírat v dalším dnu. Těchto setkání se smí zúčastnit i více lidí, např. Product Owner, avšak nesmí do diskuze zasahovat. Dle průběhu sprintu je vytvářen *Burndown Chart* (graf zobrazující, kolik ještě zbývá práce). [21]

Na grafu 4 je zobrazen průběh sprintu. Graf zobrazuje, kolik zbývá ještě splnit, aby byl celý sprint dokončen. Na ose *x* jsou zobrazeny dny sprintu a na ose *y* je zaevidováno kolik zbývá procent k dokončení sprintu. Tento graf je ukázkou přesného dodržení zadání. Křivka „kolik zbývá bodů“ splývá s křivkou „kolik by mělo zbývat bodů“. Tým pracuje tak, jak se od něj očekává. Třicátý den sprintu zbývá 0% projektu, projekt byl splněn. Na následujících grafech jsou hodnoty na ose *y* v procentech. Avšak dle [31] je dobré ohodnotit průběh na ose *y* v jednotkách man - days. Tyto jednotky zastupují pracovní den na jednoho pracovníka. Vznikly, jelikož nelze přesně odhadnout náročnost jednotlivých úloh.



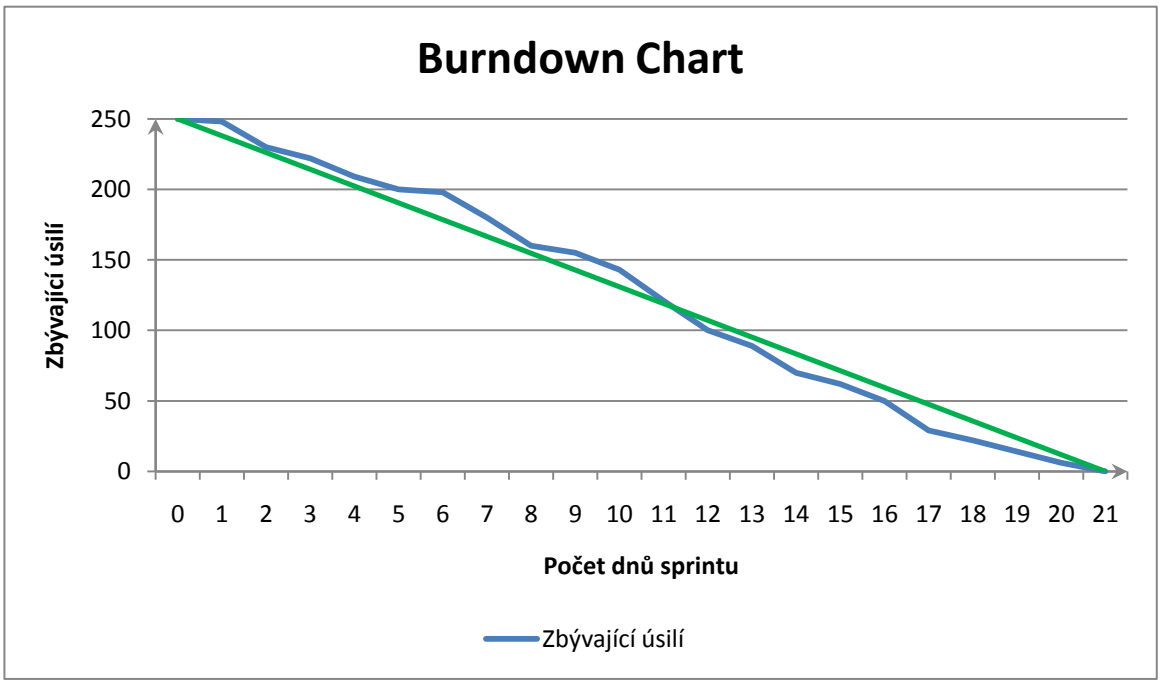
Graf 4 - Burndown Chart s přesným dodržením zadání (zdroj: autor)

Graf s přesným dodržením zadání však není obvyklý, protože úkoly jsou individuální, některé jsou snazší, jiné vyžadují k dořešení více času. Na grafu 5 je zobrazen průběh sprintu, kdy ke konci sprintu dochází ke zpomalení a tým tedy nedokončí práci včas. Křivka zbývajícího počtu bodů se k třicátému dni zastavila nad hranicí nuly, projekt tedy není dokončen.



Graf 5 - Burndown Chart s koncem sprintu nad 0% (zdroj: autor)

Tato situace ovšem také nenastává z důvodu přítomnosti ScrumMastera. Ten spočte, jaká je rychlost týmu a pokud hrozí, že by nebyly některé úkoly splněny, vrátí tyto úkoly ze Sprint Backlogu do Product Backlogu. Tím jsou na konci sprintu hotovy všechny úkoly ze Sprint Backlogu. Musí ovšem řešit důvody, proč tým pracuje pomaleji a proč nebyla dodržena stanovená rychlost. Pokud na druhou stranu vyvíjí rychleji, úkoly jsou přidány. Tzn. že ScrumMaster přesune dle zvážení času úkoly z Product Backlogu do Sprint Backlogu. Většinou vychází Burndown Chart podobný jako na grafu 6. [21]



Graf 6 - Příklad možného průběhu Burndown Chart (zdroj: autor)

6. Vývoj aplikace SCRUM Board

Součástí práce je vytvoření názorné aplikace SCRUM Board, která demonstruje použití metodiky SCRUM. Jedná se o nástroj, který zobrazuje využití metodiky na reálném projektu, a to právě vytvoření elektronické formy tabule SCRUM Board. Vývoj probíhal přesně dle metodiky SCRUM, která je do detailů popsána v kapitole 5, pouze s tím rozdílem, že autor zastupoval role všech zúčastněných. Z důvodu názorného zobrazení a lepší představivosti byli nadefinováni fiktivní uživatelé.

6.1 SCRUM Board

Pokud týmy pracují v jedné místnosti, většinou využívají prostor tabule, která je speciálně vytvořená pro sledování činností během jednoho sprintu. Po tomto sprintu se celá tabule vyčistí a započne se sprint následující. Jednotlivé úkoly jsou přitom zaznamenávány na lepící papírové štítky. Jedná se o neefektivní, avšak jednoduchý způsob zobrazení. Existují však týmy, které nesdílejí jednu místnost, či budovu, někdy se stává, že týmy nesdílejí ani jeden kontinent, proto je nutné, aby zobrazení průběhu bylo ve formě elektronické, aby všichni zúčastnění na první pohled viděli průběh sprintu, v jaké se nachází části a zda se vyskytli nějaké problémy či chyby. Forma elektronické SCRUM Board je často využívána také z důvodu lepší efektivity, zálohování i uživatelské „příjemnosti“. Uživatelé se mohou zpětně podívat, jak řešili např. podobný problém v minulosti, v jaké čase a s jakými úspěchy.

V tabulce 2 jsou zobrazeny jednotlivé části tabule. Do prvního sloupce jsou zapisovány jednotlivé Story od zákazníka, které byly vybrány dle priorit právě pro tento sprint. Následuje sloupec, kde jsou vypsány úkoly, které je nutné splnit v rámci jednotlivých stories. Ve sloupci In Process jsou zobrazeny úkoly, na kterých tým právě pracuje a ve sloupci To Verify jsou úkoly, které jsou již přichystány ke kontrole. Posledním sloupcem je sloupec Done, kde jsou již hotové a zkontrolované úkoly.

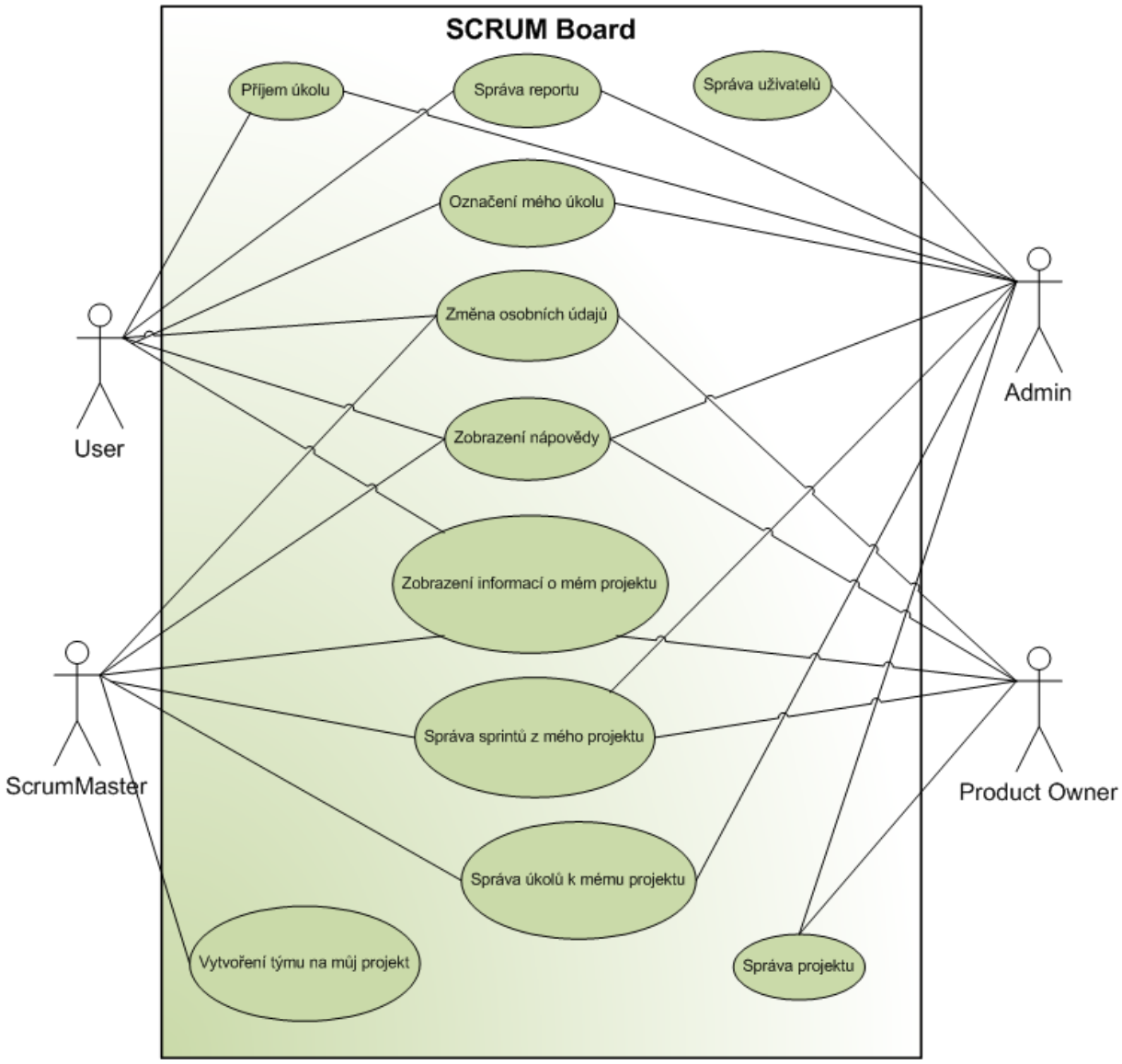
Tabulka 2 - Zobrazení SCRUM Board (zdroj: autor)

Story	To Do	In Process	To Verify	Done

6.2 Požadavky na aplikaci

Na obrázku 10 je zobrazen Use Case diagram aplikace SCRUM Board. Cílem práce není zobrazení datových modelů, a jelikož by vzniklo spoustu modelů, je zobrazen pouze jeden diagram, který nese právě požadavky na celý systém. Ostatní modely (konceptuální a logický) byly definovány a bylo dle nich postupováno při tvorbě modelu fyzického. Tabulky, které vznikly, se nachází v kapitole 6.4 této práce. Jelikož diagram případu užití má být především co nejvíce přehledný a jednoduchý, bylo využito případu užití „spravovat“, který zastupuje komplexní správu (uživatelů, projektů, sprintů, úkolů a reportů). Případy užití, které zobrazují správu, jsou označeny dle [9] zkratkou CRUD. Tato zkratka pochází z počátečních písmen anglických slov Create (vytvoření), Read (čtení nebo zobrazení), Update (aktualizace nebo změna) a Delete (smazání). V programování se tato zkratka často používá právě pro správu zákazníků, dokumentů apod. „Správa“ je pak ve scénářích rozdělena tak, že hlavní scénář zastupuje „správu“ a alternativní scénáře pak „vytvoření“, „čtení“, „aktualizaci“ a „smazání“. Každému případu užití předchází povinnost přihlášení uživatele.

