TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Návrh interní metodiky v automobilovém průmyslu

Bakalářská práce

Studijní program: B6209 – Systémové inženýrství a informatika
Studijní obor: 6209R021 – Manažerská informatika

Autor práce: **Tereza Bartošová**
Vedoucí práce: doc. Ing. Klára Antlová, Ph.D.

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Economics

Internal Methodology Concept in the Automotive Industry

Bachelor thesis

Study programme: B6209 – System Engineering and Informatics
Study branch: 6209R021 – Managerial Informatics

Author: **Tereza Bartošová**
Supervisor: doc. Ing. Klára Antlová, Ph.D.

Liberec 2018

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Akademický rok 2017 / 2018

Zadání bakalářské práce

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tereza Bartošová**
Osobní číslo:	E17000002
Studijní program:	B6209 Systémové inženýrství a informatika
Studijní obor:	B6209R021 – Manažerská informatika
Zadávající katedra:	katedra informatiky
Vedoucí práce:	doc. Ing. Klára Antlová, Ph.D.
Konzultant práce:	Ing. Pavel Tůma ŠKODA AUTO a. s. - Portfolio manažer vývoje aplikací
Název práce:	**Návrh interní metodiky v automobilovém průmyslu**

Zásady pro vypracování:

1. Možnosti agilních a rigorózních metodik.
2. Využívání metodiky pro vývoj v automobilovém průmyslu.
3. Agilní metodiky v oblasti vývoje mobilních aplikací.
4. Návrh na propojení metodik vývoje automobilu včetně softwaru.
5. Vyhodnocení navrženého propojení metodik.

Seznam odborné literatury:

ŠOCHOVÁ, Zuzana a Eduard KUNCE. 2014. *Agilní metody řízení projektů*. Brno: Computer Press. ISBN 978-80251-4194-6.

MÁCHAL, Pavel, Martina KOPEČKOVÁ a Radmila PRESOVÁ. 2015. *Světové standardy projektového řízení: pro malé a střední firmy : IPMA, PMI, PRINCE2*. Praha: GRADA Publishing. ISBN 978-80-247-5321-8.

MYSLÍN, Josef. 2016. *Scrum: průvodce agilním vývojem softwaru*. Brno: Computer Press. ISBN 978-80-251-4650-7.

PROQUEST. 2017. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2017-09-28]. Dostupné z: http://knihovna.tul.cz/

Rozsah práce:	30 normostran
Forma zpracování:	tištěná / elektronická
Datum zadání práce:	31. října 2017
Datum odevzdání práce:	31. srpna 2019

Technická univerzita v Liberci
Ekonomická fakulta

L.S.

prof. Ing. Miroslav Žižka, Ph.D.
děkan Ekonomické fakulty

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2017

Prohlášení

Byla jsem seznámena s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracovala samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Nejprve bych chtěla poděkovat vedoucímu mé bakalářské práce, paní doc. Ing. Kláře Antlové, Ph.D., za její rady při vytváření osnovy, odborné vedení a věcné připomínky. V neposlední řadě, bych také ráda poděkovala konzultantovi Ing. Pavlu Tůmovi, který mě uvedl do dané problematiky. Dále také za jeho podporu a odborné rady během vytváření bakalářské práce.

Anotace

Cílem bakalářské práce je navrhnout propojení metodik ve společnosti ŠKODA Auto a.s., pro zefektivnění procesů při vývoji softwaru. V teoretické části jsou rozebrány rigorózní a agilní metodiky, které se využívají při vývoji konkrétního softwaru (např. mobilní aplikace). Dále budou popsány nástroje pro agilní přístup a vedení projektů v automobilovém průmyslu podle interních směrnic společnosti ŠKODA Auto. V dnešní době jsou stále rozšířenější agilní metodiky, a to především díky jejich flexibilitě a schopnosti reagovat na měnící se požadavky zákazníka v průběhu vývoje, proto se tato bakalářská práce zaměřuje více na agilní metodiku SCRUM a nástroj JIRA, které jsou zde detailně popsané. Závěr bakalářské práce se zabývá vyhodnocením navrženého řešení.

Klíčová slova

Agilní metodiky, Crystal metodiky, Extrémní programování, JIRA, Kanban, Lean Development, PEP, Rational Unified Process, Rigorózní metodiky, SCRUM, Spirálový model, Vodopádový model

Anotatiton

The goal of this bachelor thesis is to propose methodologies in ŠKODA Auto a.s. to increase effectivity of processes in software development. In the theoretical part, the rigorous and agile methodologies used in the development of specific software (e.g. mobile applications) will be analyzed. In addition, tools for agile approach and project management in the automotive industry will be described according to Škoda Auto internal guidelines. Nowadays, agile methodologies are becoming more widespread, mainly because of their flexibility and ability to respond to changing customer requirements during development, so this bachelor thesis focuses more on the agile methodology SCRUM and JIRA, which are described in detail here. The conclusion of the bachelor thesis deals with the evaluation of the proposed solution.

Keywords

Agile metodologies, Crystal metodologies, Extreme programming, JIRA, Kanban, Lean Development, PEP, Rational Unified Process, Rigorous metodologies, SCRUM, Spiral model, Waterfall model

Obsah

Seznam zkratek a cizích pojmů

ŠA	ŠKODA Auto a.s.
VW	Volkswagen Group
SW	Software
PL	Projektový manažer (Project Leader)
FO	Funkční vlastník (Functional Owner)
TOV	Zodpovědný tester
RUP	Rational Unified Process
SCRUM	SCRUM Development Process
JIRA	Nástroj pro agilní vývoj
UML	Unified Modeling Language
XP	Extrémní programování (Extreme programming)
PO	Vlastník produktu (Product Owner)
PB	Produkt backlog
SB	Sprint backlog
IT	Informační technologie (Information technology)
PEP	Proces vzniku výrobku

Seznam tabulek

Seznam obrázků

Úvod

Tato bakalářská práce s názvem Návrh interní metodiky v automobilovém průmyslu se zabývá problematikou výběru rigorózních a agilních metodik pro vývoj softwaru. Toto téma je v dnešní době často diskutované. Názory na využití jednotlivých metodik se různí a každá společnost se na tuto vzniklou situaci dívá odlišně.

Teoretická část bakalářské práce se zaměřuje na vysvětlení rozdílů mezi rigorózními a agilními metodikami. Jsou zde také vybrány jedny z nejzásadnějších či nejznámějších zástupců z každé skupiny metodik. Mezi rigorózní metodiky se řadí: Vodopádový model, Spirálový model a Rational Unified Process. A zmíněné agilní metodiky jsou: SCRUM Development Process, Extrémní programování, Crystal metodiky, Kanban a Lean Development.

Další část se zabývá analýzou metodiky, která se používá při výrobě vozů. Tato metodika se ve ŠKODA Auto a.s. nazývá PEP. Dále se práce specializuje na nástroje, které se využívají pro usnadnění procesu vývoje softwaru. Existuje již celá řada vytvořených a úspěšných nástrojů, které si našly své místo. Avšak každý tým si může navrhnout svůj vlastní nástroj. Mezi nejpoužívanější nástroje se řadí JIRA a Rational team concert.

Cílem praktické části bakalářské práce je navržení interní metodiky pro společnost ŠKODA Auto a.s., která by vyřešila problém, který se týká propojení vývoje vozu spolu s vývojem mobilních aplikací pro automobily ŠKODA. Výroba vozu je v této společnosti z velké části stále spojena s tradičními přístupy, které nevyhovují vývoji softwaru pro aplikace.

Tento návrh interní metodiky, by měl posloužit k zefektivnění celkového vývoje, jelikož mobilní aplikace jsou úzce svázané s náběhem nových vozů.

1 Analýza agilních a rigorózních metodik

Bakalářská práce se na začátku zabývá vymezením pojmů metodologie a metodika. Dále je popsáno rozdělení metodik a v neposlední řadě budou vybrané metodiky představené podrobněji. Tato kapitola by měla čtenáře uvést do hlavního tématu a říci, co jsou rigorózní a agilní metodiky.

1.1 Vymezení pojmů

Metodologie

Metodologií se rozumí vědní disciplína, která se zabývá tvorbou metodik. Často jsou pojmy metodologie a metodika zaměňovány. V anglickém jazyce, tomu tak dochází díky překladu. Metodologie určuje vhodnost a použitelnost nástrojů pro jednotlivé metodiky.

Metodika

Metodika vývoje představuje souhrn metod a postupů pro vznik vývojového softwaru. Avšak lze k vývoji také využít již předem hotové řešení, komponenty či služby. Alena Buchalcevová ve své knize Metodiky budování informačních systémů [1], metodiku definuje takto: *„Metodika vývoje definuje principy, procesy, praktiky, role, techniky, nástroje a produkty, používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení.“*

1.2 Rigorózní metodiky

Jako první jsou specifikované rigorózní, neboli tradiční metodiky. Tyto metodiky vznikly dříve než ty agilní. Avšak to neplatí pro všechny tradiční metodiky, některé vznikly až postupem času. Hlavním důvodem proč jsou takto označovány, je díky tomu, že využívají tradiční přístupy k vývoji softwaru. Velký důraz je kladen na perfektní dokumentaci a celkové dokumentování práce. Rigorózní metodiky jsou postavené především na pevně daných termínech a požadavcích od zákazníků. Dalším rysem je velká spousta rolí, na kterých se nachází vysoce specializovaní lidé, kteří jsou odborníci na svou práci, ale již nejeví o ostatní fáze vývoje.

Rigorózních metodik v dnešní době existuje již poměrně velké množství. Např. Vodopádový model, Spirálový model, Rational Unified Process, V-model, Inkrementální model, Evoluční model a další. Pro lepší představu jsou níže rozepsané alespoň tři nejznámější a nejpoužívanější rigorózní metodiky. Tyto metodiky byly u vzniku softwarového inženýrství a doposud si najdou v některých firmách své místo.

1.2.1 Vodopádový model (Waterfall model)

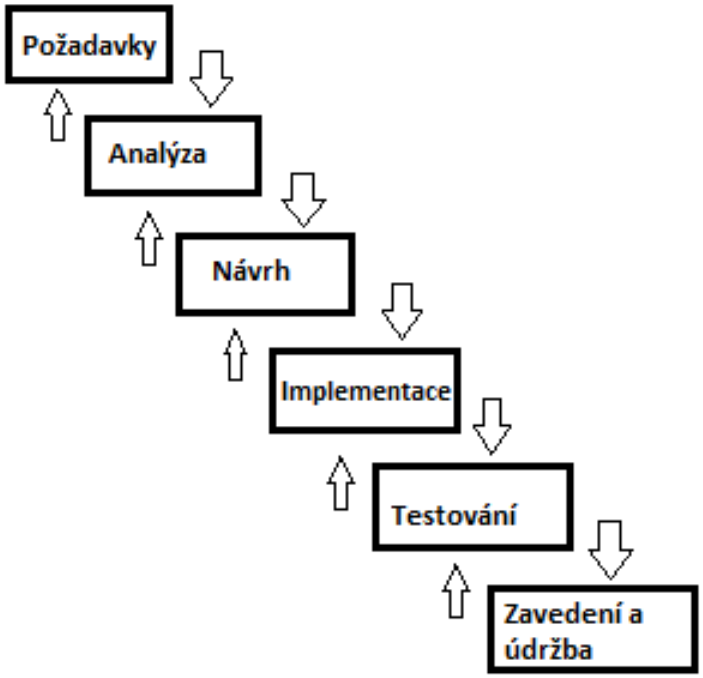
Vodopádový model je považován za jednu z nejstarších rigorózních metodik, která vznikla již v 70. letech 20. století. Byla to první ucelená metodika, která se v softwarovém inženýrství začala zdárně používat a díky svým principům a jednoduchosti se stala i velice rozšířenou a oblíbenou. Avšak není úplně vhodná pro všechny vývoje a nyní se používá i tam, kde by si našel uplatnění jiný typ metodiky.

Proces softwarového vývoje je ve vodopádovém modelu složen z několika fází. Tyto fáze jdou za sebou a nelze je přehazovat či dokonce přeskakovat, odtud také vznikl název vodopádový model. Proces vývoje je pouze jednosměrný. Pokud se ovšem zákazník ve fázi analýzy rozhodne přidat ještě další požadavek na specifikaci, může se proces vrátit o jednu fázi zpět. Dále musí však postupovat podle klasického modelu, a to po ukončení jedné fáze se proces přesune do další následující. Před začátkem každá fáze musí mít následující fáze všechny dokumenty schválené. Tím se také stává tento model velmi jednoznačným, snadno se řídí a plánuje.

Jednotlivé fáze vodopádového modelu:

- Specifikace požadavků
- Analýza
- Návrh
- Implementace
- Testování
- Zavedení
- Údržba

Pro ideální představu, jak vodopádový model vypadá, je zde schéma na kterém je tato metodika znázorněna.



