TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Inovace aplikace LaunchTrack z pohledu UX/UI

Bakalářská práce

Studijní program: B6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika

Autor práce: **Jakub Žďárský**
Vedoucí práce: Ing. Petr Weinlich, Ph.D.
Katedra informatiky

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Zadání bakalářské práce

Inovace aplikace LaunchTrack z pohledu UX/UI

Jméno a příjmení: **Jakub Žďárský**
Osobní číslo: E17000027
Studijní program: B6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika
Zadávající katedra: Katedra informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Definice procesů vývoje softwaru
2. Analýza a sumarizace stávající verze aplikace
3. Řízení a optimalizace procesů vývoje aplikace LaunchTrack
4. Návrhy inovačních komponent za účelem zvýšení uživatelské přívětivosti
5. Zhodnocení a doporučení

Rozsah grafických prací:
Rozsah pracovní zprávy: 30 normostran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

- ŘEZÁČ, Jan. 2016. *Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů*. Vydání druhé. Brno: House of Řezáč. ISBN 978-80-270-0644-1.
- HARTSON, Rex. 2012. *The UX book: process and guidelines for ensuring a quality user experience*. Waltham, MA: Morgan Kaufmann. ISBN 978-0-12-385241-0.
- ŘEPA, Václav. 2012. *Procesně řízená organizace*. Praha: Grada. Management v informační společnosti. ISBN 978-80-247-4128-4.
- WEINSCHENK, Susan. 2012. *100 věcí, které by měl každý designér vědět o lidech*. Brno: Computer Press. ISBN 978-80-251-3649-2.
- MCKAY, Everett N. 2013. *UI is communication: how to design intuitive, user centered interfaces by focusing on effective communication*. Boston: Elsevier, Morgan Kaufmann. ISBN 9780123969804.
- PROQUEST. 2019. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2019-09-26]. Dostupné z: http://knihovna.tul.cz

Konzultant: Ing. Jan Parpel

Vedoucí práce: Ing. Petr Weinlich, Ph.D.
Katedra informatiky

Datum zadání práce: 31. října 2019
Předpokládaný termín odevzdání: 31. srpna 2021

L.S.

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

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

V Liberci dne 31. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, 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 nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

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

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

31. července 2020 Jakub Žďárský

Anotace

Bakalářská práce se zaměřuje na téma inovace aplikace LaunchTRACK z pohledu UX/UI (User eXperience/User Interface). V teoretické části jsou vysvětleny pojmy související s obory vývoje softwaru a návrhu uživatelského prostředí, které přiblíží tématiku uživatelské přívětivosti. Dále je věnován prostor definování softwarových rolí vývojového týmu a popis jednotlivých kompetencí každého z členů. V praktické části jsou využívány teoretické vědomosti, například při analýze aplikace LaunchTRACK 1.0, anebo k následnému návrhu inovačních komponent pro zdokonalení nové verze. Aplikace LaunchTRACK je testována také pomocí heuristických metod a uživatelského hodnocení testery z řad potenciálních uživatelů. Je zde také popsán a analyzován návrh k přechodu na agilní metodiku vývoje softwaru, pro zvýšení efektivity a kvality dodávky.

Klíčová slova

UX, Uživatelský prožitek, Uživatelské rozhraní, SWOT analýza, heuristická analýza, ŠKODA AUTO, LauchTRACK, agilní metodika, informační architektura, interakční design, vývojářské role

Annotation

Innovation of LaunchTRACK application from the UX/UI perspective

This bachelor thesis focuses on the topic of innovation of Launch TRACK application from the UX / UI (from a User experience / User Interface) perspective. The theoretical part explains the concepts related to the fields of software development and user interface design, which will introduce the topic of user friendliness. Furthermore, space is devoted to defining the software roles of the development team and a description of the individual competencies and strengths of each member. In the practical part, theoretical knowledge is used, for example in the analysis of the Launch TRACK 1.0 application, or for the subsequent design of innovative components to improve the new version. Launch TRACK is also tested using various heuristic methods and user ratings by potential users. It also describes and analyzes a proposal for the transition to an agile software development methodology, to increase its efficiency and quality of delivery.

Keywords

UX, User eXperience, User Interface, SWOT analysis, Heuristic analysis, ŠKODA AUTO, LauchTRACK, agile development, information architecture, interaction design, development roles

Poděkování

Rád bych poděkoval panu Ing. Petru Weinlichovi, Ph.D. za odborné vedení bakalářské práce, jeho připomínky, rady, a především jeho čas, který mi věnoval. Mé poděkování patří dále odborné konzultantce paní Mgr. Tereze Semerádové, Ph.D. a celému Launch management týmu ŠKODA AUTO, ve kterém jsem v rámci dlouhodobé praxe mohl působit a všichni, tak obohatili mou pracovní zkušenost, kterou podpořili velkou řadou rad k absolvování celé dlouhodobé praxe. Následně bych rád poděkoval svým přátelům a rodině, která mě při psaní bakalářské práce, ale i v průběhu studia na vysoké škole podporovala.

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

BE	Back-end
CI/CD	Corporate identity and corporate design
FE	Front-end
HQ	Headquarter
IA	Information Architecture
ICT	Information and Communication Technology
IMP	Importer
IS	Information system
IT	Information Technology
IxD	Interaction Design
LM	Launch Management
LT	LaunchTRACK
ME	Market Entry
PoC	Proof of Concept
OOP	Object-oriented programming
SDLC	System development life cycle
ŠA	ŠKODA AUTO a.s.
UI	User Interface
UX	User Experience
VR	Virtual reality

Úvod

Moderní doba přináší moderní technologie napomáhající lidem usnadnit život a ušetřit čas. Někdy ovšem softwarová řešení toto tvrzení popírají a snaží se uživatelům jejich čas naopak vzít. Je zásadní přistupovat na změny chování uživatelů a celkový vývoj společnosti. Udržet krok s novými trendy návrhu softwaru, jak z vizuální, tak i z technologické stránky. Úkolem každého softwarového řešení by měla být motivace nenechat uživatele přemýšlet a vytvořit zcela intuitivní prostředí, ve kterém se bude uživatel cítit dobře.

„Pokud je něco použitelné, znamená to, že něco dobře funguje a že osoba s průměrnými (ba dokonce podprůměrnými) schopnostmi a zkušenostmi může používat určitou věc, a to bez větších problémů, než je nutno.“ (Krug, 2003)

Úkolem teoretické části bakalářské práce je přiblížení procesu vývoje softwaru a seznámení s pojmy, které s ním souvisí. Jsou zde popsány jednotlivé fáze životního cyklu vývoje informačního systému, jejich úkoly, úskalí a důležitost v rámci celého procesu. Dále jsou představeny definice metodiky softwarového vývoje a kompetence jednotlivých členů týmu. S ohledem na postoj uživatelů k softwarovým produktům je detailně rozebrána definice pojmu uživatelská přívětivost, se kterou je spojena řada oborů ovlivňujících spokojenost uživatele využívajícího software, produkt, či službu. V rámci UX je zahrnut proces návrhu uživatelsky přívětivého prostředí, metody uživatelského testování a rozdíly mezi pojmem UX a UI, které bývají často zaměňovány. Posláním teoretické části je tedy nabídnout čtenáři rychlý vhled do tématiky, tak, aby si mohl pojmy osvojit a případně využít.

Primárním cílem praktické části je využití nabytých zkušeností z části teoretické a vypracování analýzy prostředí aplikace LaunchTRACK, která nachází své využití v optimalizaci procesů v rámci vypouštění nových modelů automobilů společnosti ŠKODA AUTO a.s. na trh. Z analýzy aplikace LaunchTRACK 1.0 vycházejí silné a slabé stránky, ze kterých lze poté čerpat při návrhu nové verze. Jako analytický nástroj je využita SWOT analýza s navazující heuristickou analýzou tří nezávisle na sobě zvolených respondentů. Výstupy jsou dále vyhodnoceny a na jejich základě vznikají definice požadavků pro vývoj nové verze. Dalším krokem je vytvoření optimalizačního návrhu pro vývoj aplikace LaunchTRACK 2.0, který by přinesl efektivnější práci mezi zadavatelem a vývojáři. Tato změna by tak s největší pravděpodobností přinesla zlepšení kvality dodávaného softwaru. Finálním výstupem této bakalářské práce se stává návrh inovačních komponent určených pro zlepšení uživatelského prožitku a rozšíření nástrojů aplikace LaunchTRACK 2.0

1 Úvod do problematiky vývoje softwaru

V této kapitole si přiblížíme problematiku vývoje softwarového produktu. Hlavním cílem je čtenáři osvětlit základní pojmy, postupy a principy, které spadají pod celkový proces vytváření nového softwarového řešení. Je nutné formulovat metodiku zpracování, tak aby mohla být teoretická část příjemným studijním podkladem pro praktickou část bakalářské práce.

1.1 Proces vývoje softwaru

Proces vývoje softwaru je disciplínou, která je poměrně komplexní, ať už se na vývoji podílí větší, či menší vývojářský tým. Vždy je důležité, aby byl zadaný produkt jasně nadefinovaný, zanalyzovaný a byla vytvořena pevná struktura, podle které se každý člen vývojového týmu bude řídit. V následujících kapitolách se dozvíme, že vývoj můžeme dělit do konkrétních fází. Postupovat dle projektových metodik přinášejících různé cesty ke společnému výsledku. Dále si rozebereme jednotlivé role vývojového týmu, jejich kompetence a náplň práce. (Buchalcevová, 2015)

1.2 Stádia vývoje softwaru

Jedná se o pevně definované období mezi nově vyvíjeným projektem a jeho uvedením do provozu. Tento časový úsek je proto nedílnou součástí každého kroku od inovativní IT myšlenky až po nasazení velkého korporátního projektu na produkční prostředí a jeho následné údržby. Často můžeme znát tyto definice také jako životní cyklus vývoje informačního systému, který nese anglický název software development lifecycle. Široce využívanou zkratkou ve světě IT je tedy SDLC. (Buchalcevová, 2015)

Pod tímto pojmem si můžeme představit jednotlivé fáze v procesu vývoje, které na sebe neodmyslitelně navazují. Každá z těchto částí pracuje s určitými vstupy, z kterých má za povinnost vytvořit relevantní výstupy pro zachování cyklu a navázání na další fázi vývoje. Jeden krok bez druhého nedává v rámci software developmentu smysl. Proto je každý člen týmu důležitý a musí se řídit předem nadefinovanými pravidly. (Buchalcevová, 2015)

Existuje řada pravidel a postupů podle, kterých se životní cyklus vývoje softwaru může řídit. Nejstandardnější a nejvíce uznávaný princip předpokládá 7 konkrétních fází, kterými jsou:

- Fáze 1: Sběr a analýza požadavků.
- Fáze 2: Studie proveditelnosti.
- Fáze 3: Návrh.
- Fáze 4: Vývoj / Kódování.
- Fáze 5: Testování.
- Fáze 6: Instalace / Nasazení.
- Fáze 7: Údržba.

Obrázek 1 – Stádia vývoje softwaru

Zdroj: vlastní zpracování

1.2.1 Fáze 1: Sběr a analýza požadavků

První fází životního cyklu softwaru (SDLC) je požadavek. Požadavek zadává zákazník, který prezentuje své potřeby vyšším a seniorním členům vývojového týmu. Těmi jsou ve většině přístupů projektový manažer, architekt datové struktury a IT analytik. Tato fáze je také bohatá na brainstorming, zaplánování cílových požadavků a vydefinování rizik, které by mohly projekt ohrozit. Úkolem je poskytnout jasnější vhled do rozsahu práce a přiblížit tak zadavateli jednotlivé kroky v rámci vývoje. Často je zde aplikovaná SWOT analýza, která řeší rozčlenění projektu na silné/slabé stránky, příležitosti a hrozby. (Filipová, 2018)

Po získání přehledu o projektu je nezbytně nutné diskutovat problematiku společně s kompletním vývojovým týmem. Jednání probíhá většinou interně v rámci poptávané společnosti. Prvním důležitým bodem je začlenění zvoleného týmu do tématu. Diskuse s vývojáři přispěje zkušenostmi a vhledem do technologické problematiky. Je to moment, který pomáhá projektovému manažerovi a IT analytikovi nadefinovat orientační časovou osu reprezentující proces vývoje. Jeho trvání a výhledy do budoucna. Popřípadě předpřipravit body, které projekt mohou v jistých fázích omezit. Závěrem fáze sběru dat a analýzy požadavků je připravený orientační časový harmonogram a vnitřní povědomí poptávané firmy o celkové pracnosti projektu. (Filipová, 2018)

Obrázek 2 – Týmový brainstorming

Zdroj: (Fidelity, 2019)

1.2.2 Fáze 2: Studie proveditelnosti

Úkolem předchozího kroku bylo zjistit rozsah problematiky a navrhnout přístupy k řešení. V této fázi se pracuje s několika faktory jako je čas, finanční a technické zdroje, náklady, přínosy a účel vyvíjeného produktu. Je zapotřebí zdokumentovat potřeby a byznys logiku softwaru. Specifikaci softwarových požadavků známe také pod zkratkou SRS, kterou tvoří spojení anglických slov Software Requirement Specification. Dalším zásadním slovním spojením, které je využívané jako nástroj dokumentace studie proveditelnosti je Proof of Concept, neboli PoC. Tyto dokumenty zahrnují vše, co by mělo být v průběhu životního cyklu aplikace navrženo a vyvinuto. Existuje pět konkrétních ukazatelů a typů kontrol proveditelnosti, kterými jsou:

- Ekonomické: V tomto bodě si klademe otázku, zda lze projekt dokončit v rámci daného rozpočtu, či nikoliv. Popřípadě je vhodné zpětně analyzovat a diskutovat cenovku se zadavatelem.
- Technické: Je stěžejní ověřit dostupné technologie na požadované funkcionality. Zda na provedení zadané práce existují vhodné hardwarové a softwarové nástroje. Důležitým faktorem je také zkušenost týmu, který bude schopný zrealizovat jednotlivé požadavky v co největší kvalitě za co nejkratší čas.
- Právní: Aspekt, který ověřuje, zda je projekt v souladu s kybernetickým právem a může být realizován bez porušení všeobecných práv a soudních nařízení. Otázka je, zda software nepodléhá regulacím ze strany práva.
- Realizační tým: Jsme schopni sestavit tým, který je vhodný ke splnění požadavků dle očekávání klienta? Máme na tento projekt dostatečné pracovní kapacity?
- Čas: Rozhodnutí, jestli je reálné projekt zrealizovat v zadaném čase, podle předem definované časové osy.