Diagram případu užití zobrazuje vyvíjenou aplikaci z hlediska požadavků, které byly definovány pro správnou funkčnost. Pro samotnou aplikaci jsou brány v úvahu čtyři druhy uživatelů. Do skupiny User patří programátoři, databázový specialisté, analytici, testeři ad., kteří se přímo podílejí na tvorbě aplikace. Tito uživatelé si mohou zobrazit své projekty, své sprinty a vybrat úkol, na kterém budou pracovat, tento úkol pak označit jako rozpracovaný nebo hotový. Jejich úkolem je správa reportů, které jsou vázány na daný úkol. Dalším uživatelem je ScrumMaster. Jeho úlohou je kontrolovat, spravovat a starat se o Team (Usery). Definuje sprinty a úkoly, které vytváří, čte, aktualizuje i maže. Avšak nesmí zasahovat do zobrazování hotových/rozpracovaných úkolů a také nesmí na úkolech pracovat. ScrumMaster nezasahuje do správy projektů, může je zobrazit, avšak nijak je neupravuje, nevytváří, ani nemaže, pouze může do projektu přidat či odebrat uživatele s rolí User.



Obrázek 10 - Use Case zobrazující případy užití jednotlivých uživatelů (zdroj: autor)

Další uživatel je Product Owner, ten vybírá Stories a poté definuje strukturu projektu. Definuje uživatele do týmu a jako jediný může vytvářet, editovat a mazat projekty. Také může spravovat sprinty a úkoly. Reporty lze u uživatele Product Owner pouze zobrazit. Také nesmí pracovat na žádném úkolu a nesmí ho označovat jako hotový/rozpracovaný. Posledním nutným uživatelem je Admin, který je správcem celé aplikace. Účty všech uživatelů jsou spravovány právě Adminem. Každý uživatel si může změnit osobní údaje, kromě práv, které může měnit právě Admin. Práva všech uživatelů má i admin a to především z důvodu aktualizací, úprav a správy celé aplikace.

Ke každému případu užití je nutné uvést scénář. Jelikož se jedná o velké množství scénářů, je uveden pro ukázku pouze jeden scénář, zbylé scénáře se nachází v příloze č. 1 práce. Jedná se o Scénář pro Případ užití „Příjem úkolu“ (tabulka 3).

Tabulka 3 - Scénář Případu užití "Přijmout úkol" (zdroj: autor)

Název Use Case		**Příjem úkolu**
Aktér		User, Admin
Popis		Uživatel si vezme úkol, na kterém bude pracovat
Vstupy		Vypsán úkol od ScrumMastera
Výstupy		Úkol přijat pod daného uživatele
Spouštěcí událost		Rozhodnutí pracovat na daném úkolu
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na název projektu, ve kterém chce přijmout úkol
2	Systém	Zobrazí sprinty, které náleží pod daný projekt
3	Aktér	Ve sloupci *Akce* klikne na *Úkoly* v řádku sprintu, kam patří úkol
4	Systém	Zobrazí úkoly k danému sprintu
5	Aktér	Klikne ve sloupci *Akce* na *Vzít úkol*, o který má zájem
6	Systém	Ve sloupci *Přiřazeno uživateli* vypíše jméno a příjmení uživatele, který právě úkol přijmul dle jeho identifikačního čísla
7	Aktér	V záhlaví aplikace klikne na *Odhlásit*
8	Systém	Odhlásí uživatele a zobrazí přihlašovací obrazovku

Vždy je nutné, aby byl každý uživatel přihlášen pod oprávněním, které mu náleží. Posléze již hierarchicky prochází celou aplikaci. Při funkci „Vzít úkol“ je nutné, aby prostoupil skrz projekty a sprinty až k tabulce úkolů.

6.3 Programy a aplikace využité při vývoji aplikace

Požadavkem pro tvorbu aplikace bylo především intuitivní ovládání celé aplikace, aby nemuseli být programátoři speciálně proškoleni pro práci s výsledným rozhraním. V úvahu přicházel skriptovací jazyk PHP nebo programovací jazyk Java EE (Enterprise Edition). Oba tyto jazyky jsou srovnány v tabulce 4. Výběr proběhl především na základě následujících otázek, které nepřímo udávají další požadavky na programy, se kterými se bude pracovat při vývoji systému.

- Bude aplikace internetová nebo si ji bude muset každý uživatel nainstalovat?
- Je nutné udržovat skripty s kódem v utajení?
- Je možné vybírat i ze zpoplatněných produktů?
- Jedná se o rozsáhlou aplikaci, nebo stačí menší přehlednější systém?
- Existuje zde nutnost hned při tvorbě kódu definovat typ proměnných?

Tabulka 4 - Srovnání skriptovacího jazyka PHP a programovacího jazyka Java (zdroj: autor dle [23])

	PHP	**Java EE**
Náročnost na pochopení	Jednoduchý	Jednoduchý
Přístup	Internetový, snadná přenositelnost	Internetový, snadná přenositelnost
Utajení skriptu	Interpretovatelný jazyk	Nejdříve kompilace, pak interpretovatelnost
Cena	Zdarma	Zdarma
Nutná odbornost	Ne	Částečně
Velikost aplikace	Malé a střední aplikace	Velké firemní aplikace
Typování	Dynamické	Statické
Webhosting	Prakticky všude	Zřídka

Jak PHP, tak Java EE patří mezi jednoduché jazyky. Pro Javu EE je ale již třeba určitá odbornost. Také není nutné nijak skrývat výsledný skript, protože funkce, které se v kódu nachází, jsou volně dostupné jak v tištěných, tak elektronických zdrojích. To je velká výhoda jazyka PHP. Požadavkem na nástroj je možnost internetového přístupu tak, aby se uživatel např. mohl připojit k síti a provést změny nebo zobrazit soubory, týkající se projektu, na kterém pracuje. Tím je zabezpečeno, že i uživatelé, kteří nesdílí se zbytkem týmu společné prostory pro práci, jsou součástí týmu a mohou udržovat se zbylými uživateli kontakt a komunikaci.

Tím se lze dobrat k další otázce a tou je cena pořízení programovacího jazyka a tím i výsledná cena. Oba jazyky, mezi kterými se rozhoduje, jsou zdarma. Na internetu sice existují další volně stažitelné programy, avšak nejsou tak jednoduché jako tyto dva uvedené a jednoduchost hraje v metodice SCRUM důležitou roli. Posledním bodem je potřebný

pronájem prostoru na serveru (webhosting). Jelikož bylo rozhodnuto, že nástroj bude dostupný na internetové síti a hosting pro PHP je nabízen prostor prakticky od všech poskytovatelů, byl vybrán právě skriptovací jazyk PHP pro vývoj dané aplikace.

Dále bylo nutné vybrat z několika druhů databází. Po důkladném prozkoumání bylo rozhodováno mezi MySQL, PostgreSQL a Oracle. V úvahu byla brána také SQLite, ale tato databáze nedisponuje takovým množstvím funkcí, neumožňuje např. pokročilejší úpravy již vytvořených tabulek apod. Proto byla z dalšího porovnání vyřazena. Důležitými faktory pro výběr databáze byly: cena, podporovaný webhosting, dokumentace a snadná instalace. Srovnání databází je zobrazeno v tabulce 5.

Tabulka 5 - Srovnání databází (zdroj: autor dle [33])

	MySQL	**PostgreSQL**	**Oracle**
Cena	Zdarma	Zdarma	Placená databáze
Podpora na webhostingu	Téměř všechny	Občas	Prakticky nikdy
Dokumentace	Snadno dostupná a rozsáhlá	Méně dostupná	Rozsáhlá
Instalace	Snadná	Náročná	Náročná

Databáze Oracle byla vyřazena z důvodu vysoké ceny. MySQL je podporována téměř všemi společnostmi, které webhosting nabízejí. Díky webhostingu lze nahrát webové stránky na internet, bez nutnosti vlastnictví vlastního serveru. PostgreSQL již není tak rozšířeny, a proto je podpora této databáze o poznání menší. Jelikož je při tvorbě aplikace SCRUM Board bráno v úvahu, že celý systém bude možné umístit na webový hosting, bylo nutné vybrat takovou databázi, která je dostatečně rozšířena a poskytována. Proto má v této vlastnosti MySQL velkou výhodu nad ostatními databázemi. Další výhodou MySQL je snadno dostupná a rozsáhlá dokumentace, kterou lze lehce získat z internetu. S touto databází pracuje velké množství uživatelů, proto lze snadno dohledat příkazy i příklady využití. Tím se velice usnadňuje práce s touto databází. PostgreSQL nemá tolik uživatelů, proto je těžší najít využívané příkazy. Instalace MySQL je o poznání snazší než instalace PostgreSQL, která se skládá z mnoha kroků. Proto byla vybrána pro tvorbu aplikace SCRUM Board databáze MySQL.

Přímo byl použit bezplatný software phpMyAdmin, který umožňuje správu MySQL přes WWW. Rozhraní obsahuje většinu funkcí, které je nutné využívat při práci s databází.

Jedná se např. o tvorbu tabulek, procházení, kopírování, přejmenování, změna či mazání databází i tabulek. Všechny tyto funkce lze vykonávat i nad poli i indexy. Důležitou vlastností tohoto softwaru je import i export do mnoha formátů. Jedním z formátů je např. *.csv*, který lze otevřít v MS Excel nebo v jiném tabulkovém procesoru. Dalším formátem je .xml. Data lze tak exportovat a dále s nimi pracovat.

Nakonec byl pro tvorbu SCRUM Board zvolen balík EasyPHP, který se skládá ze skriptovací jazyka PHP, webového serveru Apache, SQL databázového systému MySQL a nástroje phpMyAdmin. Server Apache je nejrozšířenější mezi servery, které poskytují HTTP služby v synchronizaci s platnými normami HTTP. Celý balík se nainstaluje a tím je získána možnost jednoduchého vývoje na vlastním počítači. Rozhraní je uživatelsky příjemné, intuitivní a jednoduché.

Pro samotnou tvorbu vývojového kódu bylo použito vývojové prostředí NetBeans IDE. Jedná se o nástroj, ve kterém lze psát zdrojový kód a důležitou výhodou je, že celý nástroj je zdarma ke stažení na oficiálních stránkách produktu. Dalším důležitým hlediskem, ke kterému bylo přihlédnuto při výběru vývojového prostředí, je intuitivní ovládání, prostředí má velice dobře vyvinutou nápovědu, která nabízí již při nadpisu prvního písmena nejpoužívanější příkazy z podporovaného jazyka. V neposlední řadě přispěla k výběru zkušenost autora s tímto prostředím.

6.4 Databázové tabulky

Po instalaci balíku EasyPHP je v první řadě nutné vytvořit databázi, nadefinovat jméno a přihlašovací údaje a údaje o serveru, na kterém celá databáze běží. Databáze byla nazvána *scrumboard*, jméno i heslo databáze byly nadefinovány a server byl také identifikován. Všechny tyto údaje se nachází na přiloženém CD v souboru *db_connection.php*.

Po vytvoření databáze byly vytvořeny databázové tabulky v phpMyAdminu, kde vzniklo sedm tabulek. Sloupce v tabulkách jsou sjednoceny a pojmenovány v anglickém jazyce. Počet tabulek vychází z databázového modelu, který se nachází v části 6.1 této práce. Všechny tabulky mají primární klíč identifikátor id. Cizí klíče jsou pojmenovány vždy jako fk_název. Fk zde značí počáteční písmena foreign key – cizí klíč. První tabulkou, která vznikla je tabulka *permission*. Atributy jsou id, name a code. Id může nabývat hodnot 10, 20, 30 nebo 40. K těmto hodnotám jsou přiřazeny jména (name) a kódy (code). Pro id 10 se jedná o Uživatele s kódem user, id 20 nese jméno ScrumMaster a kód scrummaster, pro id 30 je kód

admin a jedná se o Administrátora. Posledním id disponuje Produktový manažer s kódem product_owner. Identifikace tedy probíhá dle id uživatele, ten pak je dále zařazen do skupiny se shodným id. Pokud se například jedná o Projektového manažera, jeho id bude 40.