Obrázek 1:Vodopádový model (Zdroj: Upraveno dle [16])

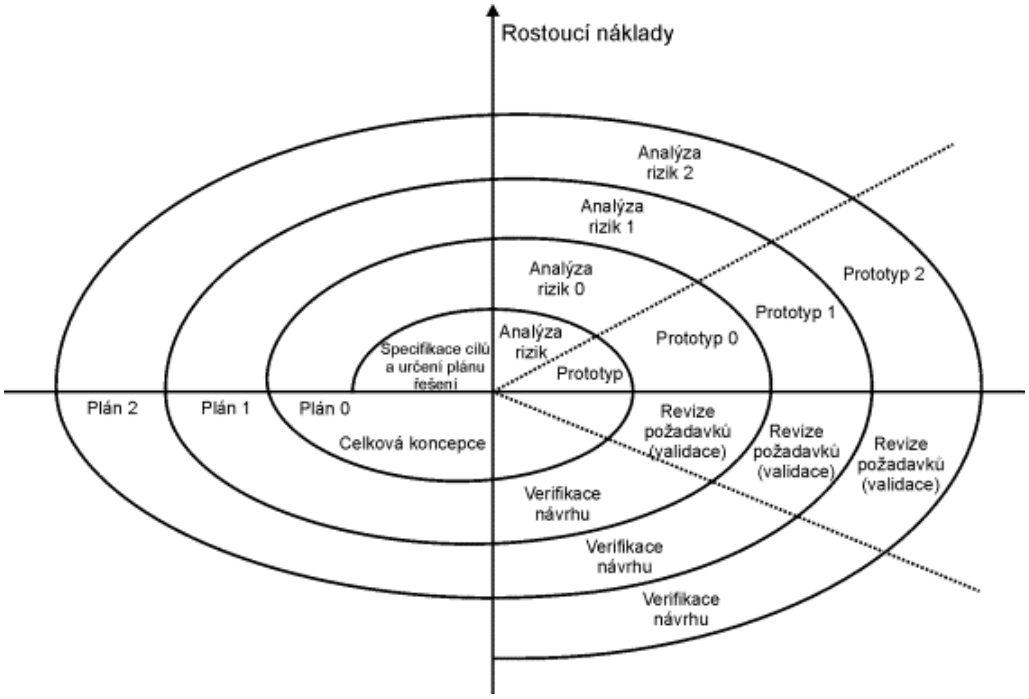
1.2.2 Spirálový model

Spirálový model se objevil roku 1985, definoval ho Barry Boehm. Spirálový model vznikl na popud softwarových inženýrů, kteří viděli nedostatky na vodopádovém modelu a snažili se je tímto zachytit a vytvořit již lepší životní cyklus softwarového produktu.

Hlavním znakem spirálového modelu je komplexnost. Tento model vychází přímo z waterfallu a přidávají se k němu dvě zásadní vlastnosti. Jedna z ní je iterace, druhá analýza rizik. V podstatě se jedná o to, že se na začátku cyklu nedělá úplná specifikace požadavků. Spirálový model se poučil a začíná pouze s rámcem architektury a s funkčností systému, podrobnosti se dodělávají postupně během cyklu. Analýza rizik probíhá během celého cyklu a tím dochází i k možnosti včas vyřešit problém, který by jinak mohl ohrozit průběh výsledku. Odhaleným rizikům je velice důležité určit možný výskyt a nebezpečnost. Spirálový model je definován čtyřmi iteracemi.

Každá iterace se skládá ze stejných fází:

- Definice cílů
- Analýza rizik
- Návrh řešení
- Ověření
- Vývoj
- Testování
- Plánování



Obrázek 2: Spirálový model (Zdroj: [20])

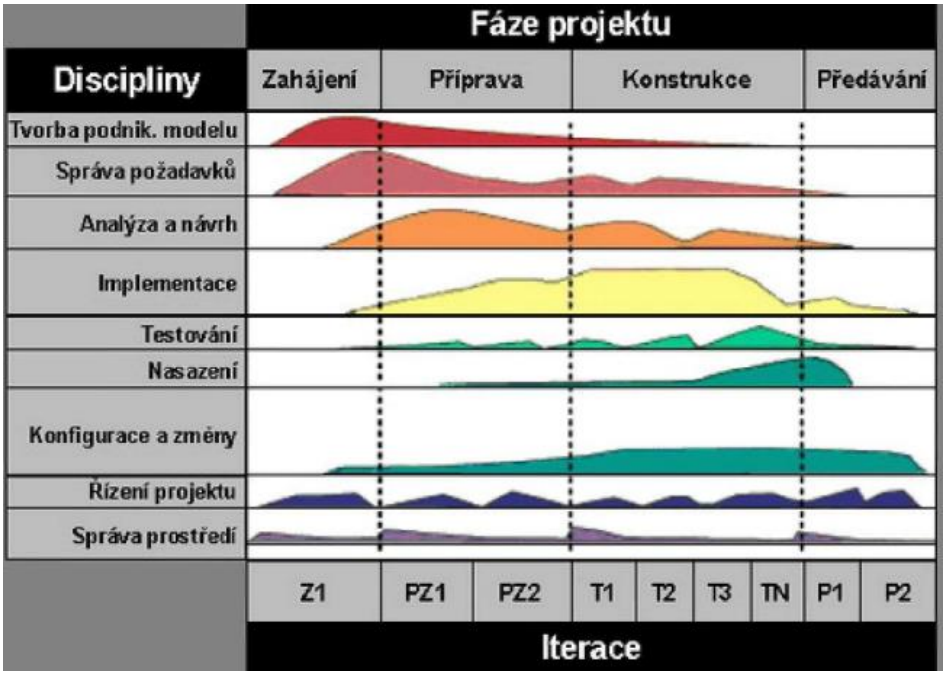
Během první iterace je důležité si identifikovat globální rizika a stanovit si východiska pro řešení. V druhé iteraci se klade důraz na specifikaci požadavků. Třetí iterace se zaměřuje na vytvoření detailního návrhu řešení a poslední čtvrtá iterace slouží k implementaci navrženého řešení a jeho testování. [1]

1.2.3 Rational Unified Process

Poslední metodika, která je práci rozepsána více dopodrobna, je Rational Unified Process (dále již jen RUP). Tento komerční produkt vytvořila společnost Rational Software Corporation. RUP je to objektově orientované metodika, která je velice rozsáhlá a podrobně propracovaná.

Při vývoji softwarového systému vychází RUP z 6ti základních praktik:

- Iterativní vývoj - objevení rizik již v průběhu, jednodušší práce se změnami, lepší komunikace se zákazníky, lze použít opakovaně.
- Správa a řízení požadavků - požadavky od zákazníka se v průběhu mohou měnit.
- Použití komponentové architektury - RUP nabízí šablony, architektonické styly a pravidla designu. Vývoj může být usnadněn znovupoužitím hotové komponenty.
- Vizuální modelování - použití UML zjednoduší komunikaci daného týmu, zvýší pravděpodobnost správnosti dohodnutého systému.
- Průběžné ověřování kvality - odstraňování problémů již během vývoje
- Řízení změn - nedochází k nesrovnalostem



Obrázek 3:Rational Unified Process (Zdroj: [18])

RUP definuje 4 základní elementy modelování, na nichž je postavený celý tento vývoj softwaru. Každý z elementů má odpověď na otázku. Prvním elementem jsou pracovníci (kdo). Pracovník není brán jako fyzická osoba, ale spíše je tím rozuměna určitá role. Každý pracovník má svou odpovědnost za daný meziprodukt. Druhým elementem jsou činnosti (jak). Každá činnost má určený konkrétní účel a je přesně definovaný vstupní a výstupní meziprodukt. Dalším elementem jsou meziprodukty (co). Meziprodukt je hmatatelný výsledek projektu. Posledním elementem jsou pracovní procesy (kdy). Pracovní proces neboli workflow určuje posloupnost činností a iterace mezi pracovníky.

1.3 Agilní metodiky

Druhým typem metodik jsou agilní metodiky. Význam slova „agilní“ definovali Zuzana Šochová a Eduard Kunce v knize Agilní metody řízení projektů [9] takto: *„Agilní je dynamický, rychlý, interaktivní, přizpůsobivý, iterativní, zábavný, hravý, rychle reagující na změnu… a jistě vymyslíme spoustu dalších synonym. Je to jiný způsob života, upřednostňující jiné hodnoty: reálný výsledek před striktními procesy, změnu před předem naplánovaným. Být agilní znamená žít agilní filosofii.“*

Agilní metodiky jsou v dnešní době mnohem využívanější a žádanější. Hlavním pilířem agilního přístupu je komunikace, spolupráce a připravenost na změnu. Zároveň tento přístup nemá striktní proces, ale také to není chaos, vše má svá pravidla. Důležitou roli hraje tým, který si sám určuje svá pravidla, podle kterých se všichni poté řídí. Pokud si sám tým určí pravidla, která jim vyhovují, znamená to, že se jim bude lépe pracovat. Tým je poté více efektivní, produktivní a dodávají i kvalitnější produkt v kratším čase. Agilní metodiky jsou zaměřené na optimalizaci funkcionalitu produktu, tak aby zákazník byl spokojený a dostal to, co potřebuje.

Agilní manifest

Základem agilních metodik je agilní manifest. Tento manifest vznikl roku 2001 v Utahu, kde se sešli odborníci ze softwarového inženýrství a vývoje softwaru. Celkem16 odborníků se dohodlo na čtyřech bodech, které vystihují celý agilní přístup.

Na webových stránkách Manifest Agilního vývoje software [13] je Agilní manifest definován takto:

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

- *Jednotlivci a interakce před procesy a nástroji*
- *Fungující software před vyčerpávající dokumentací*
- *Spolupráce se zákazníkem před vyjednáváním o smlouvě*
- *Reagování na změny před dodržováním plánu*

Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více.„

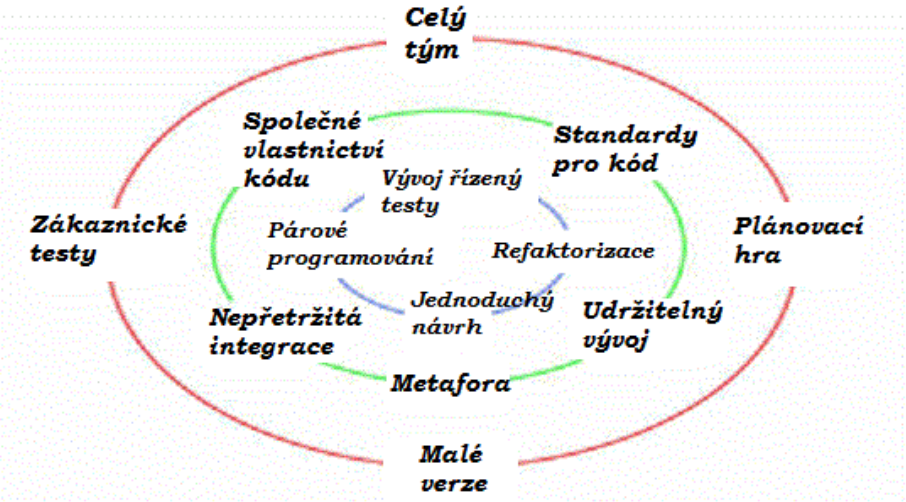
Agilní manifest obsahuje 12 principů, které jsou zde stručně představené. Časné a průběžné dodávání fungujícího softwaru zákazníkovi má jednu z největších priorit. Tento software se dodává v kratších intervalech, týdenní maximálně měsíční. Hlavním měřítkem dobře odvedené práce je fungující software. Tým, který pracuje agilně, musí být správně motivovaný, pracovat v příjemném prostředí, které pokryje jejich potřeby. Dalším důležitým faktorem je komunikace, nejefektivnější a nejúčinnější předávání informací je totiž pomocí osobní konverzace. To způsobuje, že se tým takto může častěji zamyslet nad tím, jak svou práci zefektivnit. Ovšem každodenní komunikace se netýká pouze týmu, ale i lidí z byznysu a vývoje. Jako celek tým rychleji reagují na nové požadavky od zákazníků již během vývoje, nikoliv až po dokončení softwaru. Agilitu zvyšuje pozornost výjimečnosti, designu a jednoduchost. Tým lidí, kteří jsou schopní se řídit sami, vytváří nejlepší architektury, požadavky a návrhy. Všechny tyto agilní procesy podporují udržitelné stálé trvalé tempo.

Přehled agilních metodik

Agilních metodik v dnešní době existuje již velké množství a stále vznikají nové. Nejznámější metodikou SCRUM se bude zabývat kapitola 3. Mezi další agilní metodiky se řadí Extrémní programování, Kanban, Lean Development, Crystal metodiky, Dynamic Systems Development Method, Feature-Driven Development, Adaptive Software Development a další. Níže jsou popsané ty nejznámější z nich.

1.3.1 Extrémní programování

Jedná se o jednu z nejznámějších agilních metodik, kterou vytvořili společně Kent Beck, Ward Cunningham a Ron Jeffries. Název Extrémní programování neboli Extreme programming (dále jen XP) je odvozený od toho, že mnohé principy z jiných metodik dovádí až do extrémů. V XP vše probíhá v mnohem kratších iteracích. Žádná iterace netrvá měsíc ani týden, ale jeden den, někdy pouze pár hodin. Jednotlivé fáze se neustále střídají: navrhnout, naimplementovat, otestovat, nasadit. Nejextrémnější novinkou jsou změny. Několikrát za den se mohou měnit požadavky, to způsobí například i několikrát denně zahodit kus naprogramovaného programu.



Obrázek 4:Praktiky Extrémního programování (Zdroj: [19])

Na obrázku výše jsou znázorněny všechny praktiky XP, které jsou zde rozepsané a vysvětlené. [1]

- Tým jako celek (Whole Team)
  - Tým se skládá se zákazníka (musí komunikovat s týmem každý den), testerů, programátorů, analytiků (pomáhají zákazníkovi definovat požadavky). Tým by měl být tvořen z generalistů, kteří napomáhají ke společnému cíli. Celý tým a projekt vede kouč.

- Plánovací hra (Planning Game)
  - 2 základní body: předvídá se, co bude součástí dodávky (plánování dodávky) a určuje se, co dělat dál (plánování iterace)

    - Plánování dodávky – během plánování dodávky zákazník specifikuje programátorům požadavky, ti odhadnout náročnost a zákazník podle toho vytvoří hrubý plán, který se s každou iterací upravuje a upřesňuje.
    - Plánování iterace – během plánování iterací si tým určí, co se bude dělat během dvoutýdenní iterace. Zákazník specifikuje, co během dané iterace chce implementovat, programátoři si rozdělí úkoly a určí jejich náročnost.