Po zhodnocení těchto pěti typů ukazatelů jsme následně schopni uzavřít fázi, která ověřuje proveditelnost projektu a věnovat se další fázi návrhu.

1.2.3 Fáze 3: Návrh

Ve fázi návrhu jsou předloženy analýzy a podkladové materiály, které definují požadavky zákazníka zpracované IT analytikem. Potřebné materiály pomáhají při návrhu celkové architektury systému. Fáze návrhu slouží vývojářům jako zadání pro vypracování dílčích úkolů pro splnění všech funkcionalit projektu. Vždy jsou tvořeny dva druhy návrhových dokumentů:

High-Level Design (HLD)

Pojem High-Level Design je překládaný v kruzích českých vývojových týmů jako design na vysoké úrovni. Jeho úkoly jsou stručně popsat jednotlivé moduly a vydefinovat název pro každou komponentu. Vytvořit přehled funkcí, které očekáváme od aplikace. Zběžné vztahy mezi moduly. O detailní zpracování vazeb a vztahů se následně stará Low-level design. Dalším z úkolů je vytvořit databázové struktury, kde jsou vydefinovány klíčové prvky. Jako posledním zásadním bodem je kompletace schémat architektury společně s technologickými detaily. (McKay, 2019)

Low-level design (LLD)

Tento druh návrhu je do českého jazyka překládaný jako nízko úrovňový design. Mezi jeho primární cíle patří vymezit funkční logiku komponent a modulů aplikace. Nastavení datového typu a velikosti v rámci jednotlivých atributů. Tento přístup také řeší kompletní detail uživatelského rozhraní, jako grafický mustr pro následný vývoj. Řeší vazby mezi komponentami, tak aby odpovídaly nastavené business logice. V neposlední řadě se stará o možné chyby, které se mohou v aplikaci objevit, tak aby na ně mohli uživatele upozornit. Jako například, „Tato stránka neexistuje.“, popřípadě „Nevyplnil jste požadované pole formuláře“. Poslední a nejzásadnější součástí nízko úrovňového návrhu je řešení všech vstupů a výstupů napříč aplikací. (McKay, 2019)

1.2.4 Fáze 4: Vývoj / Kódování

Po ukončení fáze návrhu se přesouvá energie na vývoj neboli kódování systému. Je to fáze, kde začínají vznikat první kroky k tomu, aby byl produkt vizuálně či infrastrukturně hmatatelný. Vývojáři se řídí dle předem zvoleného programovacího jazyka, dle kterého začínají budovat aplikaci. Jejich zadání jsou detailně rozpracovány do jednotlivých úkolů vývoje, které cílí k co nejefektivnějšímu vypracování softwaru. V největší kvalitě a co nejmenším čase. Vývoj je nejdelší fází životního cyklu vývoje softwaru. (McKay, 2019)

Úkolem všech vývojových týmů je dodržovat předem předdefinovaná pravidla pro kódování. Někdy je také používaný výraz best practices, který nám udává ty nejvhodnější postupy pro psaní kódu. Tyto postupy se rychle vyvíjejí, a proto je důležité jako vývojář věnovat čas aktualizaci svých vědomostí v poli programovacích best practices.

Struktura vývojového týmu se nejčastěji dělí na dvě větve, kterými je frontendový a backendový tým. Jejich role se zásadně liší. Definice pro zmíněné role je následovná:

- Frontendový vývojář staví stránku tak jak bude v důsledku vypadat.
- Backendový vývojář programuje za účelem tvorby funkcionalit a logiky.

V následujících kapitolách si obě role rozvedeme do většího detailu.

1.2.5 Fáze 5: Testování

V momentě dokončení první fáze vývoje je z vývojového prostředí přesunut software na prostředí testovací, kde se nachází prostor pro testovací tým, který má za úkol aplikaci do posledního detailu protestovat. Zpracovat testovací scénáře. Tyto akce jsou plněny za účelem ověření, že celá aplikace funguje podle požadavků zákazníka.

V rámci této fáze může do procesu vstupovat testovací tým a oddělení QA (quality assurance), kteří ověřují správnost a chod aplikace. Je jejich povinností shledané bugy a nedokonalosti dokumentovat a reportovat na vývojářský tým. Ten chyby opraví a pošle je zpět na přetestování. Tento proces probíhá, dokud není software bez chyb, funguje stabilně, podle definované business logiky a obchodních podmínek. (McKay, 2019)

1.2.6 Fáze 6: Instalace / nasazení

Jakmile fáze testování softwaru skončí a v systému nezůstanou žádné závažné chyby, spustí se proces finálního nasazení na produkční prostředí. Na základě zpětné vazby a rozhodnutí poskytnuté projektovým manažerem je konečný software uvolněn. Následné chyby s nasazením na produkční prostředí se řeší schvalovacím kolečkem, které prochází skrz vývojové a testovací prostředí, dokud není vše odladěno. Tedy se vracíme zpět do pomyslné fáze testování. Po úspěšném nasazení se fáze uzavírá. (McKay, 2019)

1.2.7 Fáze 7: Údržba

V okamžiku, kdy je systém nasazen a zákazník má možnost doručený systém využívat, tak dochází k následujícím činnostem, kterými jsou: (McKay, 2019)

- Oprava chyb (BUGů) – chyby které nebyly doposud objeveny, popřípadě pro ně nebyly připraveny testovací scénáře, které by problém podchytily. Tato fáze nese z angličtiny převzatý název Bug fixing.
- Upgrade – Upgradováni je fáze, v níž konkrétní využívaná technologie v rámci dodaného produktu potřebuje aktualizovat na vyšší verzi.
- Vylepšení – Vylepšením rozumíme přidání nové inovační komponenty nebo zakomponováním nových funkcionalit do stávající verze softwaru.

Primárním záměrem poslední fáze SDLC je zaručit, aby byly potřeby zákazníka naplněny po celou dobu chodu aplikace. Tedy aby byl zajištěn plynulý chod. Je také důležité dbát na zpětnou vazbu, která přichází od uživatelů a umožnit naplnění jejich požadavků.

1.3 Role a kompetence vývojového týmu

Vývojářské týmy se v jednotlivých fázích životního cyklu vývoje softwaru mohou lišit. Tyto obměny vznikají v závislosti na použité metodice, fázi i etapě, ve které se vyvíjený software právě nachází. Z toho důvodu je složení týmu velmi často proměnlivé. Tým je tedy vždy přizpůsobený aktuální situaci, která je přímo úměrná s potřebou zákazníka. V těchto situacích je důležité, aby si každý člen týmu držel dovednost flexibility a dokázal své zkušenosti směřovat i jiným směrem. V praxi velmi často vzniká moment, kdy jeden člen týmu zastává více vývojových rolí. Kupříkladu z návrháře uživatelského prostředí se v následující fázi stává frontendový specialista, který svůj návrh vizualizace převede do kódu. Velmi často se také z vývojářů stávají testeři, kteří mají za úkol odladit své nedostatky, testují napříč aplikací a následně všechna nestandardní chování debuggují. (Martinů, 2015)

Vše musí být definováno, tak aby vývojový tým znal svou strukturu a byla zde dělba práce přímo úměrná ke kompetencím daného člena týmu. Týmy mohou být rozděleny například na tým řízení projektu vývoje softwaru a vyvíjení, či modelování softwaru.

Projektové řízení vývoje softwaru

- Primární cíle této skupiny jsou finanční a organizační úroveň vývoje.
- Odpovědnost nese projektový manažer.
- Projektový manažer organizuje skupinu odborníků, kteří mu pomáhají proces řídit a plnit zadané úkoly společně s realizačním týmem.
- Jeho úkolem je společně s IT analytikem nacenit komponenty a vytvořit časovou osu, dle které se bude projekt řídit.

Softwarový vývojový tým

- Součástí jsou všichni vývojáři, kteří se na projektu podílí
- Hlavní odpovědnost nese ve většině případů IT analytik, který komunikuje s klientem na denní bázi. Převádí požadavky business týmu na reálné úkoly k vypracování pro vývojový tým
- Globálním úkolem tohoto týmu je analyzovat zadání, následně jej specifikovat, navrhnout řešení, implementovat vyvinuté komponenty, testovat, případně opravit a zavést vyvinutý prvek na produkční prostředí

Mezi další přístupy patří dělba práce dle druhu přístupu. Ten může být strukturovaný anebo nestrukturovaný. Strukturovaný přístup se koncentruje na dělbu práce dle profese jednotlivých členů týmu. Na příkladu si můžeme strukturovaný tým představit jako tým backendových nebo frontendových vývojářů. Nestrukturovaný tým se dělí dle objemu práce, který na určitou skupinu dopadá. Většinou se zadává komplexní celek úkolů, kde si nestrukturovaná skupina za pomoci projektového manažera úlohy rozdělí. V nestrukturovaném týmu jsou si všichni rovni a nejsou vázaní svou konkrétní rolí.

Pro zvolení typu struktury vývojového týmu se rozhoduje projektový manažer dle velkého rozsahu faktorů. Často je volba ovlivněna rozsahem projektu, finančními zdroji, potřebou kompaktního týmu pro zlepšení interní komunikace nebo naopak velkého vývojového týmu, ze kterého se každý bude soustředit na menší celek. Z pravidla je tím rozhodujícím faktorem čas, který je podpořený finanční stránkou projektu. Pokud jsou finanční zdroje na dostatečné úrovni, potom se ve většině případů přistupuje na strukturovaný tým, který umožňuje mít projekt více pod kontrolou. (Buchalcevová, 2015)

Obrázek 3 – Příklad složení projektového týmu

Zdroj: vlastní zpracování

1.3.1 Business/Product Owner

Product owner neboli vlastník produktu je role, která neodmyslitelně patří vývojového procesu. Je to osoba, která zastupuje koncového uživatele nebo firmu, která daný produkt zadává. Působí jako primární kontaktní osoba určující všechna rozhodnutí týkající se tvoření

a změn v rámci projektu. Má velké rozhodovací právo, aniž by musel žádat souhlasu od sponzorů projektu. Je důležité jmenovat na tuto pozici kompetentní osobu, protože jde o klíčovou roli v rámci vývoje softwaru. Tato osoba je základním stavebním kamenem celého projektu. Nese zodpovědnost za přiřazení priority jednotlivým úkolům a maximalizování návratové hodnoty investice. Známé jako ROI softwarového produktu. Součástí této role je také zdokumentovat vývojové požadavky projektu. (Filipová, 2018)

Primární kompetence product ownera (Vlastníka produktu):

- Je kontrolním orgánem předem nadefinovaných podmínek. Snaží se držet vizi a business logiku softwarového produktu.
- Určuje konečná rozhodnutí směrem k vývoji nových komponent.
- Zodpovídá za průběžnou údržbu a aktualizaci nevyřízených a nových požadavků.
- Odstraňuje požadavky, které jsou mimo rozsah realizace anebo byly nahrazeny lepším řešením.
- přezkoumání a stanovení priorit přiřazených k nevyřízeným produktům a vedení všech schůzek pro plánování projektu.
- Udržování rovnováhy a motivace v interním nebo vývojovém týmu.

1.3.2 Projektový manažer (Vedoucí vývojového týmu)

Projektový manažer je stavěn do role administrativního vedoucího projektu, který má za úkol prezentovat aktuální stav vývoje. Je zodpovědný za celkový chod projektu. S tím je spjata organizace vývoje, řízení lidských zdrojů, zajištění finanční stránky a vytvoření ideálních podmínek pro vývojový tým. Mezi jeho kompetence patří plánování rozsahu, vydefinování jednotlivých milníků projektu společně se správou a řízením rizik. Dále také dohlíží na veškeré změny v rámci týmu a projektovém plánu. Člověk na pozici projektového manažera má za úkol motivovat tým k co nejefektivnějším výkonům. (Filipová, 2018)

Primární kompetence projektového manažera:

- Kompletně vede a řídí vývojový tým.
- Organizuje práci a přiděluje dílčí úkoly členům týmu.
- Vede jednání a odpovídá za splnění nadefinovaných úkolů.
- Je nedílnou součástí v rámci řízení změn celého projektu.

1.3.3 IT analytik

IT nebo také IT Business analytik je jednou z klíčových rolí vývojového týmu. Je to člověk, který analyzuje, navrhuje a následně implementuje informační systém. Jeho cílem je optimalizovat celý chod informačního systému a tedy i procesy a efektivitu organizace, do které je softwarový produkt dodáván. Jeho misí je diskuse se zadavatelem, který mu definuje širší dlouhodobý cíl. Úkolem analytika je, zhodnotit obsáhlé dlouhodobé zadání a zajistit informaci na jaké platformě bude nejlepší systém stavět. Využívá svůj skill set jako např. modelování, informační inženýrství a účetnictví zabývající se náklady na projekt, tak aby uspokojil všechny strany. Tedy stranu vývoje, ale i business zadavatele. Jeho primárním cílem je, aby byl systém vyvinut co nejefektivnějším způsobem. Po schválení struktury se IT analytik zabývá definováním jednotlivých vývojových tasků. Následně na to řeší koordinaci a správnost implementace programátorů, tak, aby byl návrh v odpovídající kvalitě, s důrazem na efektivitu pro dodržení rámce rozpočtu. V neposlední řadě se stará o následnou optimalizaci systému a zajištění jeho bezchybného chodu ve spolupráci s testerem. IT analytik je tedy role, která spojuje zadavatele s programátory, koordinuje proces vývoje a pomáhá ze značné části k realizaci celého projektu. Mezi jeho 29roce set patří různé techniky modelovaní systému, řešení potřeb zákazníka spojené s vyvinutými komunikačními vlastnostmi, silné logické a analytické uvažování, které podporuje kvalitu vývoje projektu. (Martinů, 2015)

Primární kompetence IT analytika:

- Zajistit kvalitu pomocí definice rozhraní, funkce softwaru a jeho chování.
- Pevně specifikovat požadavky a jednotlivé úkoly pro vývoj systému.
- Vydefinovat jednotlivé typy uživatelských rolí.