Další tabulkou je tabulka *project*. Každý projekt má jako identifikátor opět id. Dalšími atributy jsou název projektu (name), popis projektu (description) a časové určení od kdy a do kdy projekt proběhne (date_from a date_to). Třetí tabulka *report* je definována identifikátorem id a dalšími atributy. Patří sem datum, kdy byl report napsán (report_date), kolik času bylo na úkolu prozatím stráveno (work_time), komentář (comment) a také se zde nachází cizí klíče, aby bylo rozpoznáno, kdo report psal a k jakému úkolu se vztahuje, tedy atributy fk_user_id a fk_task_id.

Tabulka *sprint* obsahuje identifikátor id a sprint_order, což zobrazuje, o kolikátý v pořadí se jedná sprint v daném projektu. Aby mohl být sprint přiřazen k projektu, nachází se zde cizí klíč fk_project_id, který zařadí sprint ke správnému projektu pomocí id projektu. Dalším atributem je description - popis sprintu. V pořadí páté tabulce *task* se kromě identifikátoru id nachází atributy název (name), popis (description), průběh (duration), označení zda je projekt hotový (done) a cizí klíče – fk_user_id a fk_sprint_id. Tyto cizí klíče jsou zde z nutnosti identifikace uživatele, který na úkolu pracuje a identifikace sprintu, do kterého úkol patří. Atribut průběh zde zajišťuje možné přerušení úkolu, vykázání, kolik hodin bylo odpracováno a pokračování další den. Pokud je úkol hotov, v databázové tabulce ve sloupci done se 0 přepíše na 1. Tento atribut nabývá pouze dvou hodnot – dokončený (1) nebo nedokončený (0).

Poslední dvě tabulky jsou *user* a *user_project*. V tabulce uživatelů je kromě identifikátoru id přihlašovací jméno (login), heslo (password), jméno a příjmení uživatele (first_name a last_name) a určení oprávnění, což zde zastupuje cizí klíč z tabulky permission, kdy je uživatele přiřazeno id 10, 20, 30 nebo 40 (fk_permission_id). Poslední tabulka definuje, který uživatel (fk_user_id) pracuje na kterém projektu (fk_project_id). V obou případech se jedná o cizí klíče.

6.5 Hierarchie vývoje celé aplikace dle metodiky SCRUM

Při vývoji aplikace bylo postupováno dle metodiky SCRUM, která je popsána v kapitole 5 této práce. Jelikož je tato práce samostatným dílem, nebude zde figurovat žádný tým. Pouze bude předvedeno, jak metodika funguje v praxi, avšak vše bude řízeno autorem

práce a pouze nastíněno, jak by celý vývoj fungoval v rámci týmu. Všechny soubory, nutné k chodu celé aplikace, se nachází na přiloženém CD.

6. 5.1 Product Backlog

Na počátku je nutné nadefinovat Stories v Product Backlogu, které by nadefinoval zákazník. V této části tedy bude zapsáno, jak přesně má celá aplikace SCRUM Board fungovat, jak jsou rozděleny role a co která role zajišťuje. V tabulce 6 jsou zobrazeny User Stories pro uživatele User.

Tabulka 6 - User Stories pro aplikaci SCRUM Board (zdroj: autor)

Jako uživatel User
- Mohu zobrazit členy týmu, se kterými pracuji na projektu.
- Mohu zobrazit své projekty a informace o nich (kdo pracuje na jakém úkolu, v jaké fázi se úkol nachází, číst reporty od ostatních členů týmu).
- Mohu zobrazit sprinty.
- Mohu zobrazit své úkoly.
- Mohu si vybrat úkol ze sprintu.
- Mohu editovat úkoly (popis úkolu, délka úkolu, zapsat kolik je celkem z úkolu vykázáno, označit úkol jako hotový).
- Mohu psát report.
- Mohu zobrazit nápovědu.
- Mohu změnit své heslo pro přihlášení.

Role ScrumMastera se liší od Usera především v tom, že nemůže pracovat s úkoly, pouze je prohlížet. Zjistí tedy, jaké úkoly ještě nejsou zahájeny, pokud jsou úkoly zahájeny, tak si zobrazí, v jaké jsou fázi a pokud je úkol již hotový, vidí ho jako dokončený. Může číst reporty i komentáře, avšak nemůže si sám vzít úkol a pracovat na něm. Hlídá pouze tým a čas strávený na úkolech tak, aby sprint vyšel na stanovenou dobu. User Stories pro uživatele ScrumMaster je zobrazeno v tabulce 7.

Tabulka 7 - User Stories pro uživatele ScrumMaster (zdroj: autor)

Jako uživatel ScrumMaster
- Mohu zobrazit členy týmu, kteří pracují na úkolu. - Mohu přiřadit - Mohu zobrazit projekty a informace o nich. - Mohu vytvářet sprinty. - Mohu editovat sprinty (vytvářet popis, pořadí a předpokládanou dobu trvání). - Mohu mazat sprinty. - Mohu editovat úkoly (název, popis a předpokládanou délku úkolu). - Mohu zobrazit informace o úkolu (který uživatel na něm pracuje, kolik je celkem vykázáno, zda je úkol již hotový). - Mohu mazat úkoly. - Mohu zobrazit reporty napsané k úkolům. - Mohu přidat nový úkol. - Mohu zobrazit nápovědu. - Mohu odebírat a přidávat uživatele do týmu. - Mohu změnit své heslo pro přihlášení.

Na rozdíl od ScrumMastera, Product Owner nemůže editovat úkoly a sprinty. Vytváří a popisuje projekty z Product Backlogu. Definuje požadavky na projekt a kontroluje průběh plnění úkolů a celého projektu. Zajišťuje komunikaci mezi zákazníkem a týmem tak, aby požadavky zákazníka byly týmu předloženy v co nejsrozumitelnější podobě. Nejlepší je v tomto případě osobní setkání zákazníka a jeho podílení na projektu. Další možnosti uživatele Produktový manažer jsou zobrazeny v tabulce 8. Důležitým pravidlem je, že Product Owner nijak nezasahuje do plnění úkolů. Pouze se orientuje v průběhu projektu, aby bylo vše splněno z časového hlediska. Vybírá z programátorů tým, který bude přidělen na projekt i ScrumMastera.

Tabulka 8 - User Stories pro uživatele Product Owner (zdroj: autor)

Jako uživatel Product Owner
- Mohu vytvářet projekty. - Mohu editovat projekty (název, popis, zadáno od a nutno dokončit do). - Mohu zobrazit projekty a informace o nich. - Mohu zobrazit sprinty a informace o nich (pořadí sprintu, popis, dobu trvání). - Mohu zobrazit úkoly a informace o úkolech (který uživatel má jaký úkol, kolik je vykázáno, kolik úkolů je hotových, atd.) - Mohu zobrazit reporty.

Nejvíce práv má administrátor. Ten může využívat všechny funkce, které aplikace nabízí. Vytváří uživatele a přiděluje jim práva. Může vytvářet, zobrazovat, editovat i mazat jak projekty a sprinty, tak i úkoly a reporty.

Po ukončení každého sprintu je část projektu předvedena zákazníkovi. Pokud je zákazník spokojen, pokračuje se v pořadí následujícím sprintem. Může se stát, že má zákazník nějaké připomínky, proto se úpravy navrhnou do následujícího sprintu a případné chyby nebo nesrovnalosti jsou opraveny. Celý projekt je ukončen po posledním sprintu, který zákazník odsouhlasí. Následuje další projekt.

Vstupním bodem do aplikace je *index.php*. Tato stránka slouží ke generování všech ostatních stránek. Obsahuje hlavičku, patičku a poté funkci, která vkládá ostatní stránky, které jsou tělem. Důležitou stránkou je *login.php*, která zobrazuje přihlašovací obrazovku pro vepsání přihlašovacího jména a hesla.

6. 5.2 Uživatelé, Sprinty, Reporty

Pro správu uživatelů vznikly čtyři soubory, a sice *administration.php* (list s výpisem uživatelů), *user_edit* (editace), *user_save* (uložení) a *user_delete* (smazání). List s výpisem uživatelů zobrazí všechny uživatele, kteří se nachází v databázi. Pokud zde není žádný uživatel, systém vypíše hlášení „Žádní uživatelé“. Pod tímto souborem se nachází základní obrazovka s uživateli, odtud se pak dále administrátor pohybuje při novém vytvoření, změně či smazání. Soubor s editací nejprve rozlišuje, zda se jedná o nové vytvoření uživatele (není zde žádné id), nebo pouze změnu (systém pracuje s id), posléze vytváří nebo mění údaje v databázi. Data odeslána do databáze pomocí uložení či aktualizace. Soubor smazání odstraní

uživatele. Ty samé postupy byly dodrženy při výpisu, vytvoření, editaci a smazání sprintů a reportů. Pouze vznikly soubory s počátečním názvem *sprint* a *report*, u administrace pak *sprint_list* a *report_list*.

6. 5.3 Projekty

U projektů je technologie stejná, opět vznikly soubory *main.php* (list s výpisem projektů), *project_edit*, *project_save* a *project_delete*. Avšak navíc byly vytvořeny soubory pro přiřazení uživatelů na projekt, tedy vytvoření týmu. Jedná se o soubory *project_user_list* (výpis uživatelů – týmu, který byl přiřazen na projekt), *project_user_add* (výběr uživatelů do týmu), *project_user_add_save* (uložení vybraného uživatele do týmu) a *project_user_delete* (smazání uživatele z týmu). Výběr uživatele do týmu probíhá pomocí Select Boxu z množiny uživatelů. Výběr uživatelů do týmu projektu zobrazuje obrázek 11.

Obrázek 11 - Výběr uživatelů do týmu (zdroj: autor)

6. 5.4 Úkoly

U úkolů existují opět soubory pro zobrazení listu s úkoly, editaci, uložení či smazání. Jejich názvy jsou ve stejném formátu, pouze s uvedením *task* místo např. *user*. Existují zde však ještě dva soubory navíc, a to *task_get* a *task_done*. Tyto soubory vznikly z toho důvodu, aby mohli uživatelé s oprávněním User (samozřejmě i admin) vybírat z úkolů a přijmout úkol jako svoji práci a posléze označit úkol jako hotový, nebo rozpracovaný. Uživatel tak na listu úkolů v Akci vidí příkazy „Vzít úkol“ a „Označit úkol jako hotový/dokončený“. Pokud uživatel klikne a vezme si úkol, systém zaeviduje id uživatele do databáze pod daný úkol a poté je již úkol ve výpisu zobrazován se jménem uživatele, který na něm pracuje.

6.6 Průběh tvorby projektu v aplikaci

V první řadě je nutné, aby administrátor nadefinoval uživatele, kteří se nacházejí v organizaci. U každého uživatele určí přihlašovací jméno a prozatímní heslo, jméno a příjmení. Nutné je také přidělit roli, zda se jedná o dalšího administrátora, Produktového manažera, ScrumMastera či Usera. Poté se již mohou uživatelé přihlašovat pod přiděleným přihlašovacím jménem a heslem a své údaje si změnit.

6. 6.1 Průchod celým projektem

Produktový manažer již může vytvářet projekty a definovat Product Backlog dle požadavků zákazníka. Je nutné, aby byl projekt srozumitelně popsán, např. podobně, jak je uvedeno v tabulkách 6, 7 a 8. Nutností je zadat od kdy je projekt spuštěn a především, do kdy je třeba projekt ukončit. Product Owner také vybere složení týmu, který bude na projektu pracovat. Přičemž bere v úvahu, že je nutné rozložit do týmu uživatele tak, aby bylo vyhověno všem funkcím, které je nutné vytvořit. To znamená, že např. vybere tři programátory, databázového specialistu, analytika a testera. Na tým také přidělí ScrumMastera. Posléze může projekt upravovat i smazat.