- Malé verze (Small Releases)
    - Verze jsou dodávány v krátkých intervalech. Vždy pro zákazníka přináší hodnotu. Dá se mluvit o nižší chybovosti, díky nepřetržitému testování.

- Zákaznické testy (Customer tests)
    - Úspěšná implementace se ověří pomocí akceptačního testu, který si zákazník specifikuje pro každý jeho požadavek.

- Metafora (Metaphor)
    - Metafora neboli společná vize fungování systému, kterou vytváří celý tým dohromady. Každý kdo se na vývoji podílí, musí systému rozumět a vědět jak má správně fungovat.

- Nepřetržitá integrace (Continuous Integration)
    - Každý je zodpovědný za svou část kódu. Integrace probíhá i několikrát denně.

- Společné vlastnictví kódu (Collective Ownership)
    - Na jednu část kódu dohlíží více lidí (párové programování a testování) tudíž se snižuje chybovost a zároveň zvyšuje kvalita kódu.

- Standardy pro psaní zdrojového kódu (Coding Standard)
    - Všichni dodržují stejné konvence pro psaní zdrojových kódů. Díky tomu kódu všichni bez problému rozumí a vyznají se v něm.

- Udržitelný vývoj (Sustainable Pace)
    - XP dodržuje týdenní pracovní dobu, dovolenou a maximálně 1 týden přesčasů.

- Jednoduchý návrh (Simple Design)
    - XP se drží co nejjednoduššího možného řešení.

- Párové programování (Pair Programming)
    - V XP se programuje v párech. Páry se během dne mění. Přenášejí si mezi sebou své znalosti, a tím vzniká kvalitnější software.

- Refaktorizace (Refactoring)
    - Refaktorizace je změna struktury beze změny chování. Cílem je zpřehlednění, zjednodušení a zajištění flexibility.

- Vývoj řízený testy (Test- Driven Development)
    - Před napsáním kódu, musí být napsán jednotkový test.

1.3.2 Crystal metodiky

Crystal metodiky není přímo jedna konkrétní metodika, ale označuje se tím celá „rodina“ metodik. Jejím tvůrcem je Alistair Cockburn. Metodiky se odlišují vhodností pro různě složité projekty a různě velké týmy. Každá metodika je pojmenována podle barvy (čirá, žlutá, oranžová, červená, hnědá, modrá a fialová). Čím je barva tmavší, tím je metodika mohutnější a je vhodnější pro rozsáhlejší projekty. Všechny metodiky z Crystal jsou postaveny především na komunikaci a lehkosti produktu. Konkrétní metodika se volí podle tří kritérií.

- Velikost týmu - kolik lidí se bude podílet na vývoji.
- Důležitost systému - jak kritický systém může být, jaký dopad by mohlo mít selhání systému.
- Priorita projektu

		Crystal metodiky				
		Čiré	Žlutá	Oranžová	Červená	Hnědá
Důležitost produktu	Ohrožení života	L6	L20	L40	L80	L200
	Ohroženy velké částky	E6	E20	E40	E80	E200
	Ohroženy menší částky	D6	D20	D40	D80	D200
	Ohrožen komfort	C6	C20	C40	C80	C200
		1 to 6	7 to 20	21 to 40	41 to 80	81 to 200
		Počet lidí zapojených do projektu				

Obrázek 5: Crystal metodiky (Zdroj: [14])

1.3.3 Kanban

Kanban vznikl v Japonsku přímo v chrámu. Každý, kdo vcházel do chrámu, obdržel lístek, a když odcházel, lístek vrátil a dostal ho další návštěvník. Počet lístků byl omezený, tudíž v chrámu nikdy nemohlo dojít k přeplnění. Byla jasně daná kapacita chrámu a ta se tímto způsobem nedala překročit. Později se tento princip začal používat i v továrnách.

Z toho vyplývá, že Kanban není systém pro řízení procesu nýbrž metoda pro zlepšení procesů. V podstatě vychází z Lean metodiky. Kanban nám nic nenařizuje. Všechno si sám zvolí a rozhodne tým. Přesto existují tři principy, které se musí dodržet.

- Omezit rozpracovanou práci (Work in progress) – zkrátit čas realizace (zvyšuje se schopnost dodávat pravidelně funkční produkt), zvýšení kvality celého týmu
- Minimalizovat čas průchodu – lead time
- Vizualizovat workflow – tabule s jednotlivými úkoly, tabule je rozdělena do několika procesů, problémové úkoly se identifikují každý den, úkoly se posouvají podle aktuální situace

Kanban se v dnešní době využívá nejčastěji pro maintenance týmy, kde nelze úkoly rozdělit do jednotlivých iterací. Tato metodika se dá kombinovat i se SCRUM.

1.3.4 Lean Development

Autorem metodiky Lean Development je Robert Charette. Lean v překladu znamená štíhlý, a to se promítá v celé metodice. Cílem této metodiky je vytvoření softwaru tolerantního ke změnám s třetinovou lidskou prací, s třetinovým časem, s třetinou investic do nástrojů a metod, s třetinovým úsilím přizpůsobit se novým podmínkám. [1]

Metodika Lean obsahuje 10 pravidel pro vývoj softwaru:

1. Odstranit zbytečné – odstranit vše nadbytečné, co již nepřináší žádnou hodnotu
2. Minimalizovat zásoby – mít co nejméně meziproduktů
3. Maximalizovat tok – snížit čas pro vývoj
4. Vývoj tažený poptávkou – vývoj začíná až po poptávce, přizpůsobení se požadavkům zákazníka
5. Pracovníci s rozhodovací pravomocí – rozhodují pracovníci i na nižších pozicích

6. Uspokojení zákazníka – vždy uspokojit zákazníkovy požadavky
7. Zavést zpětnou vazbu
8. Odstranit lokální optimalizaci – pokud probíhají změny, není důvod optimalizovat
9. Partnerství s dodavateli – dobré vztahy s dodavateli, popřípadě přemýšlet o nákupu komponent
10. Neustálé zlepšování – vytvoření kultury pro zlepšování

Tato metodika je méně používaná a to díky tomu, že je abstraktnější než ty ostatní. Nejznámější firmou, která využívá Lean je Toyota. Vyrábí příslušný díl, až když je potřeba. Tedy řídí výrobu takzvaným systémem tahu. Na rozdíl od ostatních, kteří vyrábí vše do zásoby.

2. Porovnání metodik pro vývoj v automobilovém průmyslu

Tato kapitola se zabývá hlavními rozdíly mezi rigorózními a agilními metodikami. Dále zde je porovnání a srovnání těchto přístupů. Konec kapitoly se zaměřuje na využití metodik, tedy, pro které projekty se jaký přístup hodí.

2.1 Rozdíly mezi rigorózními a agilními metodikami

Rigorózní a agilní metodiky jsou založené na celkově odlišném chápání procesu vývoje softwaru. Jsou si navzájem svým opakem. Tyto dvě skupiny metodik mají rozdílné předpoklady pro zavedení procesu, dále mají úplně jiný obsah daných metodik a kladou důraz na odlišné části procesu.

Rigorózní metodiky jsou zpracované velice dopodrobna, jsou přísné, striktní a formální. Tyto metodiky mají jasně dané procesy vývoje softwaru, dále přesně specifikovanou posloupnost činností a v neposlední řadě i dokonalou dokumentaci každého kroku. Na začátku celého procesu se zaměřují na důkladnou analýzu a propracovaný návrh.

Agilní metodiky tvrdí, že proces nelze přesně popsat, jelikož by měl být pružný a schopný rychle reagovat na přicházející změny. Agilní přístup uplatňuje rychlý vývoj a úpravy na základě komunikace se zákazníkem. Dalším specifikem pro agilní metodiky je tým. Agilní tým je složený z maximálně 8 vývojářů, kteří mezi sebou každodenně komunikují. Každý vývojář musí mít dokonalé znalosti a zkušenosti.

Pro jednodušší představu základních rozdílů mezi rigorózními a agilními metodiky jsou rozděleny do tří oblastí.

1. Předpoklady
    - Rigorózní metodiky – software procesy lze podrobně popsat, požadavky od zákazníka lze specifikovat dopředu
    - Agilní metodiky – software procesy nelze popsat, požadavky od zákazníka nelze detailně specifikovat hned na začátku

2. Obsah
    - Rigorózní metodiky – detailně specifikované procesy a činnosti
    - Agilní metodiky – obecná pravidla, praktiky a principy

3. Použití
    - Rigorózní metodiky – velké a standardní projekty
    - Agilní metodiky – menší týmy, nové projekty

Další rozdíly, které mezi metodikami jsou, budou představeny v tabulce níže.

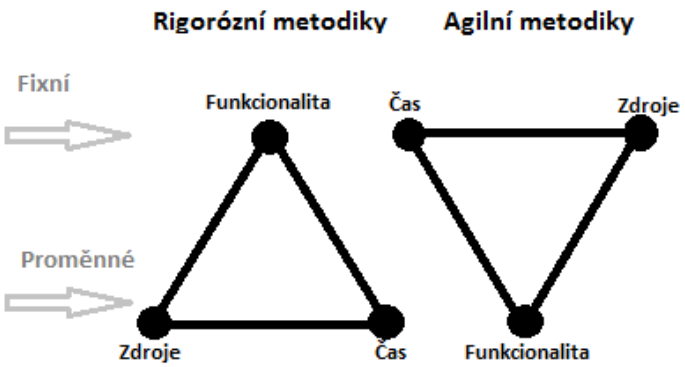
Tabulka 1: Porovnání rigorózních a agilních metodik (Zdroj: [2])

Hledisko	Rigorózní metodiky	Agilní metodiky
Náplň metodiky	Procesy, pohlíží na lidi jako na sekundární faktor	Praktiky, zaměřují se na „tacit“ znalosti, chápou lidi jako klíčové faktory úspěchu
Podrobnosti metodiky	Procesy a činnosti jsou popsány velmi podrobně	Sotva dostatečná metodika, zaměřuje se na činnosti, které vytvářejí hodnotu
Kvalita	Zaměření na kvalitu procesů a předpoklad, že kvalitní procesy povedou ke kvalitnímu výsledku	Zaměření na hodnotu pro zákazníka a vysokou kvalitu produktu
Předvídatelnost	Předpokládá předvídatelnost budoucnosti, důraz na anticipaci (sběr požadavků předem, plánování předem)	Předpokládá nepředvídatelnost budoucnosti, důraz na adaptaci na změny (přírůstkové shromažďování požadavků, plánování pro iteraci)
Změny	Změny podléhají řízení změn a je snaha změny minimalizovat	Změny jsou vítány, zákazníci mohou přehodnotit své požadavky
Definovatelnost procesu vývoje softwaru	Vývoj softwaru je definovaný proces, který je možné bez problému opakovat	Vývoj softwaru je empirický proces, nemůže být konzistentně opakován, ale vyžaduje monitorování a adaptaci
Hodnota pro zákazníka	Přílišné zaměření na procesy, ne na výsledky pro zákazníka	Nejvyšší prioritou je uspokojovat zákazníka
Participace zákazníka na projektu	Jen v počátečních a koncových fázích, po podpisu dokumentu specifikace požadavků řízení přebírá tým technologických pracovníků	Přesun nositele řízená z týmu na zákazníka, zákazník je řídícím subjektem během celého projektu, při každé iteraci zákazník může měnit priority funkcí
Rozsah řízení	Vývojáři se snaží do systému zabudovat budoucí požadavky	Pouze požadované funkce, požadavek minimalizace
Vztah zákazníků a vývojářů	Zajištěn smluvně, nedůvěra	Důvěra a spolupráce
Lidský faktor	Sekundární, dokumentačně zaměřené procesy se snaží vykázat lidi do role zaměnitelné součástky	Primární, využívá individualit a silných stránek lidí
Kvalifikace lidí	Stačí standardní jedinci	Důraz na schopnosti, znalosti a dovednosti lidí
Specialisté versus generalisté	Požadavek úzké specializace lidí	Spíše generalisté než specialisté, sdílení znalostí v týmu, týmové řešení problému

Způsob řízení	Na základě nedůvěry, direktivní řízení, kontroly	Vůdcovství a spolupráce, je formováno na důvěře a respektu
Význam programování při vývoji SW	Důraz na architekturu, požadavky a návrh, kódování a testování jsou chápány jako činnosti s nízkou hodnotou	Důraz na programování jako činnost přinášející hodnotu
Jednoduchost	Spíše složité řešení, které se snaží obsáhnout i budoucí požadavky	Důraz na jednoduché řešení, žádné zabudování budoucích požadavků
Jednoduchá versus složitá pravidla	Metodiky se snaží popsat vše, s čím se může vývojový tým setkat	Obsahují generativní pravidla- minimální množinu věcí, které je třeba dělat ve všech situacích
Modelování	Velký důraz na modelování, zejména modelování předem	Agilní modelování, při modelování nejde o model jako takový, ale o akt modelování, smyslem modelování je komunikace
Forma komunikace	Převážně písemná	Důraz na komunikaci tváří v tvář
Dokumentace	Rozsáhlá dokumentace	Podstatná není dokumentace, ale pochopení
Způsob vývoje	Spíše vodopádový, případně iterativní a přírůstkový s dlouhými iteracemi	Přírůstkový vývoj s velmi krátkými iteracemi
Ekonomika	Zdroje bývají proměnou veličinou, která zpravidla roste	Snaha vždy realizovat nejvyšší hodnotu z daných peněz, cílem je hodnota pro zákazníka, ne perfektní systém, hodnota je kombinací funkcí produktu, které odpovídají potřebám zákazníka v určitý čas a za určitou cenu

2.2 Porovnání metodik

Základní porovnání rigorózních a agilních metodik je vidět na třech nejdůležitějších položkách. Tyto položky jsou znázorněné na obrázku níže.