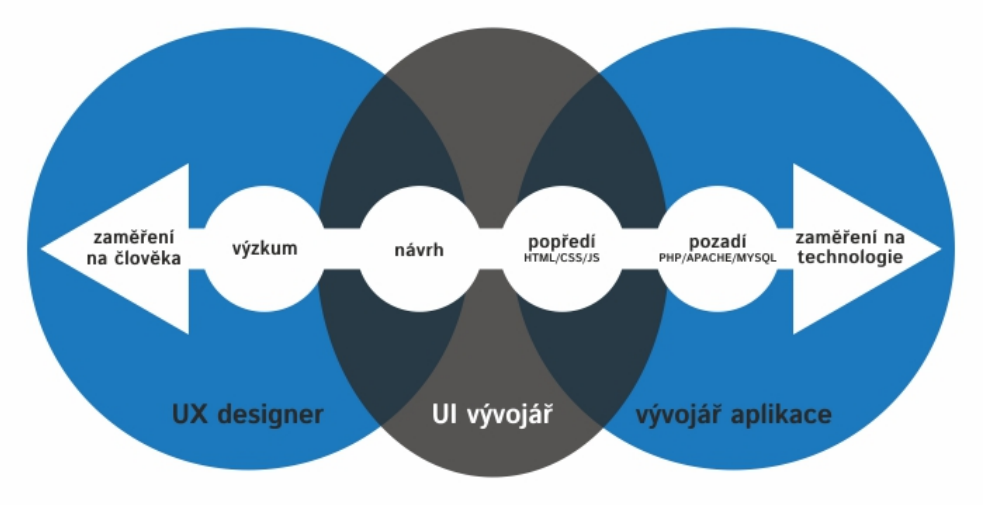
1.3.4 Designer

Designer, nebo také návrhář plní kreativně technickou úlohu. Má za úkol zpracovat vizuální stránku vyvíjeného softwarového produktu. Tato role se může dělit na 2 další, kterými je UI a UX designer. Většinou se ovšem těchto rolí ujme jeden specialista, který ovládá oba obory návrhu softwaru. UI designer má na starosti vizuální koncept celku. UI vychází ze zkratky user interface. Do češtiny přeloženo jako uživatelské rozhraní. Jeho úkolem je tedy volit správné barvy, fonty, styly jednotlivých komponent na stránce a tvoří tak celkový vizuální dojem vyvíjené platformy. UX designer, neboli User eXperience designer má za úkol analyzovat interakci mezi uživatelem a softwarovým produktem a vytvořit tedy co nejlepší

uživatelské prostředí. Mezi jeho úkony patří tvorba uživatelského výzkumu, hloubkové rozhovory, tvoření person a nebo třídění karet. Nezaměřuje se na pouhý produkt, ale i na jeho okolí, a tedy jestli vzniká v podhoubí, které je pro něj příznivé. (Martinů, 2015)

Primární kompetence designera:

- UX designer tvoří vhodné uživatelské podhoubí, aby byl software přívětivý.
- UI designer se stará o vizuální stránku softwaru a jde tedy ruku v ruce s UXD.



Obrázek 4 – Diagram rolí vývoje softwaru

Zdroj: (Brázda, 2020)

1.3.5 Databázový specialista

Databázový specialista je odborníkem zaměřeným na databázové technologie jako je například MS SQL, Oracle, Sybase, a další. Dle vydefinovaných podmínek pro návrh informačního systému rozhodne o výběru databázového prostředí. Je zodpovědný za chod a zpracování databázového prostředí, zpracování modelu softwarového produktu na implementační úrovni. Instaluje a připravuje vývojové, testovací a produkční prostředí. Tyto prostředí připraví jak pro svůj vývojový tým tak i na straně zákazníka. V neposlední řadě je zodpovědný za optimalizaci databázových serverů, na kterých systém běží. (Filipova, 2018)

Primární kompetence databázového specialisty:

- Navržení databázového prostředí na implementační úrovni systému.
- Nachystání a instalace vývojových, testovacích a produkčních prostředí.
- Optimalizace výkonu databázových serverů.

1.3.6 Programátor

Programátor neboli vývojář je srdcem celého procesu. Je to člověk, který buduje software, tak jako zedník zeď. Je odpovědný za tvorbu kódu softwarového produktu, tak, aby byl kvalitní, konzistentní, udržitelný a zcela přehledný. Konzistence a udržitelnost přináší dlouhou životnost kódu, který bude i po několika updatech verzí stále plně funkční. Od přehlednosti očekáváme, že bude kód zřejmý i pro zcela nezasvěceného programátora. Vývojáře můžeme rozdělit do dvou kategorií, kterými jsou backendový (BE) a frontendový developer. (Morris, 2020)

BE programátorovi můžeme být vděční za vše co nevidíme. Jeho úkolem je připravit logiku celého programu na pozadí tak, aby správně komunikoval se serverem. Primárně se tedy zaměřuje na optimalizaci, čistý chod aplikace – posílání, potažmo ukládání správných dat z a do databáze a následnou komunikaci s prohlížečem například přes JSON (JavaScript Object Notation) formát. Je schopný se orientovat v logice kódu a usměrňovat veškeré procedury v rámci aplikace. (Morris, 2020)

Obrázek 5 – FE versus BE programátor

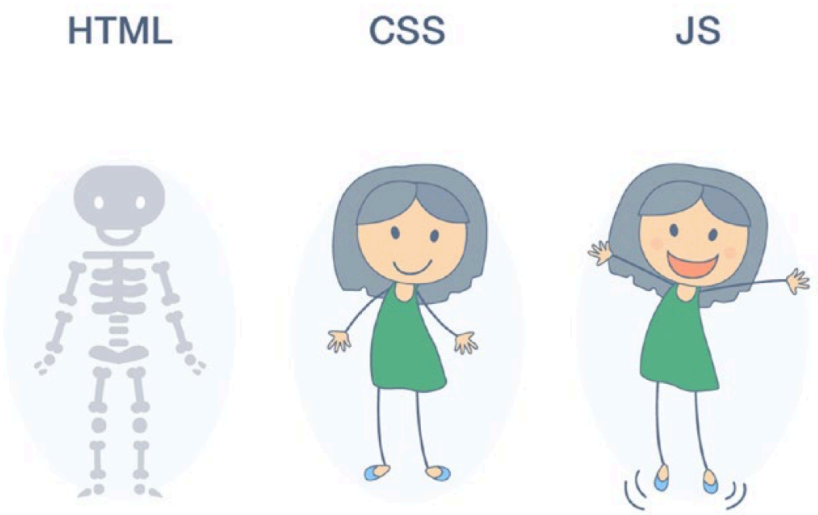
Zdroj: (Morris, 2020)

FE programátor tvoří vše, co je vidět po načtení aplikace. Vše začíná přímou kostrou, strukturou rozhraní programu, kterou je HTML (Hypertext Markup Language). Na kostru se dále váže vizuál, který je tvořený technologií CSS (Cascading Style Sheets) a pomáhá tak definovat styly celé struktury. V neposlední řadě se vývojář frontendu zabývá psaním JS (JavaScript), který určuje změny na pracovním poli aplikace v čase, zároveň je určen pro

komunikaci s backendem, tedy změn dat v programu a případné animace, či responzivitu. Toto jsou 3 pilíře, se kterými pracuje frontendový specialista a tvoří tak vizuální stránku programu. (Filipova, 2018)

Primární kompetence programátora:

- Vývojář musí psát udržitelný kód, který bude odpovídat aktuálním metodám.
- Kód musí obsahovat podmínky bezpečnosti a všechny vyžadované funkce.



Obrázek 6 – HTML / CSS / JS

Zdroj: (Filipova, 2018)

1.3.7 Tester

Softwarový tester je zapojen do vývojové fáze testování – zajištění kvality a následného nasazení softwaru na produkční prostředí. Zastupuje vývojářskou firmu a je garancí kvalit dodávaného softwarového produktu. Je zmocněn provádět manuální, či automatizované testy. Tvoří tak zpětnou kontrolu pro vývojáře a dokáže zastavit chyby ještě předtím, než se dostanou k uživateli. Tato role je nedílnou součástí celého procesu vývoje softwaru, protože může zamezit velkým časovým prodlevám a možným finančním škodám. Role testera se dlouhodobě používá skrze mnohá odvětví. Je to z důvodu zachování kvality produktu. Tuto funkci by měl plnit člověk, který je plně informován o potřebách zákazníka a má přesné povědomí o všech funkcionalitách aplikace. Tester je v pozici, kdy často přichází na nové inovativní nápady, protože ví, jak produkt funguje a jaké chování by očekával v následujících iteracích nasazení na testové či produkční prostředí. Proto je v úzké

komunikaci s analytikem, který může jeho návrhy komunikovat dál na zadavatele nebo přímo vývojáře. Tester je tedy zásadní rolí, která dokáže eliminovat značné škody a případné vícepráce. (Filipova, 2018)

Primární kompetence testera:

- Kompletní protestování všech uživatelských rolí napříč aplikací.
- Příprava testovacích manuálních, či automatizovaných procedur.
- Komunikace se zadavatelem pro udržení myšlenky business logiky.
- Vytvoření dokumentace testovacích scénářů a zapisování BUGů do systému.

1.3.8 Tvůrce dokumentací a školitel

Tvůrce dokumentací připravuje veškeré podklady k finální podobě softwaru. Zpracovává například uživatelské příručky pro jednotlivé role (Administrátor, Editor, Reader, apod.). Dále se podílí na informačních a marketingových podkladech, které podporují rozšíření softwaru s ohledem na prodej nebo získání nových uživatelů produktu. V neposlední řadě připravuje řadu tiskových materiálů, či prezentací určených pro školení. Jako školitel provádí uživatele napříč systémem za pomocí předem připravené struktury školení. Podílí se na sběru okamžitých feedbacků od uživatelů a následný report analytikovi, případně přímo vývoji. Dokáže tedy vnímat co by uživatel vyžadoval za nové funkcionality. Dále se stará o aktualizaci materiálů pro školení a přeškolování jak zákazníka tak i uživatelů. (Martinů, 2015)

1.3.9 Uživatelské podpora

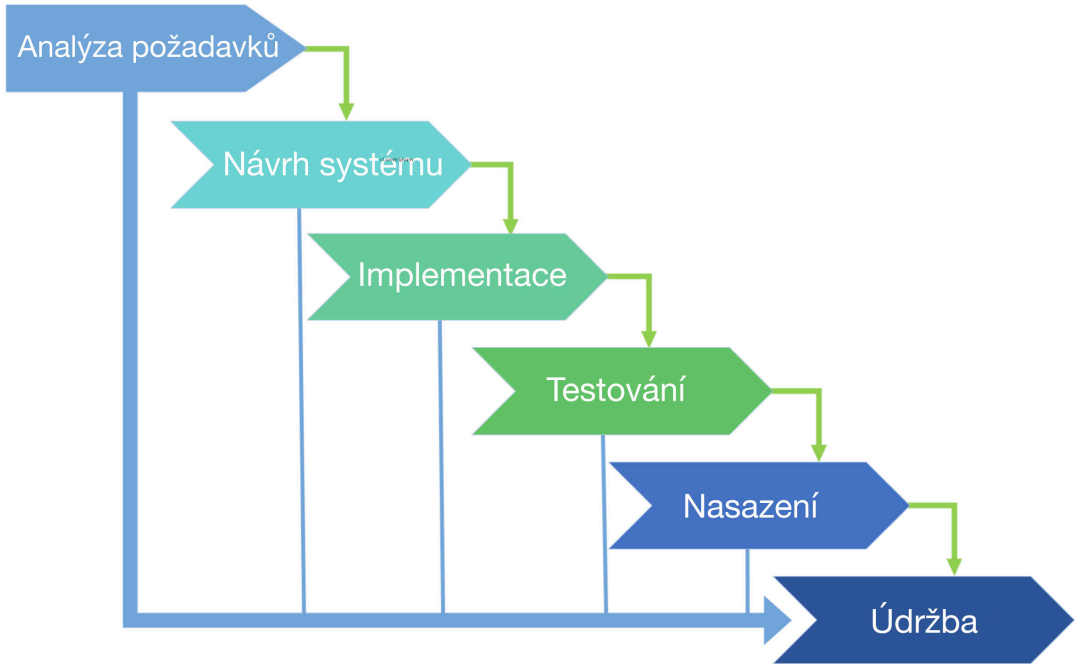
Role uživatelské podpory je významnou součástí celého týmu. Nositel této role musí mít maximální přehled v rámci vyvíjeného softwaru. V mnoho případech se jedná o osobu, která v minulých fázích vývoje působila jako tester. Je to z důvodu skvělého přehledu napříč aplikací. Tato schopnost umožňuje hbitě reagovat na dotazy ze strany uživatelů. Pokud nezná odpověď, tak má daleko snazší cestu ke konzultaci problematiky přímo s IT analytikem, popřípadě celým vývojovým týmem. (Martinů, 2015)

1.4 Metodiky aplikované při vývoji softwaru

Moderní metodiky vývoje softwaru jsou v dnešní době nezbytnou potřebou. Mohou být vnímány jako odlišné cesty na mapě vedoucí ke společnému cíli. Obor softvérového inženýrství nám v dnešní době umožňuje plnit zadané úkoly ve vysoké kvalitě. Metodiky, které známe nám však pomáhají zvyšovat efektivitu týmu a zrychlovat celkový proces vývoje, a proto využíváme odlišné přístupy k řešení různých projektů.