Poté již ScrumMaster společně s týmem vytvoří Sprint Backlog. Projekt rozdělí na sprinty a tyto sprinty ScrumMaster vytvoří v aplikaci. Pro každý sprint je nutné nadefinovat popis, pořadí a především souhrn úkolů, které sprint obsahuje. Pořadí sprintu je důležité pro zobrazení návaznosti jednotlivých sprintů. Důležitou funkcí je zde právě úprava sprintů. Pokud programátoři neodhadnou dobu nutnou pro zpracování úkolů, je možné, že některý úkol by nebyl splněn do konce sprintu. Proto ScrumMaster úkol odebere a vrátí ho do Sprint Backlogu, kde bude úkol zařazen do následujícího sprintu. Poté ScrumMaster vyhodnotí, proč nebyl úkol splněn v zadaném termínu a přijme příslušná opatření.

Tým společně se ScrumMasterem nadefinuje pro daný sprint úkoly. ScrumMaster poté tyto úkoly zadá do aplikace a uživatelé (Users) si mohou tyto úkoly rozebrat a pracovat na nich. Samotná práce na úkolech zahrnuje dokumentaci k problému. Pokud je úkol započat, ale není ukončen, je nutné vykázat, kolik hodin práce prozatím zabral a zapsat co přesně bylo provedeno do komentáře (Report). Toto je velice důležité pro případ nemoci uživatele nebo změny uživatele. Do Reportu je nutné zapsat, co přesně bylo na úkolu již provedeno a je možné poznamenat i to, co k dokončení úkolu chybí.

6. 6.2 Ukázkový příklad

V aplikaci byl vytvořen projekt vývoje aplikace SCRUM Board. Celá struktura byla nedefinována autorem této práce i celé aplikace. Čtyři uživatelé přiděleni na projekt, kteří představují tým, jsou uvedeni pouze z důvodu názorné ukázky a jejich jména jsou smyšlená. Vše, co bylo vytvořeno, je dílem autora práce. Projekt s názvem SCRUM Board byl rozdělen do čtyř sprintů. Sprinty uživatel zobrazí tak, že klikne na odkaz s názvem projektu v prvním sloupci tabulky projektů.

Všechny sprinty mají délku trvání 15 dnů. Každá společnost si tuto délku upravuje dle firemní kultury nebo momentálních požadavků. Jedná se o zcela individuální údaj, který je nutné uzpůsobit přímo na tým a určitý projekt, jen tak bude SCRUM efektivní. Sprinty nelze sjednotit hned v prvním úseku, je nutné tuto délku zpočátku měnit a zjistit tak nejlepší délku sprintu v daném prostředí. Po určitém čase je již doba stanovena na optimální úrovni a společnost tak může tvrdit, že má nastavenu délku sprintu např. na 15 dní nebo 30 dní apod. První sprint s názvem Definice Product Backlogu je rozdělen na čtyři následující úkoly.

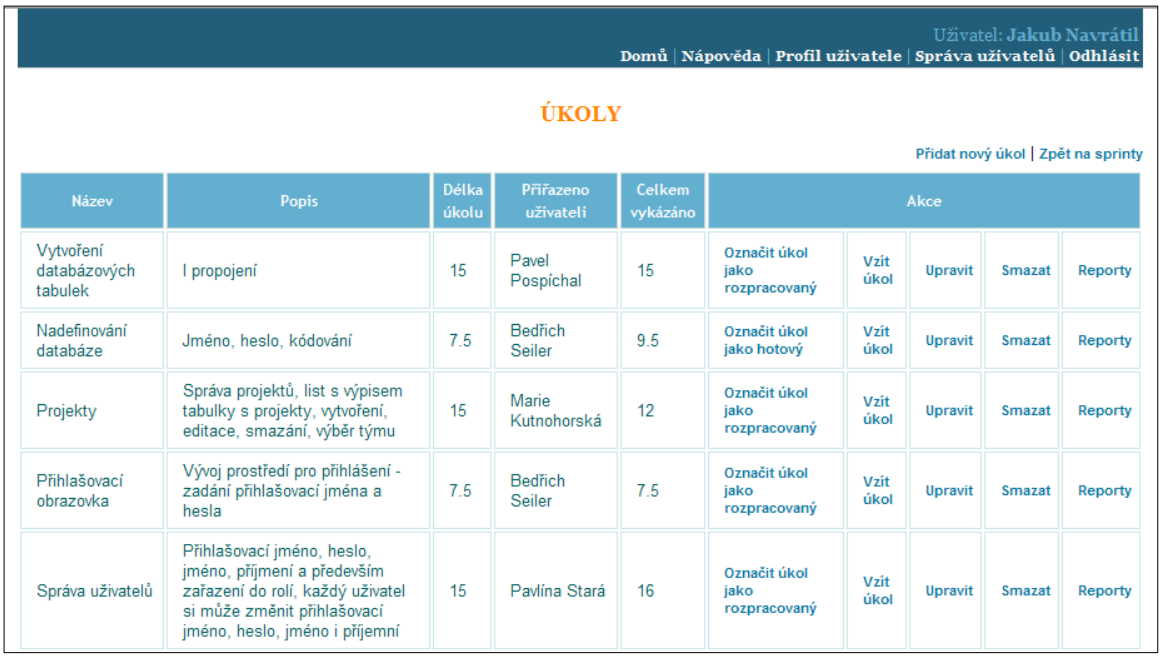
- Popis požadavků od zákazníka a výběr technologií (co má všechno aplikace SCRUM Board splňovat s ohledem na Stories definované Product Ownerem z Product Backlogu),
- stanovení cílů (stanovení cílů časových, nákladových ale i funkčních),
- Sprint Backlog (popis a plán počtu sprintů, počtu uživatelů v týmu a nástin úkolů),
- definice úkolů (tým se dohodne na úkolech v rámci jednotlivých sprintů).

Každý úkol je nastaven na 15 dnů a každý uživatel z týmu pracuje na jednom úkolu. Je nutné, aby uživatelé spolupracovali, komunikovali nad úkoly a sprinty, které vzniknou z požadavků a přizpůsobit těmto cílům další sprinty, které budou navazovat na sprint první.

Pro všechna ohodnocení byla použita bodová stupnice 1 bod = 1 man – days. Dle [31] se však hodnota bodu mění dle přístupu týmu, v případě prvních agilních projektů tato hodnota klesá. Body zde nemusí znamenat jeden pracovní den, opět se jedná o zcela individuální jednotku, v některé organizaci může být hodnota jednoho bodu 0,6 man – days v jiné např. 0,8 man - days. Ohodnocení body zbaví tým závislosti na čase a odhady ukončení jednotlivých části se zlepší. Pro všechny úkoly byly uživateli vypsány reporty, ve kterých je uveden počet bodů, který byl vynaložen pro splnění úkolu. Např. vytvoření Sprint Backlogu zabralo o dva body více, než bylo plánováno, čili 17. Na druhou stranu byl Burndown Chart

vyrovnán tím, že úkol s popisem požadavků od zákazníka a výběr technologie zabral pouze 13 bodů. Tím celý sprint trval opravdu 60 bodů, které byly naplánovány.

V pořadí druhý sprint s názvem Vytvoření databázových tabulek, propojení, správa uživatelů zahrnuje vytvoření databázových tabulek i s propojením a správu uživatelů, jejich přihlašovacích údajů a osobních údajů. Skládá se z pěti úkolů, tabulka s úkoly je zobrazena na obrázku 12.



Uživatel: Jakub Navrátil
Domů | Nápověda | Profil uživatele | Správa uživatelů | Odhlásit

ÚKOLY

Přidat nový úkol | Zpět na sprinty

Název	Popis	Délka úkolu	Přiřazeno uživateli	Celkem vykázáno	Akce				
Vytvoření databázových tabulek	I propojení	15	Pavel Pospíchal	15	Označit úkol jako rozpracovaný	Vzít úkol	Upravit	Smazat	Reporty
Nadefinování databáze	Jméno, heslo, kódování	7.5	Bedřich Seiler	9.5	Označit úkol jako hotový	Vzít úkol	Upravit	Smazat	Reporty
Projekty	Správa projektů, list s výpisem tabulky s projekty, vytvoření, editace, smazání, výběr týmu	15	Marie Kutnohorská	12	Označit úkol jako rozpracovaný	Vzít úkol	Upravit	Smazat	Reporty
Přihlašovací obrazovka	Vývoj prostředí pro přihlášení - zadání přihlašovací jména a hesla	7.5	Bedřich Seiler	7.5	Označit úkol jako rozpracovaný	Vzít úkol	Upravit	Smazat	Reporty
Správa uživatelů	Přihlašovací jméno, heslo, jméno, příjmení a především zařazení do rolí, každý uživatel si může změnit přihlašovací jméno, heslo, jméno i příjemní	15	Pavlína Stará	16	Označit úkol jako rozpracovaný	Vzít úkol	Upravit	Smazat	Reporty

Obrázek 12 - Příklad úkolů sprintu aplikace SCRUM Board (zdroj: autor)

Následuje třetí sprint, který se skládá opět z pěti úkolů. Tento sprint je zaměřen na správu sprintů, úkolů, reportů, celkovou funkčnost průchodnosti projekt → sprint → úkol → report a tvorba nápovědy k celé aplikaci. Všechny úkoly byly zpracovány uživateli, kteří přijali úkol „za svůj“ a byly vypsány reporty s ohodnocením úkolů body. Sprint s pořadovým číslem čtyři s názvem Bezpečnost a grafické rozhraní pak zaštiťuje bezpečnost a grafický vzhled celé aplikace. V tomto případě se jedná o čtyři úkoly:

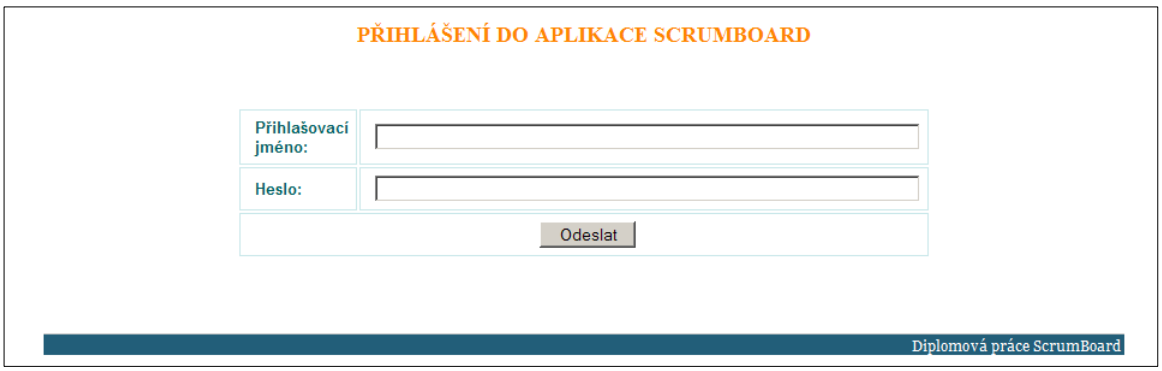
- Ošetření bezpečnosti (přihlášení pouze pro definované uživatele, odhlášení, zobrazování stránek dle oprávnění, funkce dle oprávnění),
- grafický návrh (vzhled, uživatelská příjemnost, kaskádové styly),
- povinně vyplňovaná pole (ošetření pomocí JavaScriptu),
- žádná oprávnění (generace stránek, kam je uživatel odkázán, pokud nemá dostatečná oprávnění).

Úkoly byly opět zpracovány členy týmu a reporty vykázaly, kromě malých nedostatků, dobrý odhad na bodové ohodnocení úkolů.

V posledním sprintu Testování probíhají nezávisle na sobě tři úkoly, které všechny testují funkčnost celé aplikace a průběžně jsou již otestované části projektu nasazovány buď na podnikový server, nebo na internet, dle přání zákazníka.

6.7 Návrh grafického rozhraní

Vzhled výsledného rozhraní je velice důležitý při vývoji každého informačního systému nebo internetové aplikace. Je nutné, aby byly barvy i uspořádání prvků aplikace zvoleny na základě využití. Uživatel, který bude pracovat s aplikací, vnímá příjemnost a přehlednost celého vzhledu ihned při prvním seznámení. Proto je dobré, aby uživatelé shledávali prostředí příjemné a především, aby ovládání celé aplikace bylo intuitivní. Základní přihlašovací obrazovka je zobrazena na obrázku 13. K definici barev byl využit designer barevných schémat [29]. Barvy byly zvoleny v odstínech modré a jako kontrastní barva byla zvolena oranžová. Oranžově jsou rozlišeny nadpisy, které udávají, kde se uživatel v danou chvíli nachází (Přihlášení do aplikace SCRUM Board, Projekty apod.).