Obrázek 6: Porovnání rigorózních a agilních metodik (Zdroj: Vlastní)

Pro rigorózní metodiky je fixní položkou funkcionalita. Z toho vyplývá, že všechny požadavky musí být pevně specifikované hned na začátku a nelze je již měnit. Proměnnými veličinami jsou čas a zdroje. Čas a zdroje jsou odhadnuty na začátku procesu, ovšem tyto veličiny se mění podle průběhu vývoje. Ve většině případů se během vývoje čas prodlužuje a zdroje zvyšují. Agilní metodiky mají rozložení hodnot zcela obráceně. Fixní hodnoty jsou zde čas a zdroje, které se v agilních metodikách již nemění na rozdíl od funkcionality. Ta je zde proměnná. A lze v případě nutnosti funkcionalitu kvůli času či zdrojům omezit.

Porovnání metodik agilních s rigorózními lze vyčíst již z podkapitoly výše. Zde jsou rozepsané do pěti zásadních bodů: Dokumentace, Iterativní vývoj, Komunikace, Spolupráce a Testování.

Dokumentace

Agilní metodiky jsou přesvědčené, že není potřeba mít dokonalou dokumentaci na celý projekt. Vycházejí z toho, že je důležitější a spolehlivější jako nositel informace zdrojový kód. Tradiční metodiky zastávají názor, že vše musí být podrobně zaznamenáno do dokumentace pro lepší přehlednost. Ovšem to zabírá více času.

Iterativní vývoj

Běh softwarového vývoje, podle agilních metodik, se skládá z krátkých iterací. Interval těchto iterací se liší podle velikosti daného projektu, nejčastěji se pohybuje kolem měsíce až týdnů. Tyto krátké iterace mají své výhody v rychlé zpětně vazbě od zákazníka, který tak může dát dříve najevo, že s nějakou implementovanou částí nesouhlasí. V tradičních metodikách jsou iterace také, ovšem v delších časových intervalech.

Komunikace

Osobní komunikace je pro agilní metodiky velice důležitá. Proto se tyto metodiky snaží, aby lidé v týmu mezi sebou měli dobré vztahy. Pro tyto týmy je nutná každodenní osobní komunikace, při které si navzájem přidávají informace, zkušenosti, a zároveň řeší společně nastalé problémy. Tradiční metodiky dávají přednost činnostem vývoje.

Spolupráce s uživateli

Agilní týmy oproti těm rigorózním komunikují s uživateli během celého projektu. Získávají tak od nich cenné informace. Rigorózní metodiky s uživateli komunikují spíše na začátku procesu a až po ukončení vývoje během akceptace.

Testování

Jelikož jsou v agilních metodikách kratší iterace, je častěji dodáván produkt, který má vždy implementované další požadavky. Je také samozřejmě potřeba tento produkt průběžně testovat. Toto testování odhaluje rychleji problémy, které se okamžitě mohou řešit. Rigorózní metodiky mají celý proces rozdělený do etap. Etapa testování následuje až po etapě implementace.

2.2.1 Využití metodik

Obecně lze říci, že agilní metodiky se dají využít pro projekty, které nemají na svém začátku zcela jasné požadavky. Také na projektech, u kterých lze očekávat, že se v průběhu budou tyto požadavky ještě měnit.

Pokud je projekt s přesně specifikovanými požadavky, kde nejsou změny ani dovolené, měl by se správně využít tradičních metodik. Agilní metodiky by v takovémto případě nefungovaly.

3. Metodiky pro výrobu vozu ve ŠKODA Auto

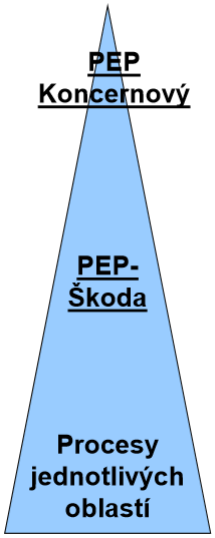
Tato kapitola objasní jaké metodiky se ve ŠKODA Auto pro vývoj automobilu používají. První část kapitoly tuto metodiku podrobně popíše. Zároveň budou popsány dva typy této metodiky. Existují dva různé metodiky PEP. Využívanější je PEP s tradičním přístupem, druhý typ se využívá agilním přístupem.

3.1 Popis procesu vývoje vozu

Společnost ŠKODA Auto pracuje s metodikou, která se označuje PEP. Tato zkratka skrývá celý název „Proces vzniku výrobku“. Na tuto metodiku PEP je v ŠA vytvořena příručka, podle které se řádí všichni pracovníci této organizace.

3.1.1 Pracovní procesy jednotlivých oblastí

Na úplný začátek je zde zobrazen obrázek, kde je přesně vidět jak na sebe navazují a jsou závislé jednotlivé části. Úplně na špici je PEP Koncernový. Koncernový PEP má jeden důležité a zásadní úkol. Tím je stanovení jednotné, závazné procesy PEP s vysvětlením milníků. Uprostřed se nachází PEP- Škoda, tato část popisuje závazné procesy PEP ve ŠA na základě koncernového zadání jako základ pro jednání grémií. Poslední jsou Procesy jednotlivých oblastí- dílčí procesy a odpovědnosti jednotlivých oblastí v rámci zadání značky jako základ pro projektovou práci.



Obrázek 7: PEP rozložení (Zdroj: [10])

Posloupnost procesů jednotlivých oblastí: [10]

1. **Spojení souvisejících činností procesu PEP ŠKODA do pracovních fází**
   - Zobrazení a posloupnost znázornění odpovídá činnostem oblasti.

2. **Doplnění dodatečných činností**
   - Zobrazení specifických, dodatečných činností. Doplnění dalších milníků, které přísluší procesu.

3. **Vyznačení odpovědnosti**
   - Stanovení odpovědnosti nebo spolupůsobení a přiřazení k příslušnému znázornění.

Pro lepší představu celého procesu, je zde obrázek s přehledem procesů jednotlivých oblastí.

- Produkt
- Technický vývoj
- Výroba
- Nákup
- Marketing/Odbyt
- Finance
- Zajištění kvality

Každá oblast ze Škoda PEP obsahuje různé procesy. Jelikož se jedná o interní záležitosti společnosti, budou zde vypsány jednotlivé procesy pouze okrajově.

Produkt

Zadání projektu, Finanční řízení projektu, Projektová organizace Pilotní hala, Schvalení, PKO, Moduly, Varianty, Prototypy, Změnové řízení, Inovace, CO2, Zákaznický přínos, Napojení na zákazníka, Struktura produktu, Pozice na trhu, Plánování množství, Plán oceňování, Investice, Umístění výroby, V-časy, Uvolňování, Řízení náběhu.

Technický vývoj

Technický vývoj je rozdělen do 14ti oblastí, které obsahují další procesy. Oblasti – Organizace projektu vozu, Produktbeeinflussung, Zadání, Analýza konkurence, Vliv produktu na zákazníka a trh, Vliv produktu na výrobu, Popis vozu, Vývoj koncepce, Vývoj

povrchových ploch, Konstrukce a uvolňování, Zajištění, Předprodukce a stavba prototypů, Integrace do vozu, Požadavky kvality.

Výroba

Stejně tak je rozdělena i Výroba, ta má 10 oblastí. Zadání projektu, Finanční řízení projektu, Organizace projektů vozu, Náklady výroby, Vliv produktu, Výrobní a logistické procesy, Integrace do vozu, Kvalita, Nákup, Předsérie a stavba vozu.

Nákup

Inovace, Umístění výroby, Finanční řízení projektu, Zadání projektu, Moduly, Plán oceňování, Kusovník, Investice, Projektová organizace Pilotní hala, CO2, Forward Sourcing, PKO, Nákup zařízení, Uvolňování, Řízení nakupovaných dílů, Řízení náběhu, Disponibilita dílů, Životní prostředí, Plánování množství.

Marketing / Odbyt

Inovace, Zákaznický přínos, CO2, Zadání projektu, Umístění na trhu, Napojení zákazníka, Plánování objemů, Struktura produktu, Plán oceňování, Projektový organizace Pilotní hala, Odbytové náklady, Koncepce produktu, Popis modelu, After Sales, Objednávání vozů, Komunikace, Řízení náběhu.

Finance

Finance mají 11 oblastí. Finanční řízení projektu, Výsledek výrobku, Rozsah, Materiálové náklady, Personální náklady výroby, Vstupní údaje – Řízení projektu, Marketing/Odbyt, Technický vývoj, Nákup, Zajištění kvality, Výroba.

Zajištění kvality

Zadání projektu, Finanční řízení projektu, Projektová organizace Pilotní hala, Kvalitativní cíle projektu, Spokojenost zákazníka, Garance, Požadavky kvality, Metody kvality, Inovace, Design, Koncepční a výrobní technika, Servisní technika, Závodový a zkušební koncept, Schvalování, Výběr dodavatele, Zajištění, Uvolňování, Vzorkování, Zajištění kvality, Řízení náběhu, Změnové řízení.

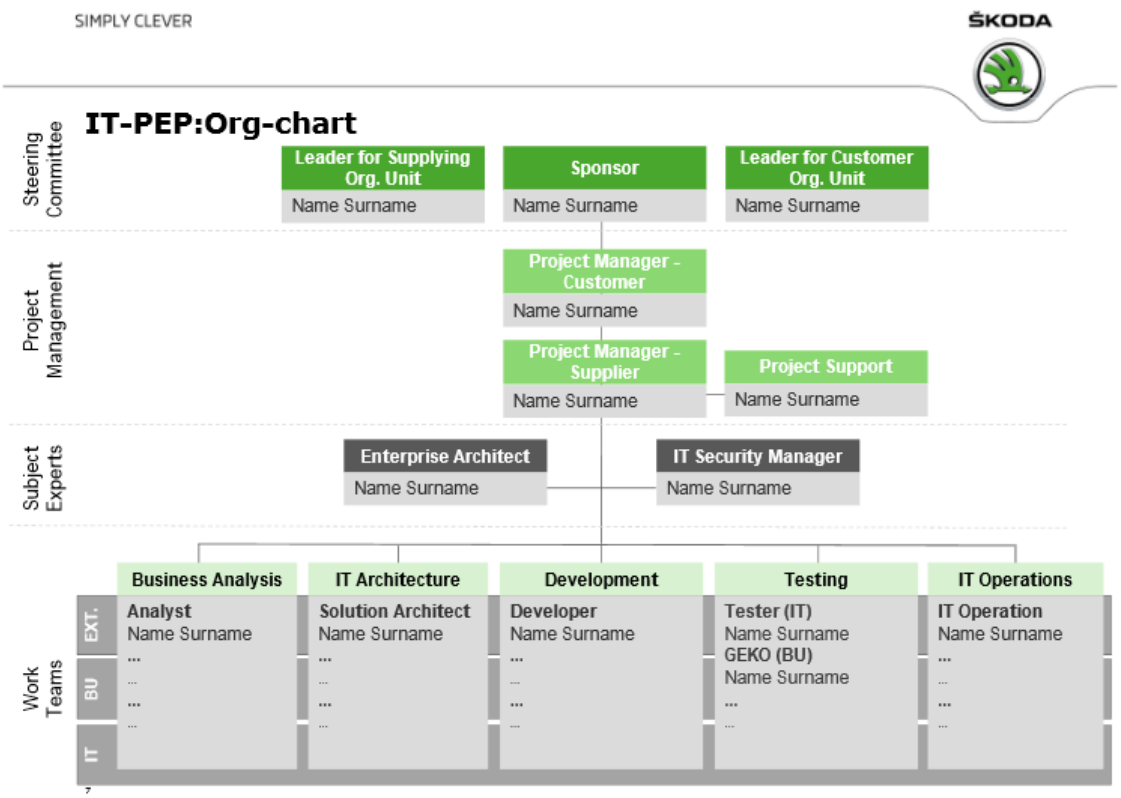
3.2 Analýza tradiční přístupu PEP

Nynější stav se opírá o metodiku IT-PEP. Tato metodika IT-PEP definuje tradiční typ řízení projektů a to Vodopádový model neboli Waterfall. Tato rigorózní metodika je nejpoužívanější metoda řízení projektů ne jen ve ŠKODA Auto, ale v celém Volkswagen koncernu (VW).

3.2.1 Organizační struktura projektu

Organizační struktura tradičního projektového řízení Waterfallu podle IT-PEP je rozdělena na čtyři úrovně.

- Řídící výbor projektu (Steering Committee)
- Projektový management
- Experti (Subject Experts)
- Pracovní týmy (Work Teams)



Obrázek 8: Schéma organizační struktury IT-PEP waterfall (Zdroj: [11])

První úroveň Řídící výbor má minimálně tři členy – sponzora, zákazníka a dodavatele. Řídící výbor se účastní práce na projektu minimálně. V těchto výjimečných případech má za úkol odsouhlasit nebo změnit důležité skutečnosti, být informován nebo provést obecně zásadní rozhodnutí.

Další úrovní je část projektového managementu. Jsou zde dva projektoví manažeři. Jeden zastupuje zákazníka a druhý dodavatele. Projektový management je v dennodenním styku se všemi lidmi z projektu, tím se také stává hlavní složkou této organizace.

Třetí vrstvou organizační struktury jsou Experti. Tato vrstva se zařadila kvůli důležitým a vysoce kvalifikovaným poradám. Pro jejich odbornost poskytují specializované poradenství a schvalují některé dokumenty. Jejich práce není potřeba na projektu každý den.

Poslední organizační úrovní jsou pracovní týmy. Tyto pracovní týmy provádějí realizační technické činnosti na projektu. Činnosti se odvíjejí podle daných profesí.