1.4.1 Waterfall model

Waterfall model je kaskádovým přístupem k životnímu cyklu vývoje softwaru, ve kterém se postupuje chronologicky, dle definovaných kroků. Po uzavření jedné fáze se následně přechází do další. Vývoj prochází fázemi analýzy, projektování, realizace, testování, implementace a údržby. Celkový proces tohoto přístupu je pečlivě dokumentován a zachází s předem nadefinovanými komponenty, které mají z každé fáze vycházet. Znázorňuje jakýsi ideální stav posloupnosti na sebe jednotlivě navazujících fází, které nepodléhají cyklickým návratům zpět. Teorie o tomto modelu mluví zcela jasně, ale ze zkušenosti vím, že není vždy, tak jednoduché tento přístup vývoje softwaru dodržet. (Existek, 2017)



Obrázek 7 – Waterfall model

Zdroj: (Existek,2017), vlastní zpracování

Tabulka 1 – Výhody a nevýhody modelu Waterfall

Výhody	Nevýhody
Zcela zřejmý na pochopení.	Zákazník vidí spustitelnou verzi až po uplynutí celého vývojového cyklu.
Snadné řízení jednotlivých fází.	Nejistota spojená s vysokým rizikem.
Ideální pro malé nebo střední projekty.	Tento přístup není vhodný pro komplexní a OOP projekty.
Chronologický přístup vývojových fází.	Není vhodné pro rozsáhlé projekty.
Vhodné pro projekty s ne zcela jasnými požadavky.	Průběh fází není jasně měřitelný pokud je software stále ve vývoji.
Je snadné definovat klíčové body ve vývojovém cyklu. Je tedy zřejmé jaké úkoly mají prioritu.	Integrace se provádí až na samém konci, což znemožní identifikovat problém předem.

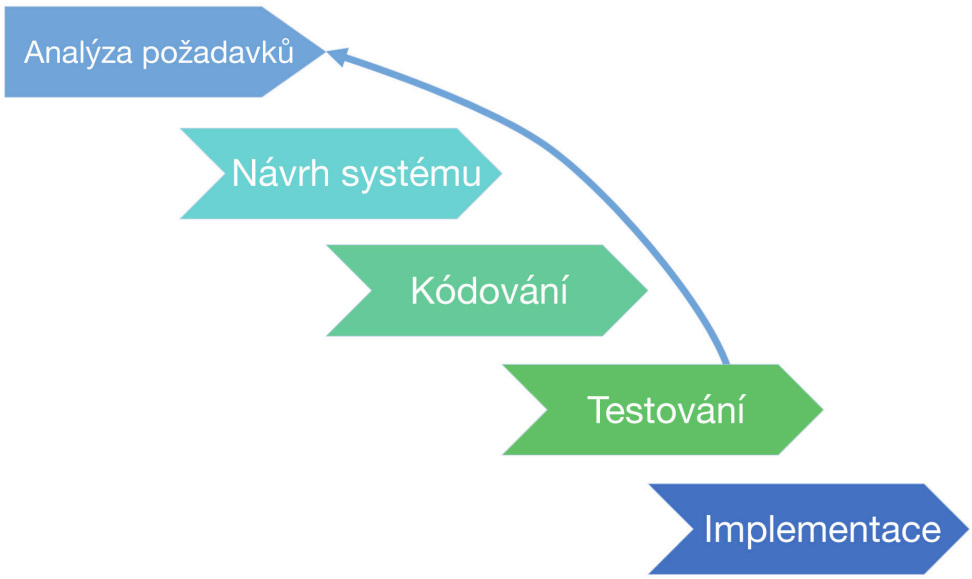
Zdroj: vlastní zpracování

Případy použití pro model Waterfall:

- Přehled používaných technologií je předdefinován. Nelze dynamicky měnit.
- Klient není schopen stanovit požadavky jednoznačně předem.
- Požadavky jsou přesně dokumentovány.
- Projekt je krátkodobý.

1.4.2 Iterativní model

Iterativní model dlouhodobého životního cyklu vývoje softwaru nepotřebuje zcela úplný seznam požadavků na projekt. Proces vývoje je zaměřen na funkční části, které je možné rozšiřovat v pozdějších iteracích. Proces vývoje se opakuje, a proto je možné vytvářet nové verze s každým dalším cyklem. Každá iterace nabývá délky od 2 do 6 týdnů. Každý z nadefinovaných iteračních cyklů umožňuje vývoj nové funkcionality, či komponenty, která je následně implementována do funkčního celku aplikace. Zadavatel je přímým účastníkem vývoje a členem týmu. Nachází se v roli Vlastníka produktu, který může být součástí všech jednání nebo minimálně definovat a schvalovat jednotlivé milníky vývoje. Iterativní model je dynamičtější a klade vyšší nároky na řízení než model Waterfall. (Buchalcevová, 2015)



Obrázek 8 – Iterativní model

Zdroj: vlastní zpracování

Tabulka 2 – Výhody a nevýhody Iterativního modelu

Výhody	Nevýhody
Progres je zcela zřejmý a viditelný.	Není vhodný pro malé projekty.
Některé funkce jsou hbitě vyvinuty už na začátku životního cyklu vývoje.	Je vyžadováno více členů týmu pro rychlejší realizaci v iteracích.
Kratší iterace nám umožňují lépe testovat a opravovat chyby.	Je zapotřebí řízení týmu projektovým manažerem.
Jednoduchost kontroly rizik. Kritické úkoly jsou odladěny jako první.	Vyšší nároky na řízení projektu a změn během jednotlivých iterací.
Přináší větší flexibilitu týmu i zadavateli.	Může dojít ke změně návrhů nebo architektury s ohledem na dynamicky se vyvíjející požadavky.

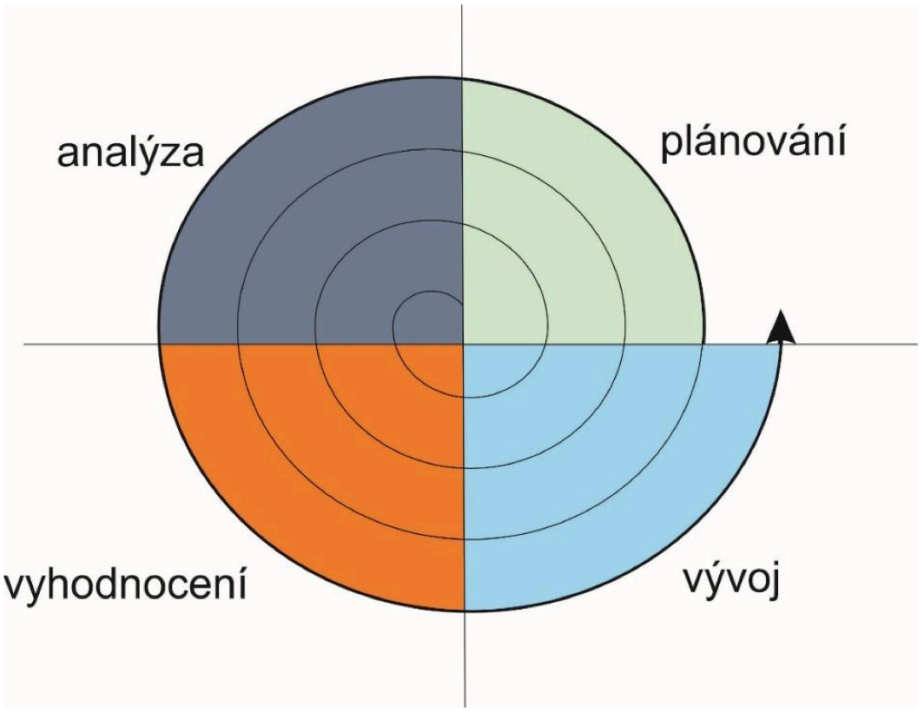
Zdroj: vlastní zpracování

Případy použití pro iterativní model:

- Svůj potenciál využije u větších projektů.
- Umožní zadavateli větší angažovanost do procesu.
- Požadavky na finální stav produktu jsou jasně předdefinované.
- Primárním úkolem je nadefinovat hlavní kostru aplikace, ale podrobnosti mohou s postupem času modifikovat.

1.4.3 Spiral Model

Spirálový model pracuje s kombinací návrhu architektury a prototypování, které probíhají v kontinuálních etapách. Spiral model sdružuje přístupy waterfall a iterativního modelu s velkým důrazem na analýzu rizik. Nejzásadnější překážkou při aplikaci spirálového modelu může být definování okamžiku a následný přechod z jedné fáze do další. Pro řešení tohoto problému se přistupuje k předběžně nastavenému časovému rámci. Tyto časové segmenty jsou vždy naplněny i v případě, že komponenty z předchozích části nejsou zcela vyhotoveny. Plán je definován na základě statických údajů zanalyzovaných analytikem a týmem vývojářů. Proto je v tomto případě každá zkušenost týmu přínosem a dopomáhá k přesnějším odhadům. (Existek, 2017)



Obrázek 9 – Spiral model

Zdroj: (Martinů, 2015)

Tabulka 3 – Výhody a nevýhody Spiral modelu

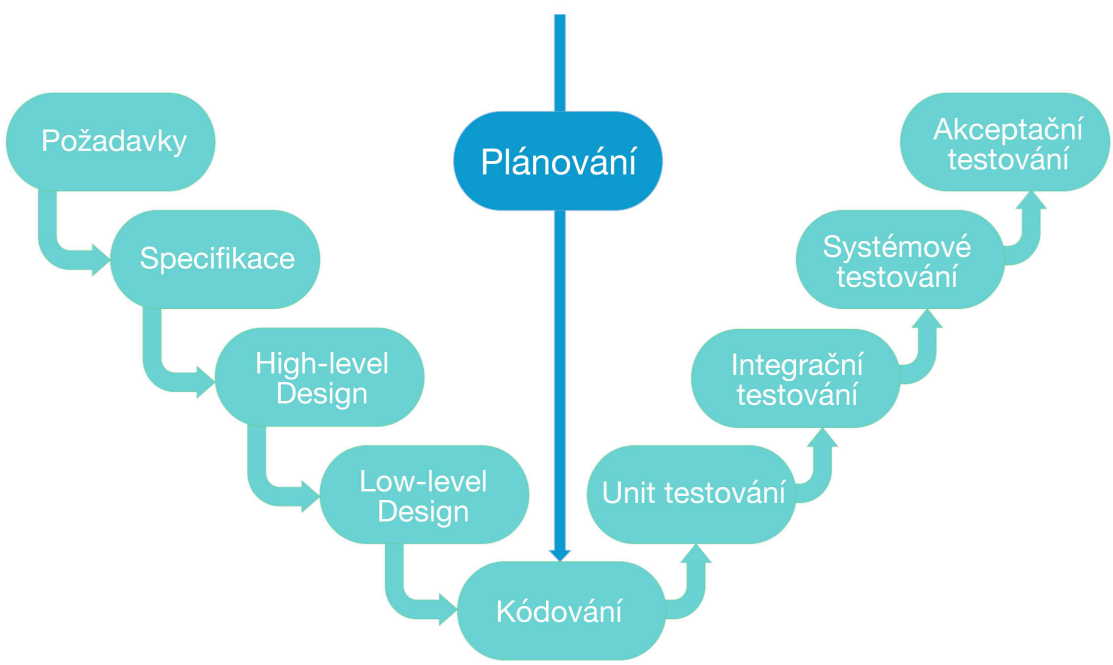
Výhody	Nevýhody
Proces vývoje je precizně dokumentován.	Může být poněkud finančně náročný.
Již první prototypy upozorní na možné nedostatky.	Kontrolu rizik musí řídit kvalifikovaní odborníci se zkušenostmi.
Možnost přidávat nové funkce i pozdních fázích.	Někdy se setkává s neefektivitou.

Případy použití pro Spiral model:

- Když zákazník nezná všechny detailní požadavky.
- Jsou očekávány velké úpravy během vývojového cyklu.
- Vhodné pro projekty s vysokým rizikem. Vstupujeme do vývoje s tím, že chceme rizika co nejvíce eliminovat. Z toho může vyplývat větší časová náročnost.
- Produkt, který je vydáván získá dostatek zpětných vazeb v několika fázích.

1.4.4 V-shaped model

Model životního cyklu vývoje softwaru nazývaný V-Shaped vychází z Waterfall přístupu. Zároveň je jeho rozšířením o konkrétní testovací moduly pro jednotlivé vývojové fáze. Jde o velice striktní model s koncentrací na kvalitu softwarového produktu. Do další vývojové fáze se může přejít až v momentě když je vše detailně otestováno a odladěno. Někdy je tento model také nazývaná jako „ověření a validace“. S každou fází přichází konkrétní přístup k řízení. Tento model je vhodné volit když chceme od projektu maximální kvalitu bez ohledu na čas. (Buchalcevová, 2015)



Obrázek 10 – V-Shaped model

Zdroj: (Existek,2017), vlastní zpracování

Tabulka 4 – Výhody a nevýhody V-shaped modelu

Výhody	Nevýhody
Striktní pravidla a výstupy.	Není flexibilní.
Zaměřený na kvalitu produktu.	Špatná volba pro malé projekty bez vize.
Testování o ověření správnosti vývoje Přichází již v raných fázích.	S ohledem na detailní testování mohou přicházet časové prodlevy
Vhodné pro projekty kde je vše jasně předdefinované.	Relativně velké riziko neefektivity.