Obrázek 13 - Přihlašovací obrazovka aplikace SCRUM Board (zdroj: autor)

Vzhled celé aplikace byl vytvořen pomocí kaskádových stylů. Na přiloženém CD se nachází soubor *style.css*, ve kterém je zdrojový kód pro úpravu vzhledu. Za pomocí tohoto nástroje je u vzhledu tabulek, nadpisů, odkazů a bloku textů zachována jednotnost a přehlednost. Obrazovka je rozlišena na hlavičku, tělo a patičku. V záhlaví je uvedeno jméno uživatele, který je přihlášen a poté dle práv uživatele nástroje pro zobrazování, upravování a celkové využívání tabule. V těle se nachází popis samotné práce – projekty, sprinty, úkoly, reporty, ale i změna přihlašovacích údajů. V patičce je uveden název aplikace a identifikace práce.

Obrázek 14 - Obrazovka s tabulkou úkolů (zdroj: autor)

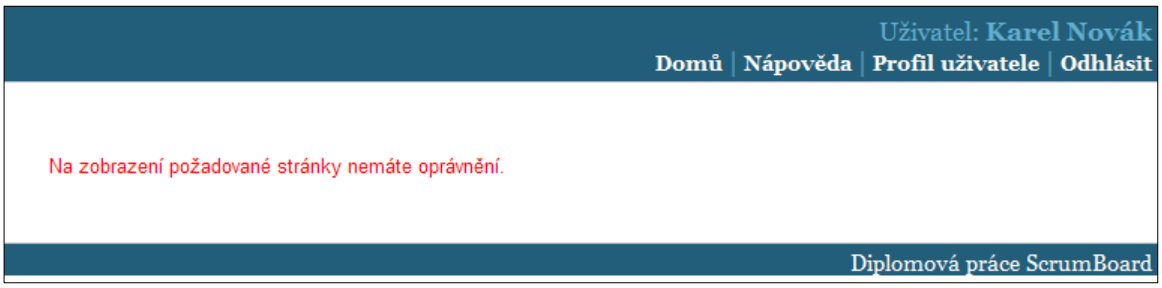
V těle celé aplikace se střídají odstíny modré, vše na bílém podkladu. Aplikace je tak přehledná a jednoduchá. Nad tabulkou se nachází název a vpravo je přidání nového projektu, sprintu či úkolu, pokud se uživatel nachází níže v projektu, objevuje se zde také tlačítko, které vrací uživatele vždy zpět o úroveň nahoru. To znamená, že pokud se uživatel nachází například v úrovni úkolů, přes tlačítko *Zpět na sprinty* se dostane na úroveň sprintů. Zobrazení lze vidět na obrázku 14.

6.8 Bezpečnost

Nejdůležitějšími kritérii pro bezpečnost celé aplikace jsou funkce, které zabrání dostat se do aplikace nepovolaným uživatelům. Proto je nezbytné disponovat přihlašovacím jménem a heslem, aby bylo možné pracovat v aplikaci. Tyto jména a hesla zpočátku generuje administrátor, každý uživatel si po přidělení může přihlašovací jméno i heslo změnit.

Další neméně důležitou vlastností celé aplikace je, že uživatel nemůže v adrese přepsat název stránky a dostat se na ni, pokud nemá práva k těmto funkcím. Například uživatele a jejich práva může definovat pouze Admin. Když uživatel s právy User v adrese přepíše *page=administration* a chce se dostat na správcovství uživatelů, rovnou je přesměrován na

page=no_permission a je mu zobrazeno hlášení na obrázku 15 s oznámením, že pro zobrazení požadované stránky nemá oprávnění.

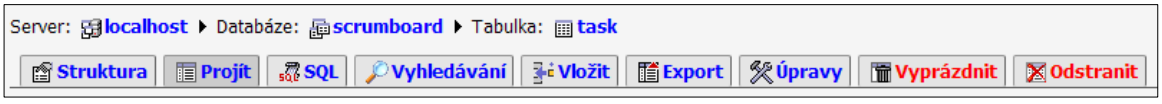


Obrázek 15 - Pokus o zobrazení stránky bez oprávnění (zdroj: autor)

Ošetření dle práv uživatelů jsou využívány v průběhu celého projektu. Pouze admin má práva pro využívání všech funkcí a definici vlastností. Pro vytvoření projektu se k administrátorovi připojuje se svými právy i Product Owner, avšak ostatní uživatelé nesmějí do aplikace zasahovat. Zabezpečení vzhledem k rolím uživatelů se opakuje ještě mnohokrát v průběhu celého vývoje softwaru. Každý uživatel se po dokončení práce s aplikací odhlásí. Odkaz pro odhlášení se nachází mezi nástroji v hlavičce aplikace. Tím je zabezpečeno, že se nikdo nedostane na účet daného uživatele a nijak neznehodnotí práci jak uživatele, jemuž účet patří, tak celého týmu do kterého náleží.

6.9 Export do tabulkového procesoru

Z phpMyAdminu lze z databáze exportovat tabulky do různých formátů. Patří sem například formát *.csv*, který lze otevřít v tabulkovém procesoru (např. MS Excel). Je nutné vybrat tabulku, která má být vyexportována a z nabídky na horní liště zvolit Export. Následuje výběr již zmíněných formátů. Lišta s nástroji v phpMyAdminu je zobrazena na obrázku 16.



Obrázek 16 - Zobrazení nástrojů v phpMyAdmin (zdroj: autor)

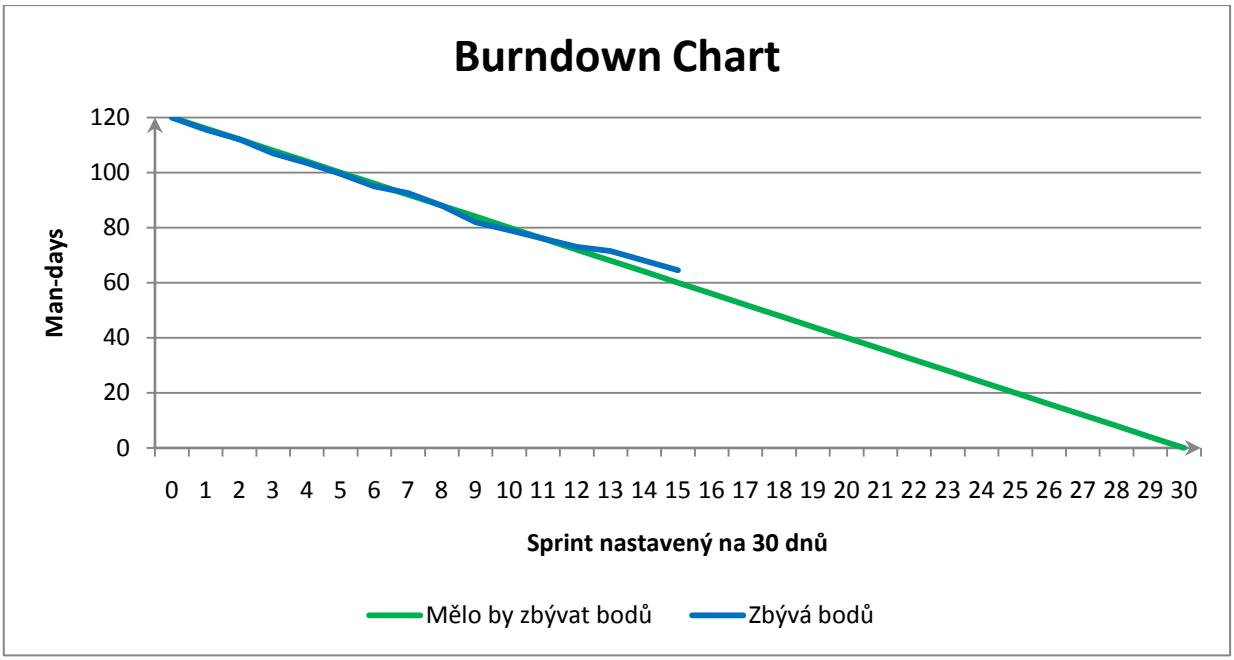
V tabulce 9 je zobrazen export tabulky Task (úkoly) z phpMyAdminu. Do posledního sloupce tabulky byly exportovány hodnoty z tabulky Report ze sloupce work_time (kolik je z úkolu již odpracováno). Z těchto exportů lze v MS Excel vygenerovat Burndown Chart. ScrumMaster může každý den na Daily meetingu vytvořit tabulky a každý den aktualizovat

graf daného sprintu. Toto je možné pouze u jednoduchých projektů, pokud by se jednalo o více projektů, je možné do aplikace přímo naprogramovat generování grafů sledování vývoje, protože pro ScrumMastera by bylo náročné sledovat více projektů společně.

Tabulka 9 - Export tabulky Task (úkolů) (zdroj: autor)

name	duration	done	FK_user_id	FK_sprint_id	work_time
"Vytvoření databázových tabulek"	"22.5"	"0"	"3"	"6"	"15"
"Bezpečnost přihlašování"	"25"	"0"	"16"	"6"	"8.5"
"Nadefinování databáze"	"7.5"	"0"	"3"	"6"	"0"
"Grafický návrh"	"12.5"	"1"	"4"	"6"	"13"
"Kaskádové styly"	"17.5"	"0"	"4"	"6"	"0.5"
"Zajištění hostingu"	"5"	"1"	"16"	"6"	"4.5"
"Správa uživatelů"	"19"	"0"	"7"	"6"	"1"
"Nápověda"	"11"	"1"	"7"	"6"	"13"

Ze sloupců duration a work_time je generován graf 7, a sice k 15. dni probíhajícího sprintu. Je patrné, že úkoly jsou zpožděny a je nutné více zapracovat, aby byl dodržen termín ukončení sprintu.



Graf 7 - Burndown Chart z exportovaných dat (zdroj: autor)

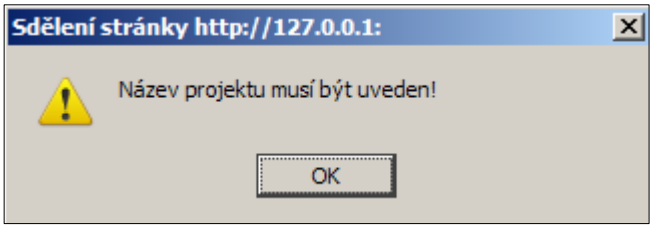
Jelikož je na tento projekt přidělen tým o čtyřech uživatelích s rolí User, počet man-days je násobek počtu Userů v týmu a počtu dnů jednoho sprintu, tedy 120, přičemž je bráno v úvahu, že uživatelé přiděleni na tento projekt, se věnují pouze tomuto projektu.

6.10 Ošetření pomocí JavaScriptu a testování

Vyplnění některých polí je povinné a tato nutnost je ošetřena pomocí JavaScriptu. Vznikl tak soubor *function.js*, který se nachází na přiloženém CD. V tomto souboru se nachází zdrojový kód všech funkcí. Pole, která jsou povinná pro vyplnění, jsou odlišená od ostatních tím, že barva textu je červená.

6. 10.1 Editace Projektu

Editace projektů provádí Product Owner nebo Admin. U projektů je nutné vypsat „Název projektu“ a poté pole, které označují plánovanou délku projektu, tedy „Zadáno od“ a „Ukončit do“. Pokud není nějaké z těchto polí vyplněno, funkce JavaScriptu oznámí uživateli nutnost vyplnění tím, že se zobrazí okno s upozorněním tak, jak je zobrazeno na obrázku 17.