3.2.2 Projekty v IT-PEP

Metodika IT-PEP pro velké projekty definuje projektové fáze, hlavní milníky, sub-milníky a rozdělení úkolů a odpovědností. V hlavních milnících jsou definovány cíle dodávky a dokumenty požadované projektovou metodikou. IT-PEP obsahuje také Harmonogram činností, který obsahuje základní údaje, KDO dělá CO a KDY. Pro vývoj vozu je charakteristická přímá a průběžná komunikace mezi všemi zúčastněnými stranami.

Skupinové milníky musí být vždy ze standardu Group IT-PEP. Pro každou značku jsou závazné. Pokud na projektu pracuje více značek, celý projekt se řídí podle milníků značky, která je odpovědná za projektové řízení.

Dokumentace během celého projektu je zde velice důležitá. Některé dokumenty jsou povinné, některé volitelně. Pro ulehčení práce, ale i pro přehlednost existují předem připravené šablony. Pokud dokumenty mají stejnou strukturu, je jednoduší se v nich pro všechny zúčastněné vyznat.

Při realizaci velkého a malého projektu jsou rozdíly, které jsou rozepsány níže.

Velký projekt podle IT-PEP

Celá realizace velkého projektu podle IT-PEP je rozdělena do sedmi fází, které jsou znázorněny na obrázku níže. Jedná se o takzvaný fázový model. Každá fáze projektu má specifické zaměření, cíle a činnosti v životním cyklu projektu a obsahuje odlišné činnosti pro různé části projektového týmu. Každá fáze je zakončena hlavními milníky. Kromě těchto hlavních existují i sub milníky. Pokud jsou hlavní milníky provedeny správně, může se pokračovat do další následující fáze.

Hlavní milníky pro jednotlivé fáze:

1. Schválení inicializace projektu
2. Schválení objednávky
3. Schválení konceptu
4. Schválení designu
5. Schválení řešení
6. Zahájení provozu
7. Ukončení projektu

Obrázek 9: Fázový model pro velké projekty (Zdroj: Upraveno dle [11])

Malé projekty v IT-PEP

Celková realizace malých projektů je rozdělena podle fázového modelu na rozdíl od velkých projektů pouze do čtyř fází. Tyto fáze jsou znázorněny na obrázku níže. Hlavní a sub milníky existují i pro malé projekty. Bez jejich splnění nelze pokračovat do další fáze.

Hlavní milníky pro jednotlivé fáze:

1. Schválení objednávky
2. Schválení konceptu
3. Schválení řešení
4. Ukončení projektu

Obrázek 10: Fázový model pro malé projekty (Zdroj: Upraveno dle [11])

3.3 Analýza agilního přístupu PEP

Druhým typem metodiky, které se využívá ve ŠKODA je ten samý PEP jako je zmíněn v předchozí kapitole, akorát vychází z agilních přístupů. Důvodem vzniku jsou rostoucí požadavky, zvýšení problémových oblastí ve zpracování projektů.

Jelikož se jedná o koncern, není tento přechod nejjednodušší. Pro ŠKODA je důležité, aby agilní metodiky vycházely z agilního standardu VW. Tento VW standard je postaven na metodě Prince2 DSDM agilního frameworku. Novinkou je využití metodiky SCRUM, která dopomáhá hladkému běhu samotného vývojového týmu.

Agilní přístup se nehodí, ale za každých okolností. Níže je popsáno několik příkladů, kdy se však hodí tento přístup využít a má své bonusy.

- Pro vývojové projekty s iterativním vývojem
- Pro projekty s většími riziky v rozpočtu a krátkou dobou dodání
- Pro flexibilní plánování v průběhu projektu podle výsledků dosahovaných přímo v samotném procesu vývoje na projektu a také podle náročnosti
- Pro prostředí, kde je vhodná a potřebná úzká aktivní vazba mezi businessem a vývojovým týmem

Projekty, které se řídí agilní metodikou ve ŠKODA Auto, jsou rozděleny do dvou skupin:

- Malé projekty (do 250 tis. euro)
- Velké projekty (nad 250 tis. euro)

Jedním z hlavních problémů proč je těžké zavádět agilní metodiky do společnosti, kde se využívají dlouhou dobu pouze tradiční metodiky je neochota lidí se přizpůsobit změnám. Ve ŠKODA Auto je problém na straně striktně daných postupů, procesů a perfektně vypracovaných specifikacích a dokumentací, které jsou důležité pro každý projekt. Avšak i agilní metodiky si zde začínají pomalu nacházet své vlastní místo.

4. Agilní metodiky v oblasti vývoje mobilních aplikací

Tato kapitola se zabývá popsáním, jak funguje agilní a tradiční přístup přímo v odvětví pro vývoj mobilních aplikací ve ŠKODA Auto. Nejprve je dopodrobna popsána jedna z nejznámějších agilních metodik a to SCRUM Development Process a možné nástroje pro využití agilních metodik, především JIRA. V poslední části této kapitoly je navržen postup, který by bylo dobré využívat při vývoji mobilních aplikací v ŠA.

4.1 SCRUM Development Process

Autory agilní metodiky SCRUM Development Process (dále jen SCRUM) jsou Ken Schwaber, Jeff Sutherland a Mike Beedle. Poprvé ji popsali roku 2002 v knize Agile Software Development with SCRUM. Název si tato metodika získala podle termínu z rugby „mlýn“, jedná se o strategii, jak dostat míč do hry, odtud tedy název SCRUM.

4.1.1 Charakteristika metodiky

SCRUM metodika je zaměřená především na řízení projektu. Tato metodika musí být flexibilní a iterativní. Velký důraz je kladen na spolupráci a komunikaci, tu zajišťuje se zákazníkem Product Owner, Scrum Master, nesmírně důležitá je ale i komunikace uvnitř samoorganizovaného týmu. Dalším znakem SCRUM metodiky je vynikající um reagovat na stále se měnící požadavky.

Na začátku životního cyklu Product Owner definuje cíle produktu, které sesumíruje požadavky do Produkt Backlogu, podle priorit. Vývoj je rozdělený do několika iterací, tedy Sprintů. V každém Sprintu si Development tým náplánuje práci (vznikne Sprint Backlog), kterou musí během daného Sprintu stihnout dokončit. S tím jim pomůže i Scrum Master. Během Sprintu probíhá velká spousta výstižných meetingů – Sprint Planning, Stand-Up, Sprint Review a Sprint Retrospective. Jednotlivé role a meetingy, zde zmíněné, jsou vysvětleny níže. Produkt je hotový, pokud jsou splněné cíle, které se plní v určitém počtu Sprintů.

4.1.2 Role

Vývojový tým SCRUM metodiky je sestaven z šesti rolí. Každá role má své pravomoci a povinnosti, které jsou uvedeny níže.

Scrum Master

Scrum Master je zodpovědný, že metodika SCRUM bude srozumitelná a přijatelná pro všechny členy týmu. Toto získá tím, že Scrum tým porozumí a bude se řídit podle teorie, praktik a pravidel SCRUM metodiky. Jednou z hlavní činností Scrum Mastera je komunikace s vývojáři, ale i s managmentem a v neposlední řadě i se zákazníkem. Pokud se vyskytnou nějaké nesrovnalosti či problémy, které brání v pokračování vývoje, jeho úkolem je najít příčinu a co nejrychleji přijít na nápravu. V tom může požádat o pomoc člena z týmu, který je v problematice více zainteresován. A snad tou nedůležitější činností Scrum Mastera je dovést tým ke zdárnému cíli, především správným vedením a také motivováním týmu.

Product Owner (Vlastník produktu)

Product Owner neboli vlastník produktu je zodpovědný za celý projekt a práci vývojového týmu. Zastupuje všechny zúčastněné strany. Nejdůležitějším úkolem je definovat vize projektu, a proto, aby se tak mohlo stát, je důležité, aby porozuměl přesně produktu. Dále určuje priority funkcionalit, které se budou dělat dříve a dokonce, které se musí úplně vypustit. Z větší části se nachází u zákazníka, pro lepší pochopení jeho potřeb.

Development Team (Scrum tým)

Scrum tým je samo-organizující projektový tým, jeho úkolem je splnit všechny cíle. Na začátku musí tým odhadnout pracnost a čas, který bude potřeba k implementaci každé z funkcionalit. Development tým se skládá z 3-9 členů, každý člen týmu by měl být zastupitelný. Tento tým si sám určuje, kdo na jaké části bude pracovat, kdo komu bude pomáhat a tak dále. Zkrátka jde o to, aby tým spolupracoval, a tím se stává i mnohem efektivnějším.

Customer (Zákazník)

Zákazník ve SCRUM hraje důležitou roli, během celého vývoje. Určuje priority, sleduje změny a funkcionality. V případě nějakého problému, je dobré, když zákazník vidí do backlogu. Tím se zajistí, že má lepší přehled a dokáže lépe pochopit vzniklé problémy týmu.

Management

Management disponuje pravomocí pro konečná rozhodnutí o projektu při uzavírání smluv. Také dohlíží na dodržování standardů a konvencí. Mezi důležité činnosti také patří vytvoření příjemného prostředí pro scrum tým a v neposlední řadě také jeho podpora.

4.1.3 Základní pojmy a artefakty

I SCRUM používá své speciální pojmy a artefakty, které jsou typické právě pro tuto metodiku. Pro úplný začátek je potřeba objasnit User story. Dále SCRUM obsahuje tři artefakty, o to jsou důležitější a je potřeba jim správně porozumět. Mezi artefakty se řadí Product Backlog, Sprint Backlog a Increment.

User story

Zjednodušeně lze říci, že User story (neboli uživatelský příběh) je uživatelský popis toho, co by systém měl dělat. Nikoliv toho, jak by to měl dělat. To uživatele vůbec nezajímá. Jednotlivé funkcionality systému tedy budou ztotožněny s jednotlivými úkoly jednotlivých rolí tak, jak je chápe zákazník. Ovšem i přes to se musí dodržovat rozumný formát. [6]

Popis tří základních částí:

- Role – každá role, kterou může uživatel při vývoji nového softwaru mít, má své úkoly. Je důležité, aby tyto úkoly byly správně spárovány s danými rolemi. Z toho vyplívá, že role a příběh jsou mezi sebou provázány. Tudíž na začátku příběhu je důležité definovat, kdo má příběh prožívat. A proto se jako první musí určit role.

- Cíl – jedná se o specifikované požadavky, co má systém dělat za činnosti. Popis by měl být výstižný, avšak musí být stručný, krátký a jednoznačný.

- Užitek – jedná se o to, čeho se tím vlastně chce dosáhnout. Tato část user story je nepovinná.

User story má danou jedinečnou formu, která je znázorněna na obrázku níže.

Obrázek 11: User story (Zdroj: Vlastní)

Backlog

Backlog je seznam nevyřízených úkolů, které je potřeba splnit. Také by se dalo říci, že jde o user stories, které je potřeba implementovat. Existují dva základní typy backlogu.

- **Product Backlog (PB)**

Za Product Backlog je zodpovědný Product Owner. Product Backlog je kompletní seznam user stories, které je potřeba vyřešit během vývoje softwaru. Nikdy nemá svou konečnou podobu, položky lze přidávat, měnit či odebírat. Pro dosažení co nejlepšího výsledku, se Product Backlog stále kontroluje a udržuje. Každá položka PB má stanovenou prioritu, popis a odhad. Jednotlivé položky jsou rozdělené do určitých sprintů a také by měla být v tomto sprintu i implementována.

- **Sprint Backlog (SB)**

Sprint Backlog je takzvaná podmnožina Product Backlogu. Ve SB se nachází jednotlivé položky, které by měly být implementovány během toho konkrétního sprintu. Položky ve sprintu se mohou měnit díky práci s taskama. Důležité je, že se nesmí měnit Sprint Goal.

Increment (Přírůstek)

Increment je součet všech položek z Product Backlogu, které byly implementovány a dokončeny v předchozích sprintech, ale i v současném sprintu. SCRUM týmem musí být označené jako „DONE“. Každý tým si sám určí a definuje, jak bude posuzovat hotové položky. Tyto hotové přírůstky se dále testují.

4.1.4 Meetingy

Pro kontrolu časových limitů, sledování nákladů a dodržování procesů jsou ve SCRUMu zavedené jednotlivé meetingy. Díky nim je tato metodika transparentní. Meetingy se dělí do tří základních typů – meetingy plánovací, hodnotící a hodnotící i plánovací. Níže jsou popsané jednotlivé meetingy.

Sprint

Sprint je první iterace SCRUM. Všechny sprinty by měli být stejně dlouhé a neměli by trvat příliš dlouhé období. Ideální se uvádí měsíční či čtrnácti denní sprinty, ovšem záleží na projektu. Výsledkem sprintu je přírůstek.

Sprint se skládá z jednotlivých částí- Sprint planningu, Stand-Upu, samotná vývojová část, Sprint review a Sprint Retrospective. Pokud SCRUM tým nestíhá či naopak má hotovo dříve, může dojít ke změně ve Sprint Backlogu, kterou může provést pouze Product Owner. Může přesunout danou položku do dalšího sprintu či naopak přidat položku z jiného sprintu, avšak cíl sprintu se nesmí změnit.

Sprint Planning

Sprint planningu se účastní celý SCRUM tým i Scrum Master, ten opět pouze jako moderátor či může zrekapitulovat aktuální situaci. Požadavky jsou seřazeny podle priorit a odhadnuty časové nároky na implementaci, které se pochopitelně postupně mění a upřesňují.

Stand-Up (Daily SCRUM)

Každý den začíná Stand-Upem. Jedná se o 15-ti minutovou schůzku, kde se Development tým sesynchronizuje a naplánuje si činnosti na dalších 24 hodin přímo u Scrum tabule. V neposlední řadě si také sdělí, co kdo dělal předchozí den, a pokud má jakýkoliv člen

týmu problém s nějakým úkolem, poradí si a vyřeší se. Scrum Master má pouze jednu funkci a to jako moderátor.