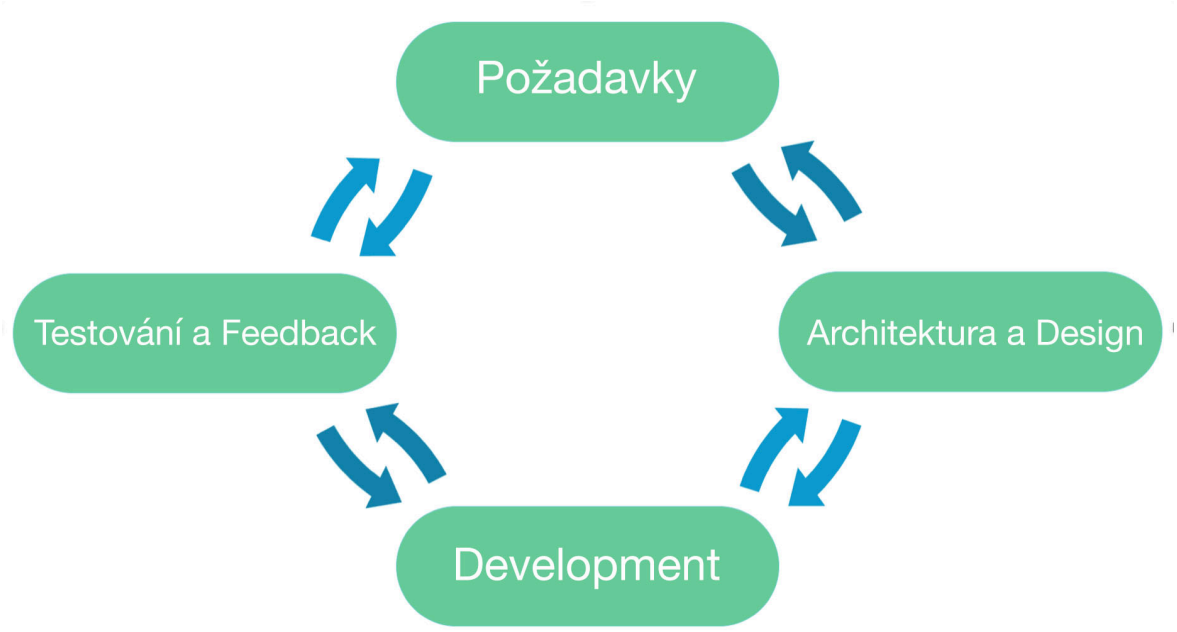
Zdroj: vlastní zpracování

Případy použití pro V-shaped model:

- Vhodné pro projekty kde se vyžaduje detailní testování každé komponenty.
- V zásadě se využívá na středně velkých projektech s jasnou strukturou a cíli.
- V momentě když je tým zkušených vývojářů pro úplné odladění aplikace.

1.4.5 Agilní model

Agilní metodika vývoje softwaru umožňuje zákazníkovi vidět po každé iteraci výstup, který jasně definuje progres v rámci jednotlivého sprintu zadávané práce a celkového procesu vývoje softwarového produktu. Vývoj se odehrává v předem stanovených cyklech, které nesou v IT světě název vývojové sprinty. Tento časový úsek se může nacházet v intervalu od 2 do 6 týdnů. Po každém uplynutí sprintu se sejde zadavatel s vývojovým týmem. Na tomto meetingu se řeší shrnutí minulého sprintu, retrospektiva a následná definice vývojových úkolů pro další sprint. Zákazník je tedy schopen udělit okamžitý feedback a projevit spokojenost, či nikoliv. To je jednou z masivních výhod agilního vývoje softwaru. Nevýhodou může být absence přesného definování požadavků. Pokud zákazník nemá ve svém týmu silného produkt ownera, který ví co chce, tak to může projekt značně zpomalit a přinést jistý druh nejistoty pro budoucí vývoj softwaru. Pro IT analytika může být také obtížné nastavit cenovku jednotlivých vývojových úkolů s ohledem na neustále se dynamicky vyvíjející požadavky, či změny v komponentech. Díky častému kontaktu zákazníka s vývojovým týmem je celý tento proces agilní metodiky postavený na otevřené komunikaci, hledání společného cíle a kompromisu. (McKay, 2019)



Obrázek 11 – Agilní model

Zdroj: (Existek,2017), vlastní zpracování

Tabulka 5 – Výhody a nevýhody Agilního modelu

Výhody	Nevýhody
Projekt je rozdělený do transparentních a ßkrátkých iterací, které zadavateli jasně představují stav vývoje aplikace.	Vznikají problémy s vyměřením finálních nákladům kvůli stálým změnám v rámci vývoje.
Riziko vývoje projektu je minimální.	Možnost překročení nadefinovaného času.
Rychlé první vypuštění na produkční prostředí. Připraveno pro uživatele.	Tým musí být vysoce kompetentní a musí umět naplnit požadavky zákazníka.
Opravy funkcionalit jsou implementovány v rámci vývoje.	Nové požadavky mohou narušit chod stávající architektury.

Zdroj: vlastní zpracování

Případy použití modelu Agile:

- V momentě když se potřeby uživatelů a klienta dynamicky mění.
- Nevyžaduje detailní plánování projektu jako například u waterfall modelu. Stáčí pouze určitý rámec a předpokládaná vize projektu a může se začít vyvíjet.
- Nižší cenovka za nasazení na produkční prostředí s ohledem na velké množství iterací.

2 Základní pojmy a postupy při návrhu UX/UI

Tato kapitola se bude věnovat přiblížení základních pojmů a procesu při návrhu designu nového softwarového řešení. Vydefinujeme si jednotlivé pojmy jako je UX a UI. Přiblížíme si společné a rozdílné množiny těchto dvou oborů. Zahrneme fáze pro návrh komplexního uživatelského prožitku. V závěru zhodnotíme přínos kvalitního UX návrhu.

2.1 Definice pojmu User eXperience (UX)

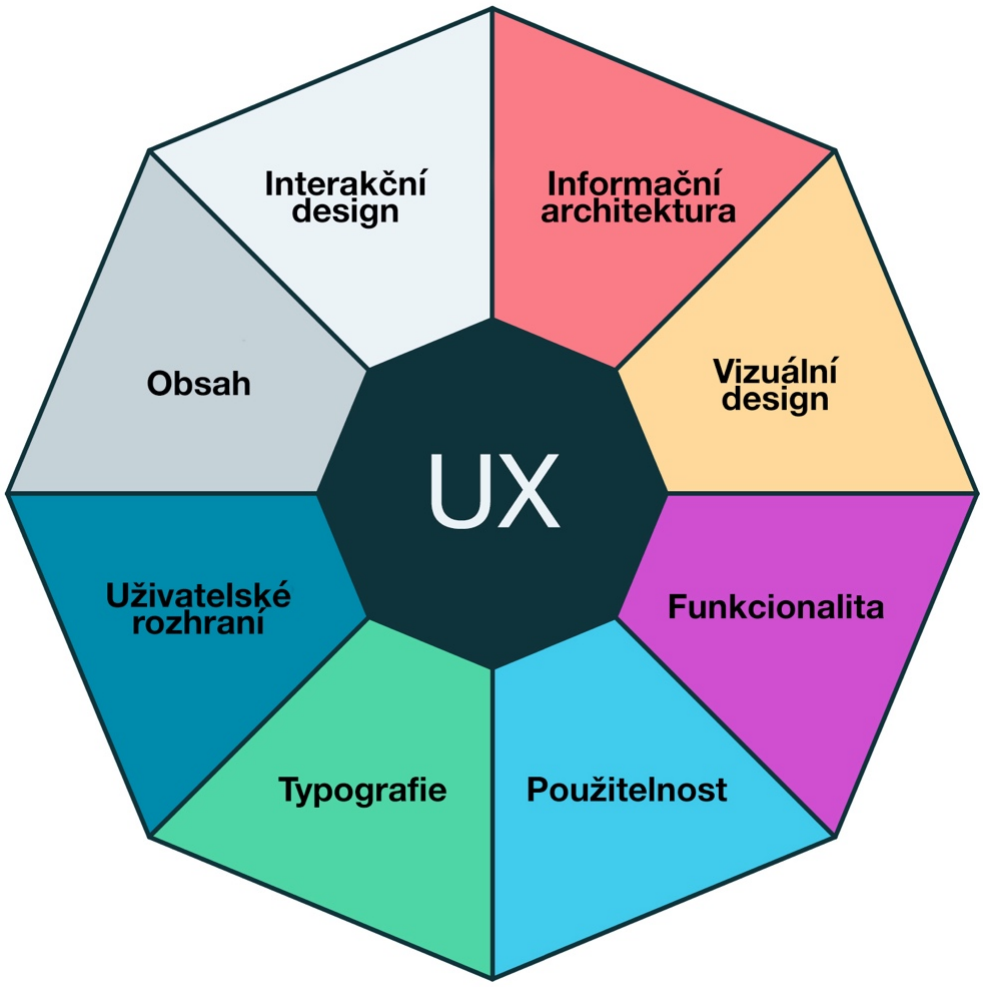
Pojem user experience je do českého jazyka v literatuře často překládán jako uživatelský prožitek. Já osobně se přikláním k českému slovu "dojem", který dle mého názoru problematiku vystihuje o něco lépe. Co ovšem tento pojem znamená? Přesto, že je uživatelský prožitek stále dynamicky vyvíjející se disciplínou, tak vychází z koncepce, která je do značné míry stejná. Mezi tyto hlavní faktory přispívající k příznivému uživatelskému prožitku patří například: použitelnost, interakční design nebo informační architektura. Osobně bych uživatelský prožitek popsal jednou větou jako motivaci uživatele setrvat na navštívené webové stránce s úspěšným nalezením informace, anebo zakoupení požadovaného produktu. Dle konzultanta UX Christian Jansen, ze společnosti Sun Microsystems se dá pojem UX popsat následovně:

„Uživatelský prožitek je prožitek jednotlivce užívajícího určitý výrobek nebo službu. Z pohledu uživatele má návštěva webové stránky vždy nějaký účel. Například: pro pronajmutí auta, zakoupení knihy nebo vyhledání určité informace. Pokud chceme zaručit, aby byl uživatelský prožitek pozitivní, musíme porozumět, kdo vlastně uživatel je, co potřebuje a v jakém kontextu zamýšlí použít výrobek či službu. Důkladné pochopení potenciální cílové skupiny nám napomáhá definovat požadavky na výrobek a pochopit, jaké vlastnosti v očích uživatele zvýší jeho hodnotu."

Ze slov odborníka vyplývá, že je důležité soustředit energii na detailní poznání a analýzu zákazníka pro maximální naplnění jeho potřeb. Ať už se jedná o softwarový nebo jiný produkt, či službu.

Jeff Johnson, prezident společnosti UI Wizards, napsal: *„Uživatelský prožitek je přesně to, co název napovídá. Všechno, co uživatel vidí a s čím se potýká, když stránku navštíví a chce ji vyzkoušet. Nenáleží sem pouze struktura stránky a její obsah, ale také to, jak uživatel stránku najde, zda funguje v jeho prohlížeči nebo mobilním zařízení, zda stránka poskytuje pomoc těm, kdo se setkají s problémem, atd. Vše musí fungovat dobře, jinak nebude stránka z uživatelského hlediska úspěšná. Pokud nefunguje, navštíví uživatel stránku jinou".*

Dle citovaných definic můžeme slova User eXperience vnímat jako spojení metod, technik a zásad, které nám umožňují tvořit celkový požitek nebo dojem z používání webového portálu, či aplikace. Toto využívání v nás vyvolává jisté pozitivní či negativní emoce a na základě těchto pocitů se na stránku vracíme nebo nikoliv. Je nám známo velké množství metod, dle kterých jsme schopni přispět ke kladnému zachování emocí po návštěvě nebo užití softwarového produktu. Ty si na následujících stránkách nastíníme a rozebereme jejich definice.

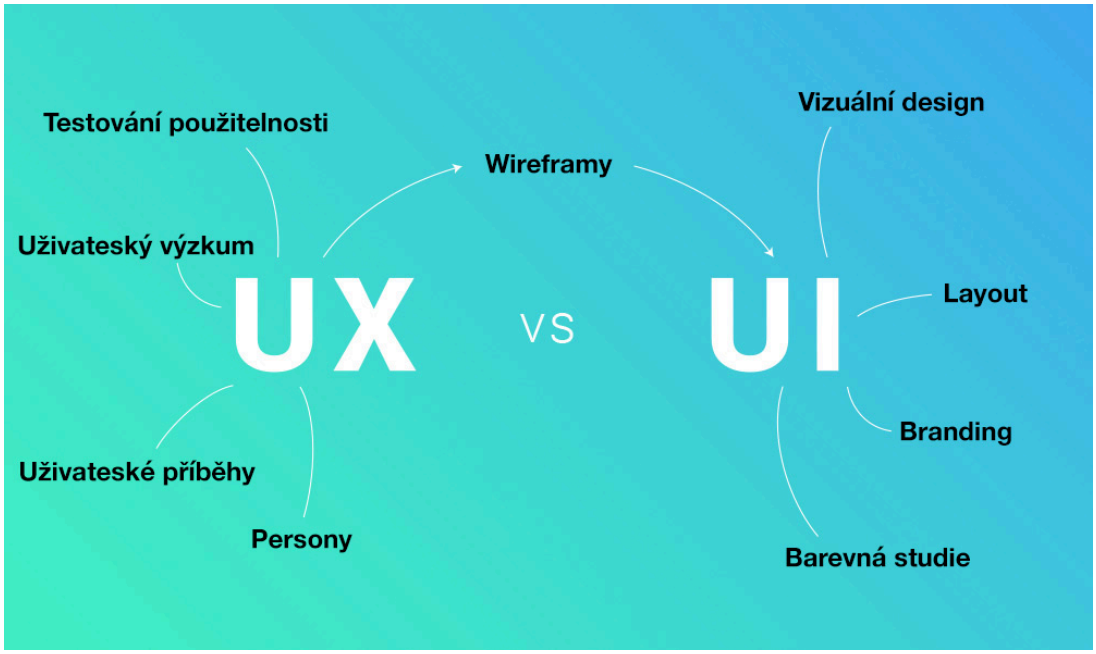


Obrázek 12 – User eXperience koncept

Zdroj: (Graphic design junction, 2020)