Obrázek 17 - Okno s upozorněním, že nebylo vyplněno povinně vyplňované pole (zdroj: autor)

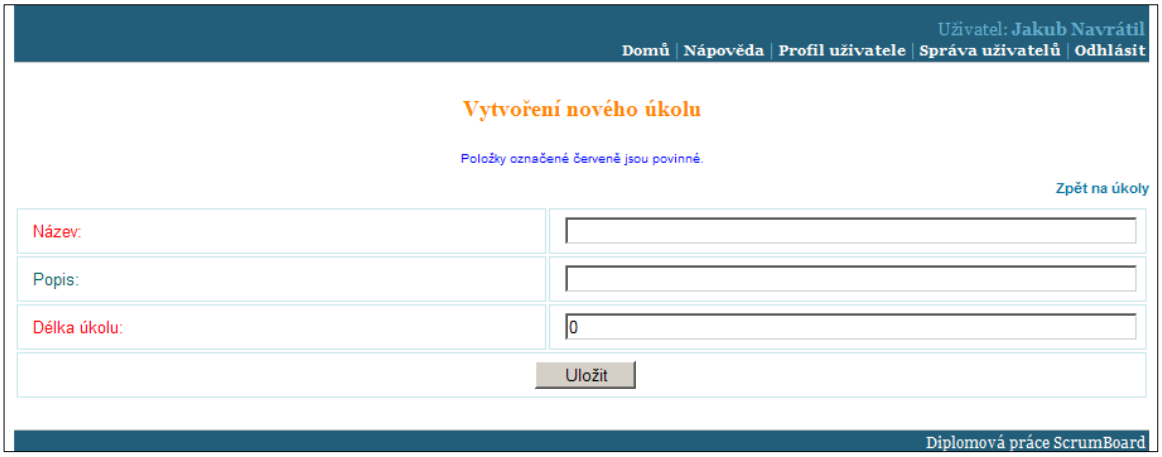
Položky výběru Product Ownera a ScrumMastera nejsou červeně označeny, ale u každého projektu je nutné tyto uživatele uvést. Ošetření bylo provedeno pomocí Select Boxu, kde není možné vybrat prázdné pole. Zde je primárně nastaven první Product Owner a první ScrumMaster, kteří byli nadefinováni.

6. 10.2 Editace Sprintu

Při tvorbě či úpravách sprintu je nutné vyplnit obě pole. Jak „Popis“, tak „Pořadí sprintu“. Pokud nejsou pole vyplněna, opět se zobrazí okno s upozorněním. Uživatel tak musí vyplnit popis sprintu i jeho pořadí. Pokud nevyplní ani jedno pole, v okně bude napsáno, že je nutné vyplnit popis. Uživatel odstraní okno s upozorněním kliknutím na tlačítko OK a vyplní popis. Pokud opět odešle sprint bez udání pořadí, zobrazí se další okno s upozorněním, že je nutné vyplnit pořadí. Okna s upozorněním jsou nastavena na postupné zobrazování tak, aby se vždy zobrazilo pouze okno vztahující se k prvnímu nevyplněnému poli. To znamená, že pokud nejsou vyplněna obě povinná pole, zobrazí se okno s upozorněním pouze pro první nevyplněné pole.

6. 10.3 Editace Úkolu

Pokud chce uživatel vytvořit nový úkol, povinné náležitosti jsou „Název“ a „Délka úkolu“. Pokud nejsou tato pole vyplněna, opět se zobrazí okno s upozorněním na nutnost vyplnění. Předpokládanou délku úkolu je nutno vyplnit z důvodu plánování sprintu celého týmu. Vzhled povinných polí při editaci úkolu je zobrazen na obrázku 18.



Obrázek 18 - Zobrazení povinně vyplňovaných polí (zdroj: autor)

Ty pole, která mají název červeným písmem, je nutné vyplnit. Pod názvem „Vytvoření nového projektu“ je modrým písmem tento fakt zaevidován. Toto upozornění modrým písmem se nachází na všech místech, kde se nachází povinně vyplňovaná pole.

6. 10.4 Vytvoření a smazání uživatele

Pro vytvoření nového uživatele je nutné vyplnit všechna pole. Výběr role je ošetřen opět pomocí Select Boxu, stejně jako u tvorby projektu. Bez přihlašovacího jména a hesla by nemohl žádný uživatel vstoupit do aplikace. Jméno a příjmení je také nutné uvádět především z důvodu správné identifikace uživatelů na projektech. Při smazání uživatele se zobrazí okno s otázkou, zda je požadavek na smazání opravdový. Tím se vyvaruje nechtěnému kliknutí na tlačítko smazat.

6. 10.5 Testování

Testování probíhalo především se zaměřením na funkčnost a použitelnost celé aplikace. Aplikace byla nahrána na server a spuštěna ve třech různých druzích prohlížečů (Internet Explorer 8, Google Chrome 10 a Mozilla Firefox 3.6). Funkčnost byla v pořádku, až na drobné chyby zobrazení, které byly opraveny. Celá aplikace i s balíkem, který byl použit k její tvorbě, byla spuštěna v operačních systémech Windows XP a Windows 7.

6.11 Nápověda

Nápověda se nachází v záhlaví mezi nástroji. Každý uživatel si může odkaz zobrazit kliknutím na text. Nejprve jsou vysvětleny pojmy v nástrojích, které se nachází v záhlaví. Jedná se o tyto odkazy:

- *Domů* - zpět na domovskou obrazovku se zobrazením vlastních projektů.
- *Nápověda* - zobrazí nápovědu k celé aplikaci.
- *Profil uživatele* - zobrazí profil uživatele (přihlašovací jméno, heslo, jméno, příjmení a oprávnění). Oprávnění smí měnit pouze admin. Zbylé hodnoty si může uživatel změnit pouze na svém účtu.
- *Správa uživatelů* - vytvoření, změna či smazání uživatele.
- *Odhlásit* - odhlášení z aplikace.

Tělo aplikace je řešeno tabulkou. Hierarchie procesů je následující:

Projekt → Sprint → Úkol → Report.

Projekt se skládá ze sprintů, sprint z úkolů a úkol z reportů. Projekt má tedy další tři podúrovně. V každé tabulce je v záhlaví sloupců uveden název (projektu, sprintu, úkolu). U sprintu je název zastoupen pořadím. U projektů je časové určení od kdy a především do kdy je projekt plánován. Pod záhlavím sloupce Akce jsou vždy funkce, které je možné nad jednotlivými úrovněmi provádět. Patří sem například zobrazení týmu u projektu, úpravy, smazání, u sprintů pak zobrazení podúrovně – úkolů. V části úkoly je ve sloupcích Akce možné označit úkol jako hotový/rozpracovaný, vzít úkol a opět upravit, smazat, ale také napsat report k úkolu.

Nápověda je tedy tvořena tak, že pod odkazem se nachází souvislý text s popisem všech funkcí, které může uživatel zobrazit. Nástroje záhlaví a nadpisy (např. Projekty, Sprinty atd.) jsou nadepsány oranžovou barvou textu. Záhlaví sloupců samotných tabulek v těle aplikace jsou pak napsány modrou barvou textu s efektem podtržení. Tak jsou rozvrženy funkce dle průchodu celou aplikací.

Závěr

V první části diplomového práce byly srovnány tradiční a agilní přístupy vývoje informačního systému. Dnešní doba si vyžaduje především rychlá řešení, a proto z tohoto srovnání vycházejí lépe agilní metodiky, které si kladou za cíl především splnění času a předepsaných zdrojů, než prvotní splnění funkcionalit předepsaných od zákazníka. To je dalším kladem, jelikož požadavky se často v průběhu vývoje mění.

Důraz byl kladen především na metodiku SCRUM. Bylo zjištěno, že na vývoji se podílí tři role uživatelů – Product Owner, který definuje požadavky od zákazníka, ScrumMaster, který odstiňuje jakákoliv rušení od týmu a právě tým, který se skládá z odborníků na samotný vývoj systému. Hierarchie této metodiky je taková, že z požadavků zákazníka Product Owner vytvoří Product Backlog, který je rozdělen na Sprint Backlog. Sprint Backlog v sobě nese rozdělení projektu na části – sprinty. Sprinty jsou pak rozděleny na úkoly a k úkolům jsou napsány reporty.

Bylo zjištěno, že významnou součástí metodiky SCRUM jsou především setkání, která probíhají v rámci týmu. Tato setkání probíhají každý den (daily meeting) a členové týmu zde hodnotí uplynulý pracovní den a nastiňují, co bude náplní práce nyní. Poté po uplynutí každého sprintu opět probíhá setkání (sprint meeting), kde jsou zhodnoceny cíle a vlastnosti dosavadního systému a je nastíněn další postup.

Dle metodiky SCRUM proběhl vývoj aplikace SCRUM Board. Tato aplikace má demonstrovat funkci daného přístupu SCRUM. V první řadě byly popsány funkce a vlastnosti, které by měla výsledná aplikace mít. Výsledný nástroj je elektronickou podobou klasické papírové tabule, která je v této metodice využívána. Tyto vlastnosti byly vypsány do Product Backlogu a byl vytvořen diagram Use Case s požadavky na systém a vznikly také scénáře k tomuto diagramu. Po ošetření vzniklo sedm databázových tabulek.

Pro samotnou tvorbu programu byl vybrán balík EasyPHP, který obsahuje programovací jazyk PHP, dále databázi MySQL a administrační rozhraní phpMyAdmin. Pomocí těchto nástrojů byla vytvořena aplikace SCRUM Board. Bezpečnost byla ošetřena pomocí přihlašovacích jmen a hesel všech nadefinovaných uživatelů. Pokud potencionální uživatel nedisponuje těmito údaji, do aplikace se nedostane. Grafické prostředí bylo vytvořeno tak, aby odpovídalo struktuře daného přístupu, je tedy především jednoduché, přehledné a intuitivní.

Při vytváření projektů, sprintů, úkolů, ale i uživatelů, s v databázi nacházejí pole, které je nutné vyplnit. Tato povinnost byla ošetřena pomocí JavaScriptu. Dále byly všechny funkce nástroje popsány do nápovědy, kterou se může každý přihlášený uživatel zobrazit. Funkce celé aplikace byly rozšířeny o export údajů, ze kterého je možné vytvořit graf sledování průběhu práce (Burndown Chart). Nástroj byl otestován a je plně funkční.

Cíl práce byl splněn. Ze srovnání vyšly lépe agilní metodiky vývoje software. Byla vytvořena aplikace SCRUM Board dle metodiky SCRUM, která demonstruje praktické použití daného přístupu. Tato aplikace slouží především jako ukázková, avšak lze ji dále vyvíjet a následně použít přímo pro vývoj informačního systému v organizaci, která se rozhodla postupovat dle metodiky SCRUM.

SEZNAM POUŽITÝCH ZDROJŮ

[1] *AAPM : The American Academy of Project Management* [online]. 1996, 2011 [cit. 2011-02-19]. Dostupné z WWW: <http://www.projectmanagementcertification.org/managernotes/what_is_pmtop.html>.

[2] Agile Alliance [online]. 2010 [cit. 2010-06-14]. Dostupné z WWW: <http://www.agilealliance.org>.

[3] *Agile alliance* [online]. 2001 [cit. 2011-02-10]. Dostupné z WWW: <http://www.agilealliance.org/the-alliance/the-agile-manifesto/>.

[4] AMBLER, Scott. *Dr Dobbs : The World of Software Development* [online]. 3.8.2006 [cit. 2011-02-10]. Dostupné z WWW: <http://www.drdobbs.com/architecture-and-design/191800169;jsessionid=I30VSRT32Q12RQE1GHPSKH4ATMY32JVN ?pgno=2>.

[5] BECK, Kent, et al. *Manifesto for Agile Software Development* [online]. 2001 [cit. 2011-02-10]. Dostupné z WWW: <http://www.agilemanifesto.org/>.

[6] BUCHALCEVOVÁ, A.; LEITL, M. *Průzkum používání agilních metodik v Č* [online]. Praha : Katedra informačních technologií VŠE Prah, 2006 [cit. 2011-02-10]. Dostupné z WWW: <http://nb.vse.cz/~buchalc/clanky/pruzkumobjekty2006.pdf>.

[7] BUCHALCEVOVÁ, Alena. *Metodiky budování informačních systémů*. první. Praha : Oeconomica, 2009. 206 s. ISBN 978-80-245-1540-3.

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

[9] COCKBURN, Alistair. *Use Cases : Jak efektivně modelovat aplikace*. Praha : Computer Press, říjen 2005. 264 s. ISBN 80-251-0721-3.

[10] *ČSSI : Česká společnost pro systémovou integraci* [online]. 2006 [cit. 2011-02-10]. Dostupné z WWW: <http://www.cssi.cz/cssi/stav-pouzivani-agilnich-metodik-v-cr>.

[11] *Dickinson College : Senior Seminar Software Engineering Basics* [online]. 24.8.2009 [cit. 2011-03-27]. Dostupné z WWW: <http://users.dickinson.edu/~wahlst/491/lectures/lec02.html>.