Sprint Review

Neformální schůzka mezi development týmem a zákazníkem. Na tomto meetingu se představí přírůstek na produktu a celkově se zhodnotí. Představují se pouze user stories, které hotové, otestované, zaznamenány v dokumentaci a zároveň akceptovány Product Ownerem. V případě potřeby může Product Owner upravit backlog.

Sprint Retrospective

Sprint Retrospective je retrospektiva na proces. Zpětná vazba na práci ve SCRUM týmu. Členové týmu se zde dozvědí, kde jsou nedostatky a co v procesu zlepšit, aby se jim lépe pracovalo a tým byl efektivnější.

4.1.5 Vyhodnocení SCRUM

V dnešní době je SCRUM metodika jedna z nejpopulárnějších agilních metodik. Nejvhodnější je pro menší týmy pouze o několika členech. Ideální stav je 3-9 členů. Avšak to neznamená, že tato metodika nelze využít i ve větších korporacích. I tak se SCRUM dá použít, bude to akorát jen složitější a může se i stát, že bude spolupracovat více týmů dohromady.

4.2 Nástroje podporující agilní přístup

Každá společnost, která využívá agilní metodiky, musí zároveň používat agilní nástroje. Cílem využívání agilních nástrojů je udělat metodiku efektivnější a přesnější.

Nástrojů podporující agilní metodiky existuje velké množství. Mezi ty nejznámější patří například Rational team concert, Redmine, Microsoft Visual SourceSafe, Ora, Microsoft Team Foundation Server a mnoho dalších. Jeden z nich bude podrobněji popsána v kapitole 3.3 a jedná se o nástroj JIRA.

4.3 JIRA

JIRA je jeden z nejvíce využívaných nástrojů v automobilovém průmyslu ŠKODA Auto a.s.. Tento nástroj vytvořila společnost Atlassian. JIRA je vyvíjena v programovacím jazyce JAVA na platformě Java Virtual Machine. V dnešní době existuje již několikátá verze, poslední verzí je prozatím JIRA 7.

4.3.1 Charakteristika JIRA

JIRA Software podporuje veškeré metodologie agilní správy projektů pro vývoj softwaru. [10] Jedná se o softwarový nástroj, který se dá použít pro jakoukoliv agilní metodiku (SCRUM, Kanban, XP, ale i vlastní agilní metodiky). Vše co je potřeba ke správnému vývoji nového softwaru obsahuje právě nástroj JIRA. Od plánování, sledování, až po správu projektu. Jedny ze základních příčin vývoje tohoto nástroje byly, zjednodušit proces řízení agilních projektů spolu s požadavky. Dále zefektivnění práce celého týmu, který zde zaznamenává svou činnost a tak každý může vidět, co se právě v projektu děje.

JIRA je flexibilní nástroj, který se dovede přizpůsobit jakékoliv metodice či kombinaci metodik jak je již zmíněno výše. A právě v takovém případě se může lehce nástroj JIRA měnit (dodat jednotlivé speciální nástoje), aby vyhovoval zvolené metodice. Například jen stručně, jak vypadají dodatkové nástroje pro SCRUM a Kanban.

JIRA pro SCRUM

Scrum pracuje v určitých iteracích s jasně danou délkou, které se opakují. Tyto iterace jsou vymezeny čtyřmi po sobě jdoucími body: Plánování Sprintů, denní Stand-UPy, Sprinty a Retrospektivou Sprintu. Pro každý zde zmíněný bod jsou různé nástroje.

Plánování Sprintů a jejich nástroje:

- Správa verzí
- Sprint Planning
- User stories
- Snadná péče o backlog
- Scrum board

Denní Stand-UPy a jejich nástroje:

- JQL a filtry
- Dashboardy

Sprinty a jejich nástroje:

- Oprávnění ke Sprintům
- Workflow
- Centrum verzí
- Běžné požadavky

Retrospektiva a její nástroje:

- Různé typy grafů (Burndown graf, Kontrolní graf, Graf rychlosti, …)
- Sestavy (Epiců, sprintů, verzí, …)
- Diagram kumulativního průběhu

JIRA pro Kanban

Agilní metodika Kanban pracuje s myšlenkou, že neustále dodává svou novou verzi. Proto JIRA zobrazuje board se sloupci, kde je vidět, aktuální stav práce. Stejně jako Scrum má Kanban své čtyři základní pilíře: Nepřetržité dodávání verzí, limity práce, seznam práce a sloupce. K tomu slouží tyto nástroje.

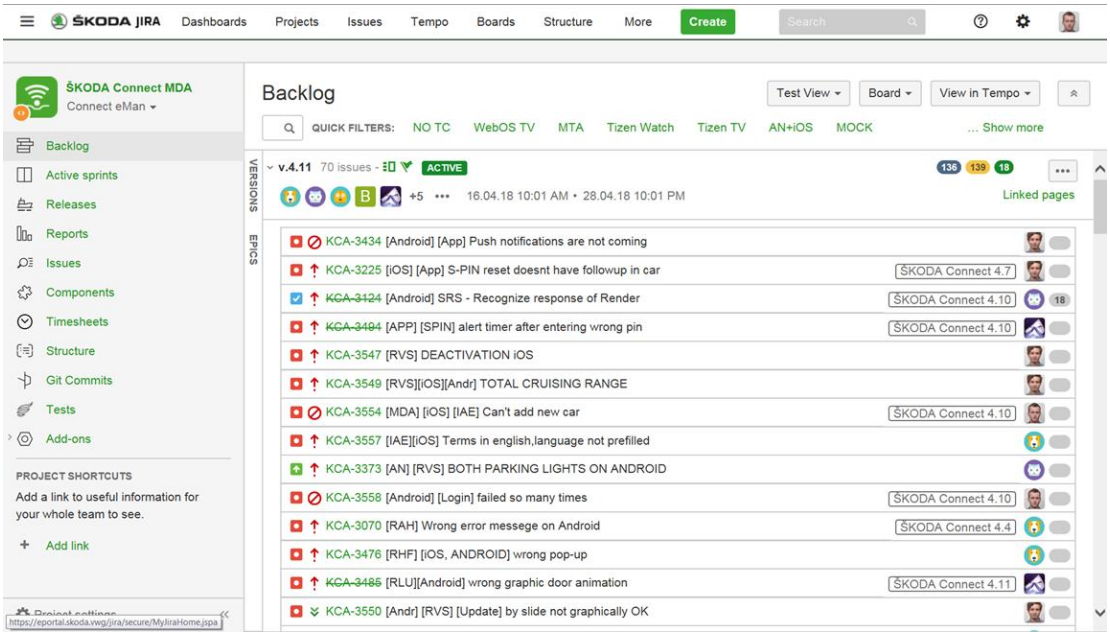
- Karty příběhu
- Dráhy a sloupce
- Konfigurace limitů probíhající práce
- Flexibilní workflow
- Kontrolní graf
- Diagram kumulativního průběhu
- Kanban board

4.3.2 Využití JIRA ve vývoji ŠKODA Auto a.s.

Pro vývoj mobilních aplikací ve ŠKODA Auto a nejen v této části společnosti se využívá nástroje JIRA. Tento nástroj byl vybrán díky jeho flexibilitě, efektivitě, jednoduchosti a přehlednosti.

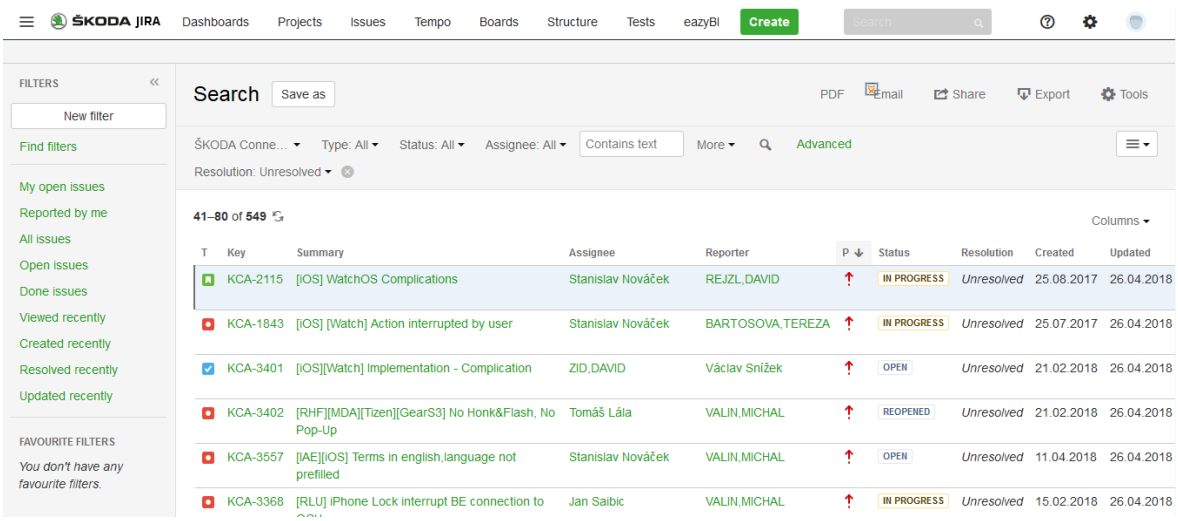
Prvním krokem nového uživatele je žádost o přístup, kde si zvolí i roli, kterou v daném projektu má. O přístup lze zažádat jak do jednotlivých projektů, tak i do celé skupiny projektů. Každá role má odlišná práva, a tím je zároveň zabezpečená síť, aby nedošlo k úniku dat. V dnešní době umožňuje společnost ŠA přístup do JIRA i dodavatelům. Tudíž lze mluvit o nástroji, který může používat jak interní, tak i externí osoba, což je velký pokrok, který práci v tomto nástroji zefektivnil.

Každý projekt, zde má své místo a své jednotlivé části. Pro lepší představu, jak se s JIRA pracuje, jsou zde připravené obrázky s vysvětlením.



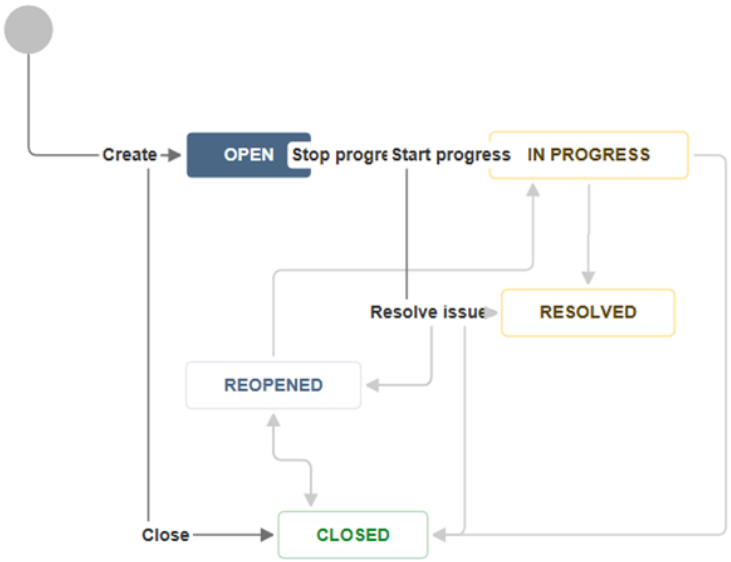
Obrázek 12: JIRA – Backlog (Zdroj:Vlastní)

JIRA v podstatě mapuje celý vývoj projektu. Začínáme přehledem backlogu, kde jsou jednotlivé tickety zadané na daný projekt. Ticket není pouze chyba (bug), ale existují i další typy: task, question, epic a improvement.



Obrázek 13: JIRA - Issue (Zdroj: Vlastní)

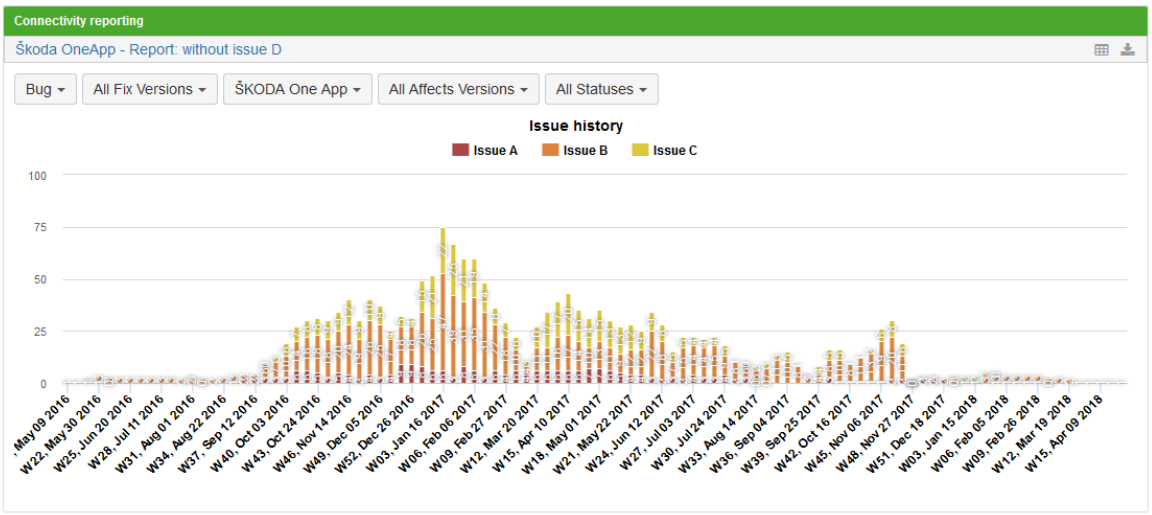
Tento obrázek zobrazuje přehled ticketů, které se při testování zadaly. Při hledání chyb, se dá využívat rozsáhlý filtr, který práci zrychlí. Jelikož do JIRA mají přístup i dodavatelé, kteří z největší části přispívají k vyřešení chyby, celý proces je rychlejší.