2.1.1 User Interface (UI)

User Interface (UI) a User eXperience (UX) jsou často zaměňované termíny. Jejich směry jsou odlišné, ale jeden bez druhého nedokáže existovat. Doplňují se v rovině vizuálního zpracování a logiky pod ním schované. Společný průnik hledají ve wireframech, kde je spjat logický celek aplikace s prvními prototypy návrhu softwaru. UI, neboli uživatelské rozhraní dává produktu vizuální tvář, která jde ruku v ruce se strukturou definovanou návrhářem uživatelské přívětivosti. Proč na uživatelském rozhraní záleží? Uživatel komunikuje s počítačem přes uživatelské rozhraní. Dochází ke společné interakci, která je buď přívětivá, či nikoliv. O podporu přívětivosti se stará i uživatelské rozhraní, které tvoří layout stránky, celkový branding a dává dohromady smysluplné barevné kombinace. Při navrhování připraveného uživatelského rozhraní dbejme na barvy, symetrii, fokus na důležitá sdělení, ikony a různé typy políček, posuvníků a tlačítek. Jednou z otázek, kterou je vhodné při návrhu pokládat je: „Co uživatel hledá a jak ho k danému místu můžeme nasměřovat?“
Dalším neméně důležitým faktorem je branding, neboli identita značky, produktu nebo softwaru. Branding nám umožňuje budovat strategii a podporuje vizi značky. Má schopnost odlišit produkt od konkurence. Vyvolává v lidech emoci, kterou mají s produktem spojenou. Je jakousi duší firmy, produktu, anebo služby. (Michl, 2016; McKay, 2013)



Obrázek 13 – UX vs UI aktivity návrhu rohraní

Zdroj: (Kuipers, 2020), vlastní zpracování

2.1.2 Interakční design

Interakční design můžeme znát také pod zkratkou IxD. Tento obor nám napomáhá porozumět a tvořit spojení mezi produktem a jeho uživatelem. Spojení nám reprezentuje jak fyzickou, tak i emocionální vazbu, která je ve většině případů tou silnější. Tvoří totiž jasný vztah k produktu, který je buď pozitivní nebo negativní a určuje, zda bude produkt nadále využíván, či nikoliv. (Siang, 2020)

Další z důležitých přístupů interakčního designu je otázka předpokladu chování uživatele. Interakční design se snaží nahlédnout do přístupů k použití daného produktu. Přemýšlet o kreativitě uživatele a tvořit myšlenkové mapy, které budou popisovat operace mezi produktem a uživatelem. Tak aby byl uživatel naplněný a produkt byl dostatečně uživatelsky přívětivý, tak musí komunikace mezi zařízením a uživatelem probíhat zcela čistě a plynule. Pokud se na cestě k užití produktu objevují překážky, tak to snižuje na kvalitě daného zařízení. Primárním cílem interakčního designu je odstranit veškeré překážky k užití zařízení, softwaru nebo produktu jiného typu, tak, aby interakce proběhla zcela hladce a uživatel produktu odcházel s pocitem vítězství. Jen po takové pozitivní zkušenosti bude mít motivaci k opětovnému užití. (Anderson, 2012)

Pro podpoření orientace uživatele na stránce by měl interakční design pracovat s pravidly a metody definovanými v knize Human interface Guidelines, která shrnuje pokyny pro vhodné rozložení jednotlivých komponent tam, kde by je uživatel očekával. Mezi nejzásadnější pravidla, které vycházejí od profesora Gillian Crampton Smith, působícího na London's Royal College of Art a profesionálního interakčního návrháře Kevin Silver patří pět dimenzí. Tyto dimenze zvažuje každý zkušený interakční návrhář a pomáhají mu tak při orientaci a usměrnění návrhu:

• **1D – Slova –** Slovo jako takové je velmi mocný nástroj, který dokáže uživatel vnímat téměř okamžitě. Proto je vhodné se slovy a textem zacházet vhodně a opatrně, tak aby uživatel získal informaci, která je srozumitelná, stručná a zcela zřejmá. (Siang, 2020)

• **2D – Visuální zobrazení –** Dimenze vizuálního zobrazení pracuje s kompletním grafickým zpracováním, které obsahuje vnímání typografie, barev, fotek, videí, ilustrací, ikon a spoustu dalších komponent z vizuálního segmentu. Tento vizuální obraz slouží jako doplnění a podpoření slov. Uživatel téměř okamžitě ví co mu tento zdroj informací chce sdělit, a proto je to velmi rychlá a jasná forma komunikace. Z historického hlediska člověk pracuje s obrazovou formou sdělování informace již od doby kamenné, kde byly používány

například jeskynní malby. Proto se vizuální forma sdělování informací fixuje do dlouhodobé paměti a umožňuje tak uživateli vnímat smysl některých znaků, či ikon. (Siang, 2020)

• **3D – Fyzické objekty a prostor –** Fyzický objekt je médium, prostřednictvím kterého má uživatel možnost integrace se službou, produktem nebo webovým rozhraním. Mezi rozhraní tohoto typu patří například počítač, klávesnice, myš, ale i prsty propojující naše akce s mobilním telefonem. Dalším bodem je prostor, který ovlivňuje míru našeho pohodlí při interakci s produktem. Jako příklad je vhodné uvést například klidnou domácnost bez jediného rušivého elementu a nebo kavárnu plnou diskutujících lidí. V těchto prostorech bude vytvořená zcela jiná atmosféra, která může naše vnímání produktu podpořit nebo zcela utlumit. (Siang, 2020)

• **4D – Čas –** Čas je dimenzí, která umožňuje vnímat proces akcí, které se mění v závislosti na čase. Jedná se o videa, animace nebo dokonce zvuky. Všechny tyto aspekty podporují vnímání produktu. Je vhodné nad těmito prvky přemýšlet, tak aby šli spustit znovu, potlačit, anebo pozastavit. (Siang, 2020)

• **5D – Chování –** Chování je pátá dimenze, kterou doplnil uznávaný interakční návrhář Kevin Silver. Když shrneme všechny čtyři předešlé fáze a jejich jednotlivé akce, tak na ně vzniká bezprostřední reakce od uživatele. Je vhodné vnímat chování a emoce, které uživatel vyjadřuje v rámci užití produktu. (Siang, 2020)

2.1.3 Vizuální design

Vizuální design je další z disciplín User eXperience, která je značně subjektivní, nicméně velmi důležitou součástí. Každý má trochu rozdílný pohled na věc a v oblibě má jiné vizuální zpracování. Vzhled je tedy faktor, kterým se ne vždy můžete zavděčit všem uživatelům. Každopádně se ale můžete soustředit na konkrétnější větší či menší cílovou skupinu a podle toho vzhled produktu, nástroje, či webu přizpůsobit. Vzhled je jednoznačně faktor, který dokáže konzumenta produktu nebo webové stránky jako první udržet v setrvání nebo je ze stránky odradit. Čím lépe je vizuální produkt zpracovaný, tím větší skupinu uživatelů dokáže zaujmout. Kvalitně zpracovaný vzhled také podporuje faktory jako je například důvěryhodnost stránky, exkluzivita zboží nebo seriózност celé společnosti. Ne jednou jsem se z vlastní zkušenosti setkal s firmou, která měla ovšem zanedbanou webovou prezentaci a tento první dojem mě nutil firmu řadit na vedlejší kolej a preferovat konkurenci. Na cestě k vytvoření kvalitního vzhledu je možné vyzdvihnout čtyři body pro ideální zpracování vizuálního designu: (Gordon, 2020)

- **Kontrast** – Kontrast je primárním nástrojem pro upoutání pozornosti diváka. Umožňuje rozdělit závažnosti informací a udržuje pozornost uživatele nad objekty, které jsou podstatné. Pomocí kontrastu lze tedy lépe soudit relativnost informací. Pokud by vizuál neobsahoval žádný kontrast, tak by byl design zcela plochý a pro divákovi oči nudný. (Gordon, 2020)
- **Hierarchie** – nám určuje důležitost jednotlivých prvků na stránce. Efektu určování priority jednotlivých komponent můžeme docílit pomocí první metody vytvoření kontrastu na stránce, ale také díky velikosti písma, barvám, velikosti objektů, atd.
- **Jednotnost** – je klíčovým principem pro fungování uživatelsky přívětivého designu. Udržuje stránku v konzistentní rovině a pomáhá budovat celkový pocit ze stránky jako takové. Stejný formát fontu, komponent a ladění barev je třešničkou na dortu celého zážitku ze stránky. (Gordon, 2020)
- **Jednoduchost** – Po splnění všech předchozích bodů je důležité neopomenout na jednoduchost. Jednoduchost je v mnoha momentech základním klíčem k úspěchu. Je vhodné se zachovat čistý design na místo stránky přeplněné informacemi. Uživatel se může cítit zbytečně zahlcený a nebude ochotný na stránce setrvat. Stejně tak jako kontrast na stránce pozornost přitahuje, tak zahlcenost ji odvádí. (Gordon, 2020)

2.1.4 Použitelnost

Použitelnost je jedním z dalších klíčových faktorů ovlivňujících uživatelský prožitek jako takový. Můžeme jí chápat z pohledu uživatele jak složité nebo snadné je řešit přístup k dané problematice. Použitelnost a její definice je nejvíce spjata se jménem Steve Krug, který je známým autorem knihy Webdesign: Nenuťte uživatele přemýšlet. V této knize definuje dobře použitelný web jako místo, které je pro uživatele zcela srozumitelné, intuitivní a snadno ovladatelné. Proto uživatel neztrácí čas nad úvahou co slouží k čemu a může tedy všechny úkony na stránce provádět zcela bez komplikací. Steve Krug také klade velký důraz na testování použitelnosti, které přirovnává k provádění turistů vaším rodným městem. „To, že někomu ukazujete něco co sami dobře znáte otevírá oči i vám a můžete tak tedy pozorovat nové detaily, kterých byste si za normálních okolností nevšimli.“ Uživatelé stránky nečtou, ale prohlíží je. Nikdy neprovádíme optimální výběr, ale děláme pouhé kompromisy. Nepřemýšlíme nad teoretickou rovinou jak věci fungují, prostě to uděláme. Další body, na které uznávaný UX odborník Steve Krug klade důraz, jsou jeho zákony použitelnosti. Definují nám jak bychom měli s uživateli zacházet. (Krug, 2003)

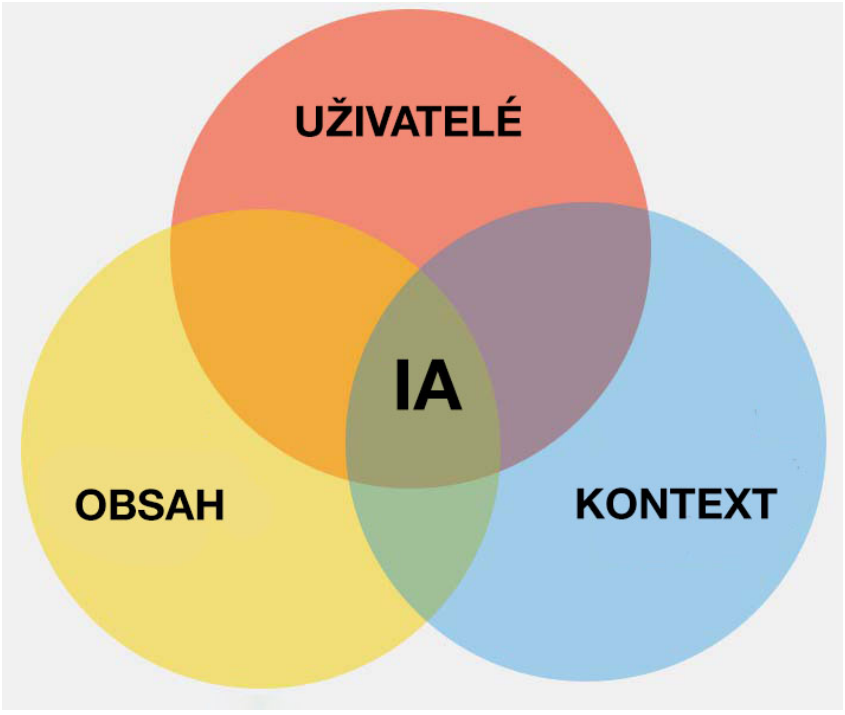
- **Vynechejte zbytečná slova** – Dlouhé a květnaté texty nikoho nezajímají. Uživatel potřebuje dostat jasnou a stručnou informaci. V jednoduchosti je síla. Studie ukázala, že při odstranění poloviny textu na webu nedojde téměř nikdy ke změně kontextu, nýbrž ke zjednodušení a tudíž i zlepšení použitelnosti. Vyčistí se tedy zbytečný šum a sdělení bude jasnější. (Krug, 2003)
- **Navrhujte stránky pro prohlížení, nikoliv pro čtení** – Klíčové je na každé stránce zachovat konzistentní vizuální layout a hierarchii. Zásadní je pracovat s kontrastem, který odděluje důležité informace od těch méně důležitých. Vědec Jakob Niels vytvořil v roce 1997 studii, kde popisuje, že pouze 16% návštěvníků webu čte text slovo od slova. Osobně se domnívám, že aktuální procento bude značně menší. Proto je vhodné využívat zaběhnuté funkcionality a vizuální sdělení, na které jsou uživatelé zvyklí. Nezbytnou součástí webu je jeho mapa, která ukazuje rozvrstvenou strukturu a uživatel ví, kde se nachází a kam se může z aktuálního místa přesunout vpřed nebo vzad. Dále je také nezbytné minimalizovat vizuální šum, kterým je například velké spektrum barev, reklamy, sdělení vyžadující pozornost, atd. Tento bod vyplývá z pravidel interakčního designu, kde popisuji pravidla pro komunikaci s uživatelem. (Krug, 2003)
- **Nenuťte mě přemýšlet** – Poslední a zcela nejzákladnější bod. Již první dojem z webu by v nás měl vyvolat jasnou vizi a záměr, který nám má obsah webu předat. Žádný z uživatelů nestojí o dlouhé přemýšlení nad kontextem webu. Dle článku z Princeton University je prokázáno, že uživatel si vytváří první dojem v pouhých 0,05 vteřinách. Do sedmi vteřin zjistí vše podstatné a hodnotí, zda jsou pro něj informace relevantní, či nikoliv. Hranice deseti vteřin je kritická, kdy se uživatel rozhodne, zda stránku opustí nebo na ní setrvá. Zajímavým poznatkem je, že uživatelé přistupující z mobilních telefonů mají posunutou hranici na dvacet vteřin. Proto by uživatel o použití stránky neměl být nucen přemýšlet a všechny úkony provádět intuitivně. (Krug, 2003)

2.1.5 Informační architektura

Pojem informační architektura poprvé představil Richard Saul Wurmanv v roce 1976. Bohužel tento termín nebyl přijat, a proto upadl na pár let v zapomnění. Následně, ale pojem uvádí kniha Information Architecture for the World Wide Web od autorů Louise Rosenfelda a Petera Morville v roce 1998. Vědní obor informační architektura se zabývá tvorbou struktury webového rozhraní, vytvoření hierarchie a kompletní navigační mapy webu.