[12] HASS, Kathleen. *Project Smart : The Blending of Traditional and Agile Project Management* [online]. 05 2007 [cit. 2011-02-21]. Dostupné z WWW:

<http://www.projectsmart.co.uk/the-blending-of-traditional-and-agile-project-management.html>.

[13] *IBM* [online]. 2011 [cit. 2011-02-14]. Dostupné z WWW: <http://www-01.ibm.com/software/awdtools/rup/>.

[14] JEFFRIES, Ronald. *XProgramming.com* [online]. 1999, 2011 [cit. 2011-02-18]. Dostupné z WWW: <http://xprogramming.com/xpmag/whatisxp#whole>.

[15] KADLEC, Václav. Agilní programování : Metodiky efektivního vývoje softwaru. Brno : Computer Press, 2004. 278 s. ISBN 80-251-0342-0.

[16] KADLEC, Václav. *ŽIVĚ* [online]. 2003 [cit. 2011-02-15]. Dostupné z WWW: <http://www.zive.cz/clanky/programujte-agilne-nic-jineho-vam-nezbyva-a-nebo-ano/sc-3-a-111219/default.aspx>.

[17] KADLEC, Václav. *ŽIVĚ : Extremismus nemusí být na škodu: extrémní programování* [online]. 2003 [cit. 2011-02-15]. Dostupné z WWW: <http://www.zive.cz/clanky/extremismus-nemusi-byt-na-skodu-extremni-programovani/sc-3-a-111714/default.aspx>.

[18] KADLEC, Václav. *ŽIVĚ : Extrémní programování pod drobnohledem* [online]. 2003 [cit. 2011-02-19]. Dostupné z WWW: <http://www.zive.cz/clanky/extremni-programovani-pod-drobnohledem/sc-3-a-111952/default.aspx>.

[19] KADLEC, Václav. *ŽIVĚ : Umějí Češi programovat extrémně?* [online]. 2003 [cit. 2011-02-19]. Dostupné z WWW: <http://www.zive.cz/clanky/umeji-cesi-programovat-extremne/sc-3-a-112665/default.aspx>.

[20] KHRAMTCHENK, Serguei. *Feature Driven Development : Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments* [online].Harvard University, 17.5.2004 [cit. 2011-02-19]. Dostupné z WWW: <http://www.featuredrivendevelopment.com/files/FDD_vs_XP.pdf>.

[21] KNESL, Jiří. *Zdroják.cz* [online]. 18.12.2009 [cit. 2011-02-27]. Dostupné z WWW: <http://zdrojak.root.cz/clanky/agilni-vyvoj-scrum/>.

[22] KNESL, Jiří. *Zdroják.cz* [online]. 11.12.2009 [cit. 2011-02-27]. Dostupné z WWW: <http://zdrojak.root.cz/clanky/agilni-vyvoj-uvod/>.

[23] KUČERA, František. *Zdroják.cz* [online]. 10.3.2010 [cit. 2011-04-07]. Dostupné z WWW: <http://zdrojak.root.cz/clanky/java-na-webovem-serveru-porovnani-javy-a-php/>.

[24] Mountain goat software [online]. 1998-2010 [cit. 2010-06-14]. Dostupné z WWW: {$<$}http://www.mountaingoatsoftware.com/{$>$}.

[25] *Mountain Goat Software : What is Scrum?* [online]. 1998, 2011 [cit. 2011-02-28]. Dostupné z WWW: <http://www.mountaingoatsoftware.com/topics/scrum>.

[26] *Objektová analýza, návrh a programování* [online]. 2008 [cit. 2011-02-11]. Dostupné z WWW: <http://objekty.vse.cz/Objekty/Rup2>.

[27] *Project Management Research Institute* [online]. 2010 [cit. 2011-03-28]. Dostupné z WWW: <http://www.collabteam.com/pmrinew/1consulting_scrum.html>.

[28] SCHWABER, Ken. Agile Project Management With Scrum. United States : Microsoft Press, 2004. 163 s. ISBN 978073561993.

[29] STANICEK, Petr. *Color Scheme Designer 3* [online]. 2002, 2010 [cit. 2011-03-26]. Dostupné z WWW: <http://colorschemedesigner.com/>.

[30] ŠOCHOVÁ, Zuzana. *CIO : Business world* [online]. 25.6.2009 [cit. 2011-02-10]. Dostupné z WWW: <http://businessworld.cz/business-rizeni-podniku/Agilni-metody-a-Scrum-4740>.

[31] ŠOCHOVÁ, Zuzana. *Zuzi´s blog : Agile, Scrum, XP* [online]. 30.7.2008 [cit. 2011-03-27]. Dostupné z WWW: <http://soch.cz/blog/management/agile/scrum-management/ohodnoceni-uloh-body/>.

[32] ŠOCHOVÁ, Zuzana. *Zuzi´s blog : Agile, Scrum, XP* [online]. 22.7.2008 [cit. 2011-02-10]. Dostupné z WWW: <http://soch.cz/blog/management/proc-neco-menit/#comments>.

[33] VRÁNA, Jakub. *PHP triky : Srovnání funkcí MySQL a PostgreSQL* [online]. 17.6.2009 [cit. 2011-04-07]. Dostupné z WWW: <http://php.vrana.cz/srovnani-funkci-mysql-a-postgresql.php>.

[34] WATERS, Kelly. *All About Agile* [online]. 16.8.2010 [cit. 2011-02-19]. Dostupné z WWW: <http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/>.

[35] WELLS, Don. *XP : Extreme programming* [online]. 1999 [cit. 2011-02-18]. Dostupné z WWW: <http://www.extremeprogramming.org/rules.html>.

[36] ZIKMUND, Martin. *Agilní projektové řízení - novinka stará přes 20 let* [online]. 12.4.2010 [cit. 2011-02-21]. Dostupné z WWW: <http://www.businessvize.cz/rizeni-a-optimalizace/agilni-projektove-rizeni>.

SEZNAM ZKRATEK

Apod.	*A podobně*
Atd.	*A tak dále*
CSS	*Cascading Style Sheets* (jazyk pro popis zobrazení internetových stránek)
CRUD	*Create, Read, Update, Delete* (vytvoření, čtení, aktualizace a smazání)
ČR	*Česká republika*
HTTP	*Hypertext Transfer Protocol* (soubor pravidel pro přenos souborů přes World Wide Web)
MS Excel	*Microsoft Excel* (tabulkový procesor)
MySQL	*My Structured Query Language* (databázový systém)
Např.	*Například*
PHP	*PHP: Hypertext preprocessor, Personal home page* (skriptovací programovací jazyk)
RUP	*Rational Unified Process*
SQL	*Structured Query Language*
Tzn.	*To znamená*
UML	*Unified model language* **(**unifikovaný modelovací jazyk**)**
USD	*United States Dolar* (americké dolary – měna)
VŠE	*Vysoká škola ekonomická*
WWW	*World wide web* (celosvětová síť hypertextových dokumentů)

SEZNAM PŘÍLOH

PŘÍLOHA 1- SCÉNÁŘE K DIAGRAMU PŘÍPADU UŽITÍ

Scénář "Označení mého úkolu" (zdroj: autor)

Název Use Case		**Označení mého úkolu**
Aktér		User, Admin
Popis		Uživatel označí úkol buď jako rozpracovaný nebo jako hotový
Vstupy		Uživatel začal pracovat na úkolu
Výstupy		Úkol je rozpracovaný nebo hotový
Spouštěcí událost		Uživatel má úkol rozpracovaný nebo ukončený
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na název projektu, ve kterém chce přijmout úkol
2	Systém	Zobrazí sprinty, které náleží pod daný projekt
3	Aktér	Ve sloupci *Akce* klikne na *Úkoly* v řádku sprintu, ve kterém se úkol nachází
4	Systém	Zobrazí úkoly k danému sprintu
5	Aktér	POKUD má úkol hotový, klikne ve sloupci *Akce* na *Označit úkol jako hotový* JINAK klikne v tom samém sloupci na *Označit úkol jako rozpracovaný* KONEC POKUD
6	Systém	Ve sloupci *Akce* zobrazí úkol jako hotový nebo rozpracovaný

Scénář „Zobrazení informací o mém projektu“ (zdroj: autor)

Název Use Case		**Zobrazení informací o mém projektu**
Aktér		User, ScrumMaster
Popis		Uživatel si zobrazí veškeré informace o projektu, na kterém pracuje
Vstupy		Uživatel musí být zařazen do týmu, který na daném projektu pracuje
Výstupy		Zobrazení popisu projektu, sprintů, úkolů, reportů v daném projektu
Spouštěcí událost		Uživatel je zařazen do týmu projektu
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na název projektu, o kterém chce zjistit informace
2	Systém	Zobrazí počet sprintů, pořadí, popis, předpokládanou dobu trvání, kolik je na projektu zatím vykázáno a zobrazení akcí dle role uživatele
3	Aktér	Ve sloupci *Akce* klikne na *Úkoly* v řádku daného sprintu
4	Systém	Zobrazí úkoly k danému sprintu, jejich název, popis, délku úkolu, jméno uživatele, který na projektu pracuje, kolik je prozatím na úkolu vykázáno a dále akce dle role uživatele
5	Aktér	Klikne na *Reporty* k danému úkolu
6	Systém	Zobrazí jak dlouho bylo prozatím na úkolu pracováno, kdo napsal report, komentář k reportu a kdy byl report napsán, dále dle role uživatele zobrazí akce

Scénář „Změna osobních údajů“ (zdroj: autor)

Název Use Case		**Změna osobních údajů**
Aktér		User, ScrumMaster, Product Owner
Popis		Uživatel změní své osobní údaje (přihlašovací jméno, heslo, jméno, příjmení)
Vstupy		Přihlášený uživatel
Výstupy		Změna osobních údajů
Spouštěcí událost		Rozhodnutí pro změnu některého z osobních údajů
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne v záložkách v hlavičce aplikace na *Profil uživatele*
2	Systém	Zobrazí osobní údaje uživatele (přihlašovací jméno, heslo, jméno, příjmení)
3	Aktér	Si změní v polích některý údaj a klikne na tlačítko *Uložit*
4	Systém	Změní údaje a vrátí uživatele na základní obrazovku s výpisem projektů

Scénář „Zobrazení nápovědy“ (zdroj: autor)

Název Use Case		**Zobrazení nápovědy**
Aktér		User, ScrumMaster, Product Owner, Admin
Popis		Uživatel zobrazí nápovědu k celé aplikaci
Vstupy		Přihlášený uživatel
Výstupy		Zobrazení nápovědy
Spouštěcí událost		Uživatel nerozumí nějaké položce v aplikaci
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne v záložkách v hlavičce aplikace na *Nápověda*
2	Systém	Zobrazí nápovědu k aplikaci
3	Aktér	Čte nápovědu

Scénář „Vytvoření týmu na můj projekt“ (zdroj: autor)

Název Use Case		**Vytvoření týmu na můj projekt**
Aktér		ScrumMaster
Popis		Aktér přiřadí na projekt Tým z uživatelů s rolí User
Vstupy		Byl vytvořen projekt
Výstupy		Funkční tým přiřazený na projekt
Spouštěcí událost		Projekt nemá žádný tým, který by na něm pracoval, má pouze nadefinovaného Product Ownera a ScrumMastera
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Tým*
2	Systém	Zobrazí seznam uživatelů, kteří jsou v týmu a pracují na tomto projektu
3	Aktér	POKUD chce aktér přidat uživatele, klikne na *Přidat nového uživatele do týmu* tak, že vybere rolety, která nabízí uživatele, kteří již na tomto projektu nepracují JINAK může odebrat uživatele z týmu tak, že klikne na *Odstranit uživatele z týmu* KONEC POKUD
4	Systém	Zobrazí seznam uživatelů, kteří jsou v týmu a pracují na tomto projektu

Scénář „Správa uživatelů“ (zdroj: autor)

Název Use Case		**Správa uživatelů**
Aktér		Admin
Popis		Aktér pracuje s daty uživatelů aplikace
Výstupy		Nový uživatel, změna údajů uživatele, zobrazení údajů uživatele nebo smazání uživatele
Spouštěcí událost		Nový zaměstnanec, změna zaměstnance, odchod zaměstnance
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve v záložkách záhlaví aplikace na *Správa uživatelů*
2	Systém	Zobrazí seznam uživatelů, kteří evidováni v aplikaci
3	Aktér	Spravuje uživatele