Obrázek 14: JIRA- Workflow (Zdroj: Vlastní)

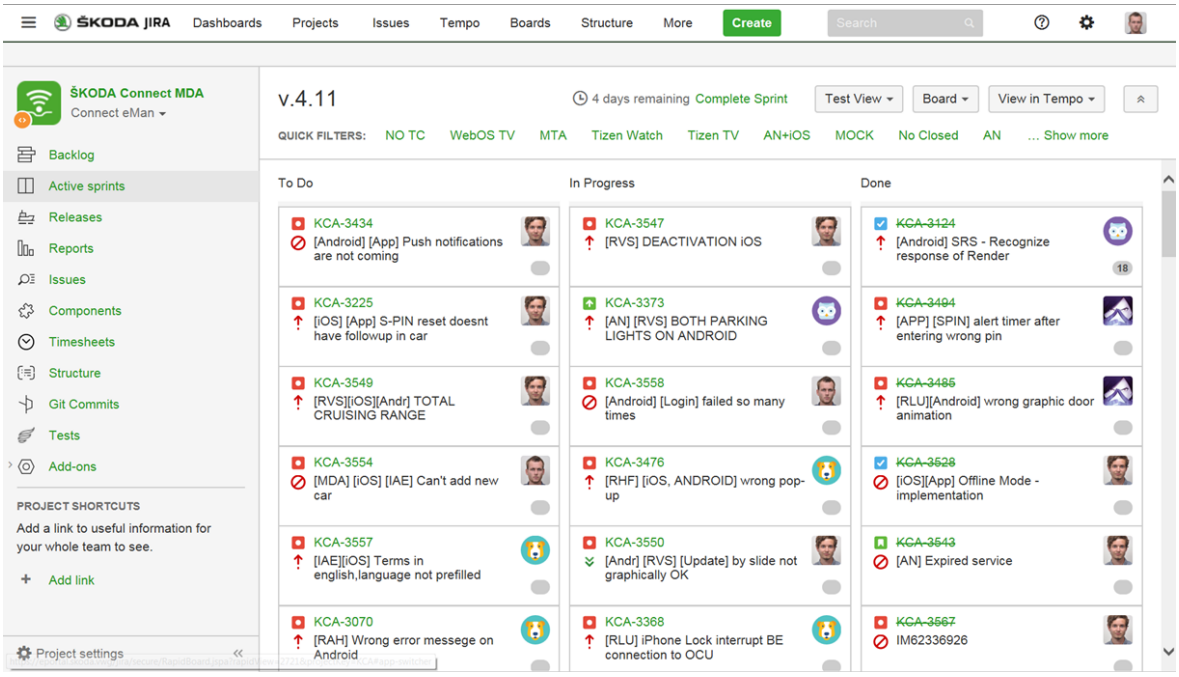
JIRA umožňuje rychlé mapování chyb, a pro lepší práci s takovými to případy, zde je velmi dobře rozpracované workflow. Workflow definuje vytvoření, vyřešení a uzavření jednotlivého ticketu. To přispívá k přehlednému průběhu procesu.

.



Obrázek 15: JIRA – Grafy (Zdroj: Vlastní)

JIRA má nepřeberně velké množství grafů, které lze jednoduše vygenerovat jedním stisknutím tlačítka.



Obrázek 16: JIRA - Aktuální Sprint (Zdroj: Vlastní)

Poslední obrázek z JIRA zobrazuje aktivní Sprint. Na této stránce jsou vidět jednotlivé položky z backlogu, tedy Sprint Backlog. Celý aktuální Sprint se rozděluje do tří částí: To Do, In Progress a Done.

Jeden větší nedostatek nástroj JIRA má. Jedná se o ukládání souborů, které zde není možné. Pro tuto nutnost bylo zapotřebí zavést další nástroj a to JIRA Confluence.

4.4 Návrh metodiky pro vývoj mobilních aplikací ve ŠKODA Auto a.s.

Oddělení zabývající se vývojem mobilních aplikací pro automobilový průmysl ŠKODA začala s přechodem na kombinaci rigorózních a agilních metodik ke konci roku 2014. A jelikož se do dnešní doby vůz vyvíjí pouze tradičními přístupy, je problém tyto dvě metodiky spojit dohromady. Z toho důvodu zde byl navržen proces, který ve vývoji mobilních aplikací funguje lépe spolu s výrobou automobilu.

Začátek využití agilních přístupů ve vývoji mobilních aplikací vedl k dlouhé diskuzi a dohadům, zda je to vůbec v tak velké společnosti s tak striktními přístupy možné. Avšak se podařilo si tento postoj prosadit, a tak se začal vývoj mobilních aplikací měnit a přizpůsobovat agilnímu přístupu, podle zde zmíněného návrhu.

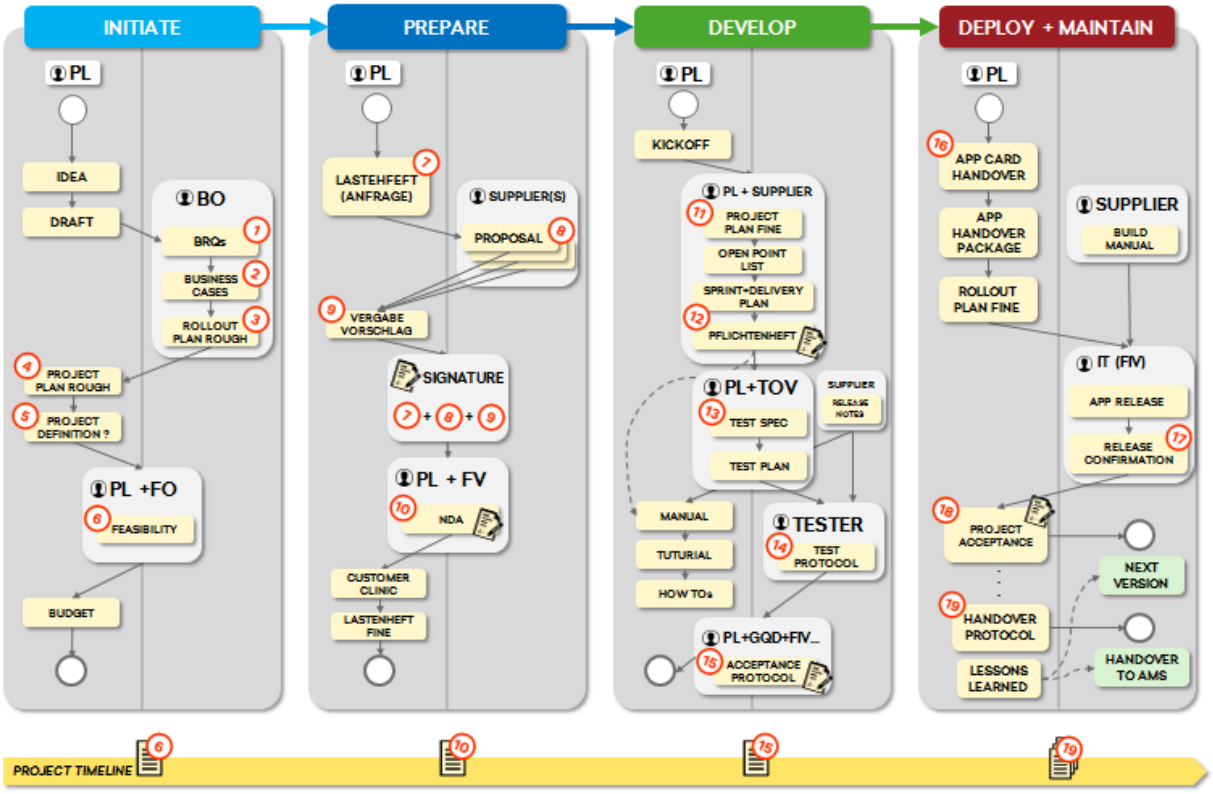
Tento návrh se již zavedl a je využíván v plné míře. Níže je tento návrh procesu popsán více do detailu.

Každý projekt se skládá ze čtyř fází:

- Initiate (Zahájení)
    - V této první fázi projektu vzniká koncept produktu (aplikace), který je na konci této fáze schválen na vyšších pozicích.
    - Nejdůležitější položky ve fázi Initiate jsou Draft (návrh) a Budget (rozpočet)
- Prepare (Plánovací)
    - V plánovací fázi se vyhodnocují náklady na implementaci produktu přes všechna oddělení ŠA. Zároveň se vyhodnocuje business case, aby se prověřilo, že projekt bude ziskový.
    - Během této fáze Prepare je nejzásadnější vytvoření Lastenheftu.
- Develop (Vývoj)
    - Ve třetí fázi implementace a vývoje produktu je hotový plán projektu se všemi specifikacemi.
    - Celý vývoje je rozdělen do 14ti denních Sprintů.
    - Během fáze Develop je produkt stále testován, ze strany dodavatele i ze strany zákazníka (ŠA).
    - Tato fáze se uzavírá akceptačním protokolem.

- Deploy + Maintain (Nasazení + Údržba)
    - Projekt je ukončen nasazením do produkce a podepsáním akceptačního protokolu všemi zainteresovanými stranami.

Jednotlivé fáze i s jejich náležitostmi jsou znázorněny na obrázku níže.



Obrázek 17: Proces vývoje mobilní aplikace (Zdroj: Vlastní)

Role v projektu dle interní metodiky ŠA pro vývoj mobilních aplikací:

- PL (Projektový manažer)
    - Je zodpovědný za celý projekt
    - Komunikuje každodenně s dodavatelem a celým týmem
    - Vytváří Lastenheft, Pflichtenheft a podílí se na Test specifikaci
- FO (Funkční vlastník)
    - Je zodpovědný za funcionalitu
    - Komunikuje každodenně s celým týmem, občas i s dodavatelem
    - Vytváří Lastenheft, Pflichtenheft a podílí se na Test specifikaci

- TOV (Zodpovědný tester)
    - Je zodpovědný za stav konkrétní aplikace
    - Komunikuje každodenně s celým týmem, občas komunikace i s dodavatelem
    - Vytváří Test specifikaci a Test casy, Test reporty a Test protokol
- Tester
    - Komunikuje každodenně s celým týmem
    - Podílí se na Test casech a Test reportech

5. Vyhodnocení navrženého propojení metodik

Poslední část bakalářské práce se zabývá celkovým vyhodnocením navrženého propojení metodik. Jedná se o propojení vývoje mobilních aplikací spolu s vývojem vozu. Tato kapitola také představuje kritéria pro vyhodnocení.

5.1 Kritéria pro vyhodnocení

Jednotlivé metodiky se hodnotí podle čtyř hlavních skupin kritérií. Mezi ně patří Proces, Podpora, Produkt a Lidé. Při hodnocení všech kritérií se postupuje podle stupnice 0 – 5.

0-1 – nízký stupeň splnění kritéria
2-3 – střední stupeň splnění kritéria
4-5 – vysoký stupeň splnění kritéria

5.1.1 Kritéria skupiny Proces

Kritéria spadající do skupiny Proces jsou zaměřená především na procesy životního cyklu, model životního cyklu, metriky, způsob vývoje a role. Jedná se tedy o procesy při vzniku softwaru. V této skupině se nachází celkem sedm kritérií.

- Kritérium Model životního cyklu
- Kritérium Rozsah
- Kritérium Role
- Kritérium Podrobnost popisu procesu
- Kritérium Dokumenty
- Kritérium Metriky
- Kritérium Řízení kvality

5.1.2 Kritéria skupiny Podpora

Kritéria spadající do skupiny Podpora hodnotí, dostupnost metodiky a kvalifikovaných lidé. Dále také podporu pro získání, publikování a přizpůsobení. Tato skupina obsahuje osm podkritérií.

- Kritérium Celistvost zdrojů
- Kritérium Dostupnosti
- Kritérium Podpora metodiky softwarovými nástroji
- Kritérium Podpora zavedení metodiky
- Kritérium Přizpůsobení metodiky
- Kritérium Výuka na vysokých školách
- Kritérium Školení a certifikace
- Kritérium Lokalizace

5.1.3 Kritéria skupiny Produkt

Tato skupina Produkt označuje charakteristiky produktu právě toho konkrétního projektu. Do skupiny kritéria skupiny Produkt se řadí celkem pět podkritérií.

- Kritérium Důležitost produktu
- Kritérium Délka projektu
- Kritérium Stálost požadavků
- Kritérium Znuvupoužitelnost
- Kritérium Velikost řešení

5.1.4 Kritéria skupiny Lidé

Tato skupina kritérií Lidé charakterizuje především tým, který pracuje na projektu. Kritéria skupiny Lidé se dělí do šesti částí.