Příprava a aplikace této struktury uživateli napomáhá k jednoduššímu přístupu k informacím. Přidává možnost sjednotit logické celky, tak, aby uživatel věděl, kde je má očekávat. Nekvalitně zpracovaná informační architektura zvyšuje náklady na údržbu, školení, a nalezení konkrétní informace. Návrh informační architektury se koncentruje na tři na sebe vázaně navazující pojmy, které literatura prezentuje formou Vennova diagramu. Tyto množiny mají společný průnik a kompletují, tak myšlenku informační architektury. (Rosenfeld a Morville, 2002)

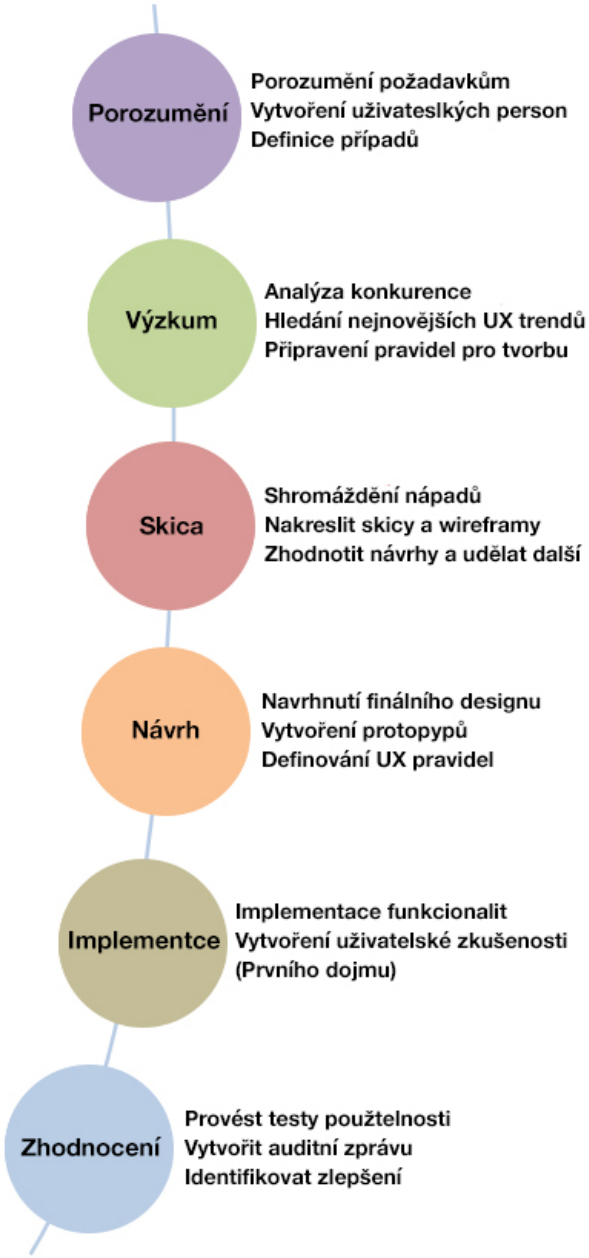
- **Uživatelé –** celá koncepce UX se koncentruje na uživatele, proto je důležité zkoumat jeho chování a potřeby. Informační architektura napomáhá přizpůsobit logické celky, tak aby pro uživatele dávala struktura webu nebo aplikace smysl
- **Obsah –** je za potřebí, aby měl přidanou hodnotu, musí zahrnovat informace, které uživatel vyhledává. Tato sdělení je důležité vhodně formátovat, vytvořit strukturu a zvolit jazyk, kterému uživatel bezpečně rozumí.
- **Kontext –** Do kontextu lze zařadit business cíle společnosti, politiku, kulturu, technologie a omezení, které vznikají v prostředí vyvíjeného produktu. Kontext určuje nadefinované možnosti a mantinely pro tvorbu IA. (SEOMining, 2020)



Obrázek 14 – Vennův diagram definující AI
Zdroj: (SEOMining, 2020), vlastní zpracování

2.2 UX proces

Návrh uživatelského prožitku je iterativní proces, který lze neustále vylepšovat. Je tedy dobré znát jaký chceme mít finální produkt a co od něj očekáváme. V průběhu procesu návrhu procházíme přes jednotlivé fáze, které následně hodnotíme a hledáme možnosti vylepšení. Každá z fází má svá pravidla a přístupy k řešení. Proces návrhu UX zahrnuje následujících šest bodů. (Minhas, 2018)



Obrázek 15 – UX Proces

Zdroj: vlastní zpracování, inspirováno (Minhas, 2018)

2.2.1 Porozumění

Chceme-li poskytovat přístupy k řešení, tak musíme nejdříve porozumět problematice. Proto je žádoucí seznámit nejdříve celý tým designerů s požadavky, tak aby mohli vznikat kreativní a logické přístupy k řešení. Jako první krok je vhodné podstoupit standartní metody uživatelského výzkumu, mezi které patří například individuální a kontextové rozhovory. Dále je vhodné vést diskusi s klientem a prezentovat mu svou stávající práci, případně moodboardy, které by mohly k produktu pasovat. Znalost uživatele vám napomůže k určení jasného směru rozvržení a návrhu. IT analytik je rolí, která získává požadavky od zadavatele, a proto je žádoucí, aby s ním udržoval tým designerů v úzký kontakt. (Minhas, 2018)

Pro tuto fázi je tedy zásadní, seznámit se společně se zadavatelem a případně uživateli aplikace, získat povědomí o jejich potřebách, porozumět uživatelům a jejich prostředí. Následně zanalyzovat požadavky a v závěru definovat persony a případy použití. (Minhas, 2018)

Výstup:

- Uživatelské persony – detailní rozpracování potenciálních uživatelů
- Use Case diagramy – informuje nás o možných případech užití
- User stories – definice funkcionalit, které ctí business logiku

Obrázek 16 – Uživatelské persony

Zdroj: vlastní zpracování, inspirováno (Minhas, 2018)

2.2.2 Výzkum

Výzkum je základním stavebním kamenem k vytvoření uživatelsky přívětivého prostředí. Tým designerů se snaží zajistit více informací ohledně konkurence na trhu, získat inspiraci a řídit se moderními trendy návrhu uživatelského rozhraní. (Minhas, 2018)

„Trvalo mi několik vteřin, než jsem to nakreslil, ale trvalo mi 34 let, než jsem se naučil, jak je nakreslit za pár vteřin.“ (Scher, 2020)

Produktový designer Sherif Amin nazval **3 body UX konkurenční analýzy**:

1. Pochopit konkurenci na trhu.
2. Získat detailní informace k vývoji produktu.
3. Zajistit inspiraci a nápady v konkurenčním prostředí.

Další nezbytnou součástí analytické fáze je získání povědomí o aktuálních trendech a požadavky ze strany uživatelů. Ve fázi výzkumu již začíná tým přemýšlet nad prvními návrhy k realizaci. Z velké částí se jedná o fázi zkoumání a brainstormingu. (Minhas, 2018)

Výstupy:

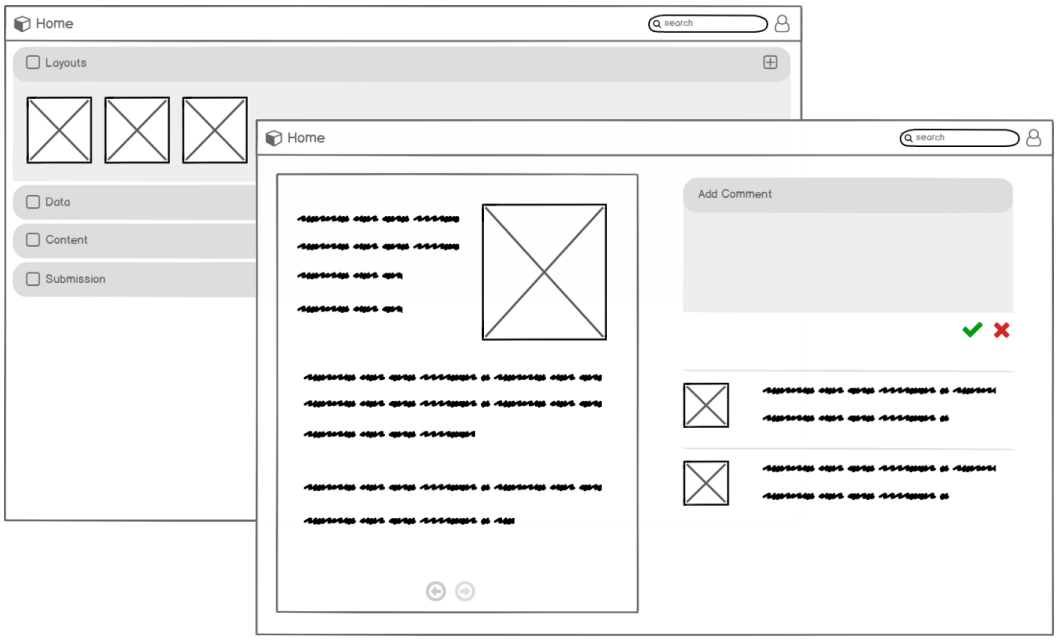
- Detailní průzkum konkurence
- Výzkum požadovaných funkcí na trhu
- Analýza UX/UI trendů
- Definice požadavků na první návrhy
- Velká škála materiálů díky kterým lze začít s realizací návrhu

2.2.3 Skica

Tato fáze je sama o sobě iteračním procesem, který probíhá formou brainstormingu. Má za úkol vytvořit UI (user interface), neboli uživatelské rozhraní. Fáze skicování je postavena na předchozích definicích, které vznikly ve fázích procesů porozumění a analýzy. Důležitou součástí této fáze je testování a zhodnocení wireframů. Primární cíl fáze je vytvořit řadu návrhů a zvolit správnou cestu pro vyladění uživatelsky přívětivého prostředí. Proto se opakuje schvalovací kolečko v rámci návrhů, testování a následném schvalování. (Minhas, 2018)

Výstupy:

- Velké množství návrhových skic
- Wireframy a Mockupy (Digitální návrhy ve kterých je základní layout/rozložení jednotlivých komponent)
- User flows diagramy



Obrázek 17 – Wireframy

Zdroj: (Minhas, 2018)

2.2.4 Návrh

Tato fáze reprezentuje finální kompletaci návrhů. Je vytvořena barevná paleta a návrh vizuální stránky jednotlivých komponent. Zpracovávají se zde předpřipravené wireframy a převádějí se do programového řešení. Po zpracování UI konceptu softwarového produktu je vhodné specifikovat pokyny potřebné k implementaci. V současné fázi se finalizují principy barev, typografie, ikon a dalších vizuálních forem prezentace produktu. Důležitým bodem je konzultace se zadavatelem a společné schválení vizuálu s vazbou na logiku užití. (Minhas, 2018)

Výstupy:

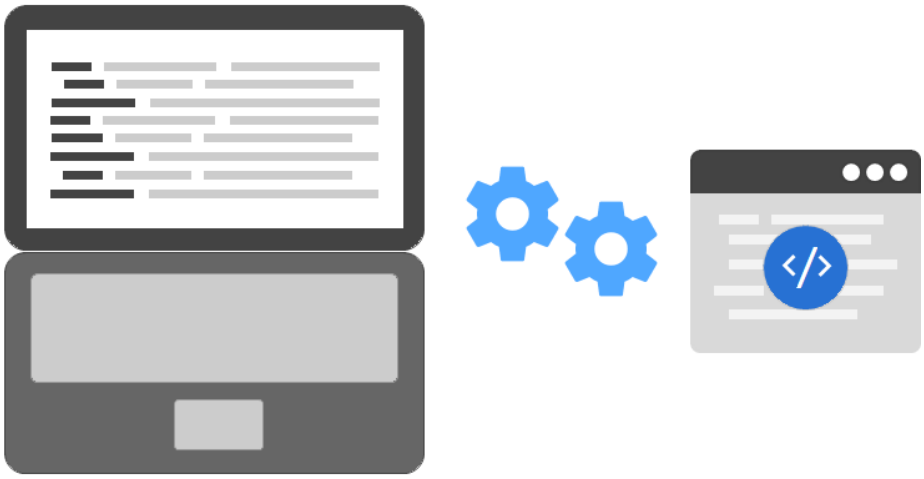
- Navržené UI softwarového produktu
- Podrobný design specifikace jako barvy, téma, styly, pokyny, ikony

2.2.5 Implementace

V této fázi je kladen důraz na kvalitně provedenou implementaci všech vytvořených návrhů pomocí backendových a frontendových technologií. Zde se zapojuje i designer, aby napomohl odchytat vizuální nedostatky, které tvořili vývojáři. Je to moment, kdy je schopný proces zastavit a zadat vývojářům požadavek na finální vizuální úpravy. (Minhas, 2018)

Výstupy:

- Implementovat plně vyvinuté uživatelské rozhraní s předem nadefinovanými funkcionalitami a podle zvoleného stylu



Obrázek 18 – Implementace funkcionalit

Zdroj: (Minhas, 2018)

2.2.6 Zhodnocení

Po implementaci všech funkcionalit přecházíme do fáze zhodnocení. Produkt je vyprodukovaný a dalším úkolem je zhodnotit jeho kvality a slabé stránky. Jsou kladeny otázky jako například, zda je produkt použitelný? Umožňuje všem koncovým uživatelům intuitivní zacházení a jednoduchou manipulaci? Je zcela flexibilní k vývoji a rozšíření možností? Je snadné vyhotovené prvky změnit? Poskytuje všechny požadované přístupy k řešení dané problematiky? Je produkt dostatečně důvěryhodný? (Minhas, 2018)

Po zodpovězení těchto a jiných otázek tým hodnotí, zda vypuštění produktu proběhlo úspěšně a přemýšlí jak dále uživatelské prostředí zlepšovat. Pokud jsou identifikovány nové přístupy k vylepšení, tak se vracíme zpět do fáze návrhu, kde probíhají další změny a opakujeme tento iterativní proces do úplné spokojenosti s vytvořeným produktem. Dále se také srovnávají předchozí a aktuální implementace a definují se nová pravidla pro zlepšení.