Scénář „Vytvoření nového uživatele“ (zdroj: autor)

Název Use Case		**Vytvoření nového uživatele**
Aktér		Admin
Popis		Aktér vytvoří nového uživatele
Vstupy		Nutnost vytvoření nového uživatele
Výstupy		Nový uživatel
Spouštěcí událost		Nový zaměstnanec
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na *Přidat nového uživatele*
2	Systém	Zobrazí pole přihlašovací jméno, heslo, jméno, příjmení a oprávnění
3	Aktér	Vyplní údaje uživatele a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že byl uživatel vytvořen a zobrazí seznam všech uživatelů

Scénář „Změna údajů uživatele“ (zdroj: autor)

Název Use Case		**Změna údajů uživatele**
Aktér		Admin
Popis		Aktér změní údaje uživatele
Výstupy		Změna údajů uživatele
Spouštěcí událost		Změna údajů zaměstnance, např. příjmení, oprávnění apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Upravit*
2	Systém	Zobrazí pole s vyplněným přihlašovacím jménem, heslem, jménem, příjmením a oprávněním
3	Aktér	Změní údaje uživatele a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že údaje uživatele byly změněny a zobrazí seznam všech uživatelů

Scénář „Smazání uživatele“ (zdroj: autor)

Název Use Case		**Smazání uživatele**
Aktér		Admin
Popis		Aktér smaže existujícího uživatele
Výstupy		Uživatel již není veden v databázi uživatelů
Spouštěcí událost		Odchod zaměstnance
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Smazat*
2	Systém	Zobrazí oznámení, zda chce aktér opravdu uživatele smazat
3	Aktér	Klikne na *OK*
4	Systém	Smaže uživatele a zobrazí seznam všech uživatelů

Scénář „Správa projektů“ (zdroj: autor)

Název Use Case		**Správa projektů**
Aktér		Admin, Product Owner
Popis		Aktér pracuje s projekty
Výstupy		Nový projekt, změna údajů v projektu, zobrazení informací o projektu nebo smazání projektu
Spouštěcí událost		Nový projekt, změna projektu, projekt nebude realizován
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Pracuje s tabulkou projektů
2	Systém	Zobrazí seznam všech projektů
3	Aktér	Spravuje projekty

Scénář „Vytvoření nového projektu“ (zdroj: autor)

Název Use Case		**Vytvoření nového projektu**
Aktér		Admin, Product Owner
Popis		Aktér vytvoří nový projekt
Výstupy		Nový projekt
Spouštěcí událost		Nový projekt zadaný od zákazníka
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na *Přidat nový projekt*
2	Systém	Zobrazí pole název, popis, zadáno od, zadáno do a výběr Product Ownera a ScrumMastera
3	Aktér	Vyplní údaje o projektu a vybere Product Ownera a ScrumMastera a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že byl projekt vytvořen a zobrazí seznam všech projektů

Scénář „Změna údajů projektu“ (zdroj: autor)

Název Use Case		**Změna údajů projektu**
Aktér		Admin, Product Owner
Popis		Aktér změní údaje o projektu
Výstupy		Změna údajů v projektu
Spouštěcí událost		Změna požadavků zákazníka, času na zpracování apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Upravit*
2	Systém	Zobrazí pole s vyplněným názvem, popisem a daty začátku a konce projektu
3	Aktér	Změní údaje projektu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že údaje projektu byly změněny a zobrazí seznam všech projektů

Scénář „Smazání projektu“ (zdroj: autor)

Název Use Case		**Smazání projektu**
Aktér		Admin, Product Owner
Popis		Aktér smaže existující projekt
Výstupy		Projekt již není veden v databázi projektů
Spouštěcí událost		Projekt nebude realizován
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Smazat*
2	Systém	Zobrazí oznámení, zda chce aktér opravdu projekt smazat
3	Aktér	Klikne na *OK*
4	Systém	Smaže projekt a zobrazí seznam všech zbylých projektů

Scénář „Zobrazení detailů projektu“ (zdroj: autor)

Název Use Case		**Zobrazení detailů projektu**
Aktér		Admin, Product Owner, ScrumMaster
Popis		Aktér zobrazí detaily projektu
Výstupy		Zobrazení detailů projektu
Spouštěcí událost		Product Owner kontroluje průběh projektu
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na název daného projektu
2	Systém	Zobrazí seznam a informace o sprintu
3	Aktér	Klikne na úkoly daného sprintu
4	Systém	Zobrazí seznam a informace o úkolu
5	Aktér	Klikne na report k danému úkolu
6	Systém	Zobrazí popis daného reportu

Scénář „Správa sprintů“ (zdroj: autor)

Název Use Case		**Správa sprintů**
Aktér		Admin, Product Owner, ScrumMaster
Popis		Aktér pracuje se sprinty
Výstupy		Nový sprint, změna údajů ve sprintu, zobrazení informací o sprintu nebo smazání sprintu
Spouštěcí událost		Nový sprint, změna sprintu, sprint nebude realizován
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na odkaz pod názvem projektu
2	Systém	Zobrazí seznam všech sprintů
3	Aktér	Pracuje se sprinty

Scénář „Vytvoření nového sprintu“ (zdroj: autor)

Název Use Case		**Vytvoření nového sprintu**
Aktér		Admin, Product Owner, ScrumMaster
Popis		Aktér vytvoří nový sprint
Výstupy		Nový sprint
Spouštěcí událost		Začátek nového sprintu
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na *Přidat nový sprint*
2	Systém	Zobrazí pole popis a pořadí sprintu
3	Aktér	Vyplní údaje o sprintu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že sprint byl vytvořen a zobrazí seznam všech sprintů

Scénář „Změna údajů sprintu“ (zdroj: autor)

Název Use Case		**Změna údajů sprintu**
Aktér		Admin, Product Owner, ScrumMaster
Popis		Aktér změní údaje o sprintu
Výstupy		Změna údajů ve sprintu
Spouštěcí událost		Změna popisu sprintu, pořadí apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Upravit*
2	Systém	Zobrazí pole s vyplněným popisem a pořadím sprintu
3	Aktér	Změní údaje sprintu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že údaje sprintu byly změněny a zobrazí seznam všech sprintů v daném projektu

Scénář „Smazání sprintu“ (zdroj: autor)

Název Use Case		**Smazání sprintu**
Aktér		Admin, Product Owner, ScrumMaster
Popis		Aktér smaže existující sprint
Výstupy		Sprint již není veden v databázi sprintů
Spouštěcí událost		Sprint nebude realizován, uživatelé mají hotové všechny úkoly k projektu apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Smazat*
2	Systém	Zobrazí oznámení, zda chce aktér opravdu sprint smazat
3	Aktér	Klikne na *OK*
4	Systém	Smaže sprint a zobrazí seznam všech zbylých sprintů vztahujících se k danému projektu

Scénář „Správa úkolů“ (zdroj: autor)

Název Use Case		**Správa úkolů**
Aktér		Admin, ScrumMaster
Popis		Aktér pracuje s úkoly
Výstupy		Nový úkol, změna údajů úkolu, zobrazení informací o úkolu nebo smazání úkolu
Spouštěcí událost		Nový úkol, změna úkolu, úkol nebude realizován
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na odkaz pod názvem projektu
2	Systém	Zobrazí seznam všech sprintů
3	Aktér	Klikne ve sloupci *Akce* na *Úkoly* k danému sprintu
4	Systém	Zobrazí tabulku s úkoly
5	Aktér	Pracuje s úkoly

Scénář „Vytvoření nového úkolu“ (zdroj: autor)

Název Use Case		**Vytvoření nového úkolu**
Aktér		Admin, ScrumMaster
Popis		Aktér vytvoří nový úkol
Výstupy		Nový úkol
Spouštěcí událost		Další požadavek od zákazníka apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na *Přidat nový úkol*
2	Systém	Zobrazí pole s názvem, popisem a délkou úkolu
3	Aktér	Vyplní údaje o úkolu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že úkol byl vytvořen a zobrazí seznam všech úkolů

Scénář „Změna údajů úkolu“ (zdroj: autor)

Název Use Case		**Změna údajů úkolu**
Aktér		Admin, ScrumMaster
Popis		Aktér změní údaje o úkolu
Výstupy		Změna údajů úkolů
Spouštěcí událost		Změna popisu úkolu, délky trvání apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Upravit*
2	Systém	Zobrazí pole s vyplněným názvem, popisem a délkou úkolu
3	Aktér	Změní údaje o úkolu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že údaje úkolu byly změněny a zobrazí seznam všech úkolů v daném sprintu

Scénář „Smazání úkolu“ (zdroj: autor)

Název Use Case		**Smazání úkolu**
Aktér		Admin, ScrumMaster
Popis		Aktér smaže existující úkol
Výstupy		Úkol již není veden v databázi úkolů
Spouštěcí událost		Úkol nebude realizován, zákazník změnil požadavky apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Smazat*
2	Systém	Zobrazí oznámení, zda chce aktér opravdu úkol smazat
3	Aktér	Klikne na *OK*
4	Systém	Smaže úkol a zobrazí seznam všech zbylých úkolů vztahujících se k danému sprintu

Scénář „Správa reportů“ (zdroj: autor)

Název Use Case		**Správa reportů**
Aktér		Admin, User
Popis		Aktér pracuje s reporty
Výstupy		Nový report, změna údajů reportu, zobrazení informací o reportu nebo smazání reportu
Spouštěcí událost		Nový report, změna reportu, report je chybný
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na odkaz pod názvem projektu
2	Systém	Zobrazí seznam všech sprintů
3	Aktér	Klikne ve sloupci *Akce* na *Úkoly* k danému sprintu
4	Systém	Zobrazí tabulku s úkoly
5	Aktér	Klikne na *Reporty* ve sloupci *Akce*
11	Systém	Zobrazí tabulku s reporty
12	Aktér	Pracuje s reporty

Scénář „Vytvoření nového reportu“ (zdroj: autor)

Název Use Case		**Vytvoření nového reportu**
Aktér		Admin, User
Popis		Aktér vytvoří nový report
Výstupy		Nový report
Spouštěcí událost		Změna průběhu úkolu apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne na *Přidat nový report*
2	Systém	Zobrazí pole s délkou prací na úkolu a komentářem
3	Aktér	Vyplní údaje o reportu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že report byl vytvořen a zobrazí seznam všech reportů, kdo je vypsal a kdy

Scénář „Změna údajů reportu“ (zdroj: autor)

Název Use Case		**Změna údajů reportu**
Aktér		Admin, User
Popis		Aktér změní údaje v reportu
Výstupy		Změna údajů reportu
Spouštěcí událost		Změna popisu úkolu, délky trvání, nebo jiný uživatel apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Upravit*
2	Systém	Zobrazí pole s vyplněnou délkou práce na úkolu a komentářem
3	Aktér	Změní údaje o reportu a klikne na tlačítko *Uložit*
4	Systém	Oznámí, že údaje reportu byly změněny a zobrazí seznam všech reportů k danému úkolu

Scénář „Smazání reportu“ (zdroj: autor)

Název Use Case		**Smazání reportu**
Aktér		Admin, User
Popis		Aktér smaže existující report
Výstupy		Report již není veden v databázi reportů
Spouštěcí událost		Úkol nebude realizován, zákazník změnil požadavky nebo je report chybný apod.
Hlavní scénář		
Krok	**Role**	**Akce**
1	Aktér	Klikne ve sloupci *Akce* na *Smazat*
2	Systém	Zobrazí oznámení, zda chce aktér opravdu report smazat
3	Aktér	Klikne na *OK*
4	Systém	Smaže report a zobrazí seznam všech zbylých reportů vztahujících se k danému úkolu

PŘÍLOHA 2 - CD – APLIKACE SCRUM BOARD

Přiložené CD obsahuje:

- zdrojové kódy nástroje SCRUM Board v jazyce PHP
- zdrojový kód pro grafickou úpravu aplikace SCRUM Board – kaskádové styly
- zdrojový kód pro ošetření povinně vyplňovaných polí v SCRUM Board v JavaScriptu
- zdrojový kód pro vytvoření databázových tabulek i samotné databáze SCRUM Board - SQL