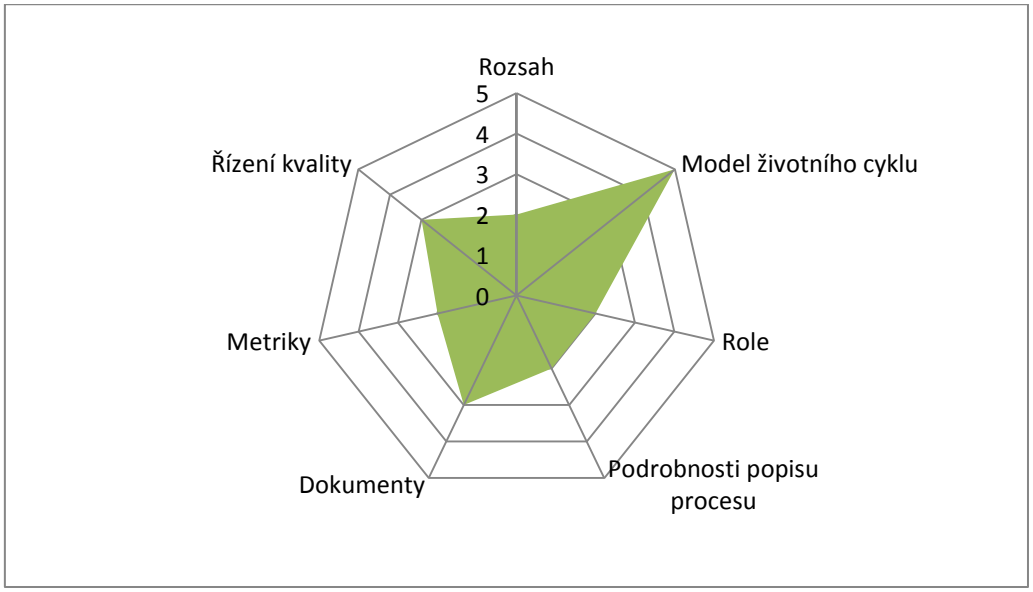
- Kritérium Zkušenosti manažera projektu
- Kritérium Kvalifikace členů týmu
- Kritérium Motivace členů

- Kritérium Dostupnost uživatelů
- Kritérium Velikost týmu
- Kritérium Rozmístění

5.2 Vyhodnocení řešení

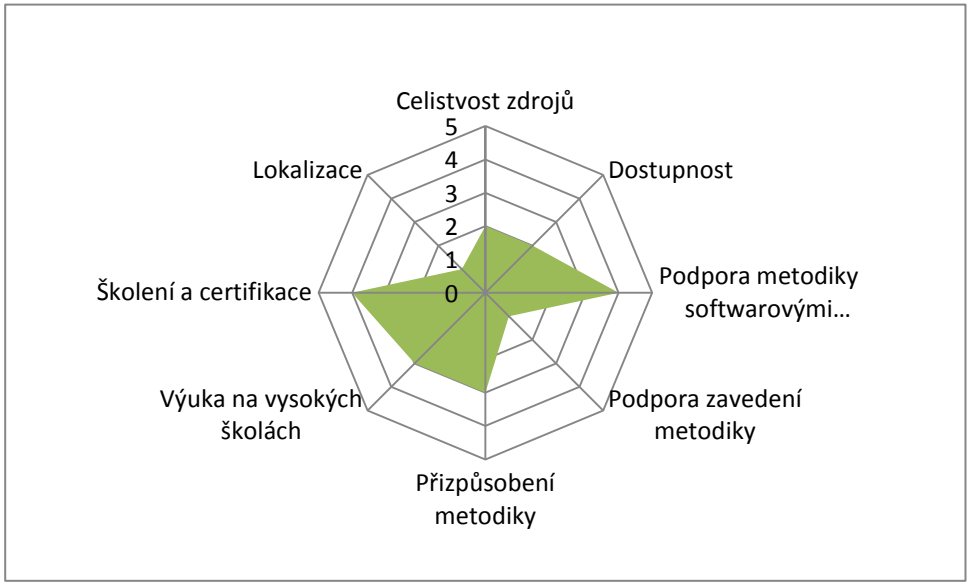
V poslední páté kapitole je vyhodnoceno navržené řešení. A proto je zde popsáno vyhodnocení agilní metodiky, která se využívá pro vývoj mobilních aplikací, podle kritérií výše.

Pro jednodušší představu jsou zde vytvořené tři paprskové grafy, které přímo odrážejí danou situaci. První zobrazuje schéma kritérií skupiny Proces, druhý skupiny Podpora a třetí skupiny Produkt spolu se skupinou Lidé.



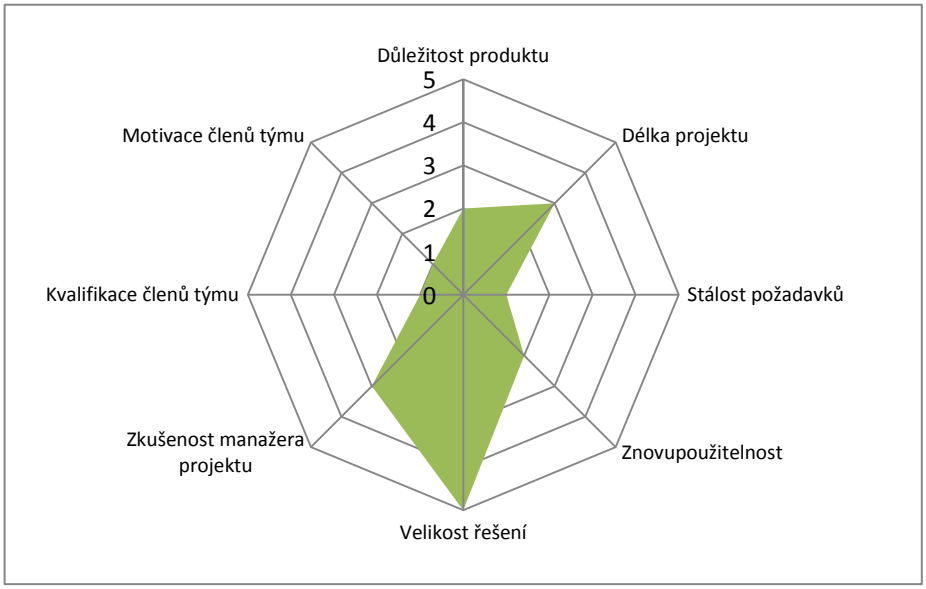
Obrázek 18: Paprskový graf- Kritéria skupiny Proces (Zdroj: Vlastní)

Tento paprskový graf zobrazuje jednotlivé kritéria skupiny Proces. Nejvíce zde vyniká Kritérium Model životního cyklu – tato metodika využívá velmi krátké iterace, a to 14ti denní, proto je označená na stupnici 5. Jelikož se jedná o velkou společnost, která se převážně řídí tradičními přístupy, je nutné vést dokumentaci i v navrženém řešení. Tudíž je Kritérium Dokumentace na stupnici 3.



Obrázek 19: Paprskový graf- Kritéria skupiny Podpora (Zdroj: Vlastní)

Druhý paprskový graf znázorňuje skupinu kritérií Podpora. V této skupině jsou na stupnici 4 dvě kritéria – Podpora metodiky softwarovými nástroji a Školení a certifikace. Pro vybranou metodiku se používá ve ŠA používá nástroj JIRA.



Obrázek 20: Paprskový graf- Kritérium skupin Produkt a Lidé (Zdroj: Vlastní)

Poslední paprskový graf zobrazuje dvě skupiny kritérií Produkt a Lidé. Na první pohled zde vyčnívá kritérium Velikost řešení. Toto kritérium znázorňuje velikost zdrojového kódu. Proto je v grafu vybrán nejvyšší stupeň. Dalšími kritérií s vyššími čísly jsou Délka projektu a Zkušenost manažera projektu. Délka projektu se ve ŠA liší projekt od projektu, každá aplikace se vyvíjí jinak dlouho. Tudíž nelze zcela jasně učit jednotnou délku.

Závěr

Cílem bakalářské práce bylo nejprve analyzovat současný stav metodik ve vývoji produktu a to jak rigorózní, tak agilní. Dále bylo cílem navrhnout metodiku pro automobilový průmysl.

V první části práce byly popsány hlavní charakteristiky těchto metodik a dále byly porovnány mezi sebou. Z porovnání metodik vyšlo, že tyto metodiky jsou vhodné pro různé účely vývoje softwaru. Rigorózní metodiky jsou vhodnější pro vývoj produktu, kde je zadání známé už v začátku projektu a v průběhu se nemění. Oproti tomu agilní metodiky vycházejí z nepřesného zadání, které se v průběhu projektu často upřesňuje, a během vývoje přicházejí stále další požadavky.

Stejný závěr se také ukázal při analýze současného stavu vývoje produktu a používání metodik ve ŠKODA Auto. ŠA využívá rigorózní metodiky při výrobě auta a jeho dílů, ale pro vývoj softwaru a konkrétně mobilních aplikací komunikujících s vozem, používá metodiku agilní. Tato metodika čerpá z klasických agilních metodik, především ze SCRUM.

Cíl bakalářské práce byl zdárně splněn navržením agilního přístupu ve vývoji mobilních aplikací, které se propojily s tradiční rigorózní metodikou pro výrobu vozu tak, aby naplnily cíle finálního produktu, tedy vozu, tak IT systému a mobilních aplikací, které komunikují s vozem a nejsou tak přímo závislé na výrobním procesu vozu. Jelikož původní problém se především týkal toho, jak spojit výrobu vozu a vývoj aplikací, které jsou přímo závislé na náběhu nového automobilu.

Dalším důležitým bodem bylo přizpůsobit aktuální agilní metodiku vývoje softwaru s metodikou PEP pro vývoj celého vozu. V rámci naplnění tohoto cíle byly v třetí kapitole zanalyzovány metodiky pro vývoj vozů ve ŠKODA Auto.

Vzhledem k tomu, že na sobě tyto metodiky pro vývoj vozu a mobilních aplikací byly nezávislé, bylo nutné navrhnout jejich propojení, aby softwarový produkt, jako je mobilní aplikace, přímo navazovala na výrobní proces vozu a byla tak brána v úvahu jako klasický hardwarový díl vozu, který je v metodice výroby vozu začleněn. Aby toto propojení mohlo fungovat, byly ve vývoji mobilních aplikací zachovány důležité dokumenty Lastenheft a Pflichtenheft.

V závěrečné části práce bylo navržené propojení metodik vyhodnoceno dle kritérií a vytvořené 3 paprskové grafy pro jednotlivé skupiny.

Vzhledem k úspěšnému nasazení mobilních aplikací do produkce současně s vozem a vzhledem ke spokojenosti zákazníků s produkty se ukazuje, že využití kombinace metodiky rigorózní a agilní je vhodná a přínosná i pro tradiční automobilový průmysl ŠKODA Auto.

Seznam použité literatury

Knižní zdroje

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

[2] BUCHALCEVOVÁ, Alena. *Návrh metodického rámce IS/ITC.* Doktorská disertační práce. Praha: KIT VŠE, 2004. 172 s.

[3] COHN, Mike. *User stories applied: for agile software development.* Boston: Addison-Wesley, c2004. ISBN 9780321205681.

[4] KOTRLA, Tomáš. *Agilní metodiky vývoje software.* Diplomová práce. Brno: MU, 2005. 55 s.

[5] MÁCHAL, Pavel, Martina ONDROUCHOVÁ a Radmila PRESOVÁ. *Světové standardy projektového řízení: pro malé a střední firmy: IPMA, PMI, PRINCE2*. Praha: Grada, 2015. Manažer. ISBN 978-80-247-5321-8.

[6] MYSLÍN, Josef. *Scrum: průvodce agilním vývojem softwaru.* Brno: Computer Press, 2016. ISBN 978-80-251-4650-7.

[7] SODOMKA, Petr a Hana KLČOVÁ. *Informační systémy v podnikové praxi. 2.*, aktualiz. a rozš. vyd. Brno: Computer Press, 2010. ISBN 978-80-251-2878-7.

[8] STOBER, Thomas. a Uwe HANSMANN. *Agile software development: best practices for large software development projects.* New York: Springer, c2010. ISBN 9783540708308.

[9] ŠOCHOVÁ, Zuzana a Eduard KUNCE. *Agilní metody řízení projektů.* Brno: Computer Press, 2014. ISBN 978-80251-4194-6.

[10] Interní materiál společnosti ŠKODA Auto – Prezentace PEP

[11] Interní materiál společnosti ŠKODA Auto – Popis metodiky PEP

Internetové zdroje

[12] Agile nástroje pro softwarové týmy – Jira Software | Atlassian. *Atlassian | Nástroje pro vývoj softwaru a spolupráci* [online]. Copyright © 2017 Atlassian [cit. 22.03.2018]. Dostupné z: https://cs.atlassian.com/software/jira/agile

[13] BECK, K., et al. *Manifest Agilního vývoje software* [online]. 2001 [cit. 2014-03-22]. Dostupné z: http://agilemanifesto.org/iso/cs/.

[14] It's all about being Agile : Crystal - It is about Self-awareness. *It's all about being Agile* [online]. Dostupné z: http://agilegod.blogspot.cz/2014/05/crystal.html

[15] Lean Development - What Is It, Info, |Demo| and Free WHITEPAPER. *Software Intelligence for Digital Leaders | ČÁST* [online]. Dostupné z: https://www.castsoftware.com/glossary/lean-development

[16] Proč používat verifikaci softwaru? - 09/06/2009 - Control Engineering Česko. Hlavní strana - *Control Engineering Česko* [online]. Copyright © 2007 [cit. 30.04.2018]. Dostupné z: http://www.controlengcesko.com/hlavni-menu/artykuly/artykul/article/proc-pouzivat-verifikaci-softwaru/

[17] Project Management tools you should know in 2018 | Apiumhub. *Software Development Company in Barcelona, Spain | Apiumhub*[online]. Copyright © 2018 Apiumhub All rights reserved [cit. 22.03.2018]. Dostupné z: https://apiumhub.com/tech-blog-barcelona/agile-project-management-tools/

[18] RUP - Rational Unified Process | Testování softwaru. *Testování softwaru* [online]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/rup/

[19] The XP Developer Shortage - SolutionsIQ. *Agile Enterprise Solutions - Scaling Agile for Organizations* [online]. Copyright © 2018 SolutionsIQ. All rights reserved. [cit. 22.03.2018]. Dostupné z: https://www.solutionsiq.com/resource/blog-post/the-xp-developer-shortage/

[20] 301 Moved Permanently. *301 Moved Permanently* [online]. Dostupné z: http://www.fi.muni.cz/~smid/mis-zivcyk.htm