Výstupy:

- Získání zpětné vazby od uživatelů
- Auditní zpráva o uživatelském rozhraní
- Zaznamenat zlepšení a oblasti pro jeho další vývoj

Obrázek 19 – Uživatelské testování

Zdroj: (Minhas, 2018)

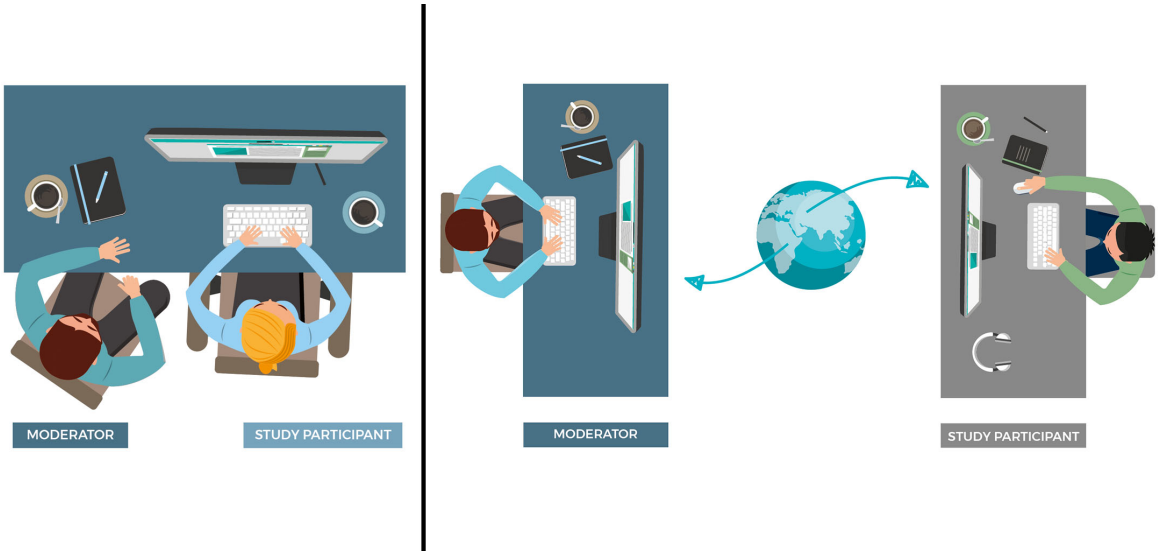
2.3 Metody uživatelského testování použitelnosti

S technologickým vývojem odborníci přinášejí i nové metody testování. Roste potřeba testovat i ty nejkritičtější cesty a odchytat nedostatky aplikací do posledního detailu. Dříve byla prioritní pouhá funkčnost systému. Dnes se stává důležitou součást uživatelská přívětivost, která napomáhá k motivaci uživatelů produkt využívat. Metody pro uživatelské testování použitelnosti napomáhají odhalit překážky v užití, ale i pozitivní aspekty vyvíjeného softwaru. V tomto kontextu je testování UX základním stavebním kamenem životního cyklu softwarového produktu. Do jiných odvětví testování vstupuje na trh automatizace všech testů. Aspekt uživatelské přívětivosti bude vždy vyžadovat podněty od lidských zdrojů. Tuto roli přebírá do svých rukou tester, který má za úkol řídit a hodnotit následující procesy uživatelského testování. Pozoruje logické toky uživatelů a jak k danému softwaru přistupují. Zda využívají intuici a nebo musí dlouze přemýšlet nad jednotlivými úkony. Tyto faktory analyzují následující metody uživatelského testování přívětivosti a umožňují, tak lépe specifikovat potřeby uživatelů. (Nielsen, 2012)

2.3.1 Moderované testování použitelnosti

Moderované testování použitelnosti je vedené formou rozhovoru s účastníky, kteří měli možnost aplikaci testovat. Jde konkrétně o hloubkové rozhovory k tématu aplikace. Existují zde vždy dva nebo více účastníků rozhovoru, který řídí moderátor. Většinou se jedná o roli testera, který software detailně zná a může se tedy ptát na široké spektrum otázek. Jeho úkolem je začít rozhovor zcela jednoduchými otázkami, na které dokáží respondenti odpovědět intuitivně. V čase jsou pak otázky pokládány více do hloubky business logiky tématu. Z odpovědí je následně možné vyvodit myšlenky, frustrace, očekávání a touhy účastníků, které v nich aplikace vyvolala. Rozhovory jsou prováděny formou brainstormingu, který má ovšem danou strukturu a detailně se zaznamenává. Tento přístup testování je vhodné využívat pro komplexnější softwarová řešení. Jako další metodu moderovaného testování použitelnosti můžeme uvést vzdálené testování. Dnešní technologie nám umožňují řešit tuto problematiku z jiných koutů světa a analyzovat výsledky pomocí moderních technologií. Vzdálené testování je v mnoha případech rychlejší, efektivnější a levnější. Rizikem zde může být ne zcela detailní vhled do hlavy participanta, který použitelnost aplikace testuje. (Řezáč, 2016)

„Existují dva typy otázek. Na jedny chcete získat odpověď a ty druhé si můžete dovolit položit respondentům.“ (Řezáč, 2016)



Obrázek 20 – Moderované osobní a vzdálené uživatelské testování

Zdroj: (PacktPub, 2020)

2.3.2 Třídění karet

Jednou z podstatných metod uživatelského testování použitelnosti je třídění karet, neboli z anglického spojení card sorting. Primárním cílem metody třídění karet je pochopit, jak lidé o problematice přemýšlejí. K cart sortingu je možné přistupovat digitální, či analogovou formou. Odborníci preferují analogovou variantu za pomocí lepících papírků. Nejčastěji se touto metodou testuje použitelnost navigace na webu. Po vyhodnocení přináší zlepšení infrastruktury webu a optimalizaci informační architektury. Třídění karet nám umožňuje jak přístup otevřený, tak i uzavřený. Otevřený přístup cart sortingu znamená, že uživatelé dostanou kartičky a mají z nich za úkol vytvořit skupiny. Uzavřený přístup má již skupiny vytvořené a uživatelé jednotlivé kartičky pouze přiřazují do skupin. (Experience UX, 2020)

Proces testování pomocí metody třídění karet:

1. Zvolení oblasti nebo kategorie pro výzkum.
2. Sepsání kartiček, které je za potřebí uspořádat.
3. Žádost uživatelů o selektování a setřídění karet do předem definovaných kategorií v případě uzavřeného cart sortingu. V případě otevřeného cart sortingu si uživatelé vytvoří skupiny sami.
4. Pod další krok otevřeného cart sortingu spadá pojmenování kategorií, vytvoření priorit jednotlivých kategorií a definice hierarchických vztahů.
5. Je vhodné si během výzkumu dělat poznámky, fotit průběh a finální verzi a nikdy nenechat uživatele přemýšlet nahlas, aby neovlivňoval jiné respondenty.
6. Proces iterativně opakujte s větším množstvím uživatelů, pro zvýšení relevance dat.
7. Zanalyzujte výsledky a vyzdvihněte kategorie, které uživatelé používali nejvíce.

Obrázek 21 – Card sorting (Třídění karet)

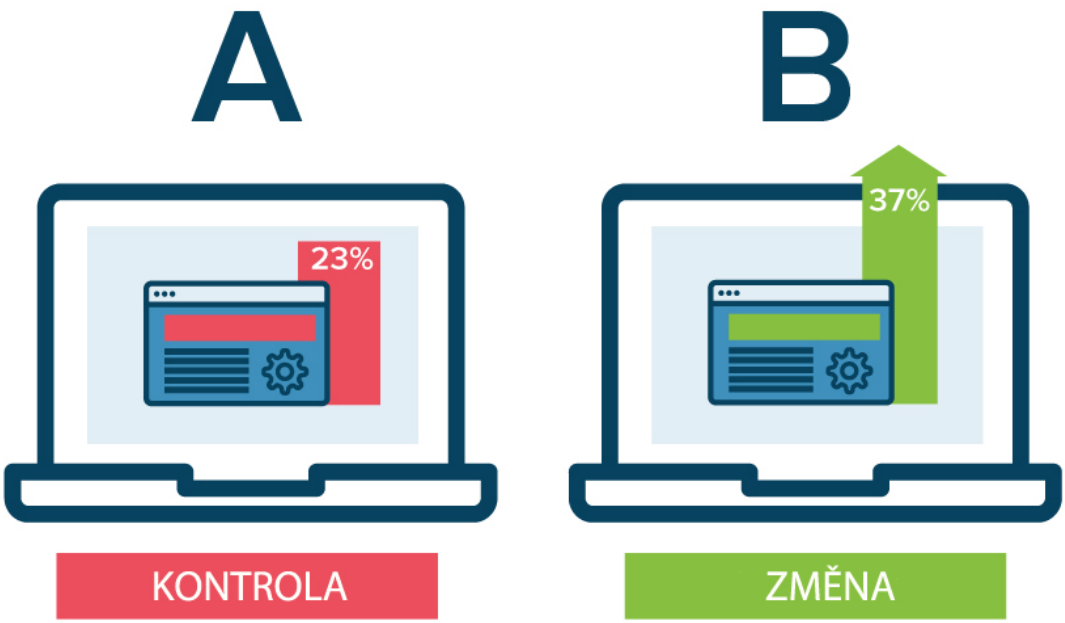
Zdroj: (Experience UX, 2020)

2.3.3 A/B testování

Metodu A/B testování je vhodné využít v momentě, když jsou předloženy dvě odlišné verze návrhu aplikace, tak aby bylo zhodnoceno, které z nich je více uživatelsky přívětivá. Provádí se zejména za účelem optimalizace malých změn stávající verze. A/B testování je proces, který lze již dnes automatizovat. Využívají se nástroje jako například Outbrain, Crazy Egg, anebo Hubspot. (Koníček, 2016)

Manuální A/B testování probíhá v následujících krocích:

1. Vytvoří se jedna varianta webu v červené barvě.
2. Vznikne předpoklad, že web v zelené barvě bude působit na uživatele příjemněji.
3. Vytvoří se tedy verze v zelené barvě.
4. Obě varianty návrhu jsou vypuštěny mezi respondenty, kteří zhodnotí preference.
5. Barevná varianta, která zvítězí může být využita anebo se zařadí do dalšího testování.
6. Hypotézy jsou zaznamenány. Zpětně můžeme dohledat uživatelské preference.



Obrázek 22 – A/B testování

Zdroj: (Koníček, 2016)

2.3.4 Eye-tracking (Testování oční kamerou)

Eye-tracking je moderní metodou, která umožňuje sledovat pohyb očí uživatelů, při prohlížení softwarového uživatelského rozhraní. Nejčastěji se používá k testování webů, prototypů a softwarových produktů. Tato technologie nám pomáhá určit, na jaká místa se

testovaný uživatel na stránce koncentruje, jaké barvy, loga, ikony a popřípadě komponenty přitahují jeho pozornost. Díky tomu je pak možné analyzovat jak tvořit celkový layout stránky a jak poskládat jednotlivé komponenty, tak aby uživatel viděl vše co je podstatné. Jako například tlačítko „koupit“. Tato metoda je silný analytický nástroj, který ovšem vyžaduje čas na zpracování dat a vybavenou laboratoř pro výzkum. Častým technologickým nástrojem pro testování oční kamerou bývá VR, neboli virtuální realita, která umožňuje přesné snímání pohybů lidského oka. (Dawson, 2020)

Obrázek 23 – Technologie eye-tracking

Zdroj: (Dawson, 2020)

Obrázek 24 – Ukázka tepelných map

Zdroj: (Moran, 2019)

3 Analýza a sumarizace aplikace LaunchTRACK 1.0

V kapitole analýza a sumarizace aplikace LaunchTRACK 1.0 uvedu aplikaci jako takovou. Vysvětlím její business logiku a procesy s ní spjaté. Dále popíši některé z komponent, které aplikace využívá. Zanalyzuji silné/slabé stránky, příležitosti a hrozby dle známé SWOT analýzy. Jako další krok využiji heuristické analýzy k testování použitelnosti.V závěru kapitoly budu prezentovat smysl využití aplikace pro oddělení Launch management a možnost využití pro celou strukturu ŠKODA AUTO a.s. Odpovíme si také na otázky jak velkým přínosem byla aplikace doposud a proč její osud leží v rukou uživatelů.

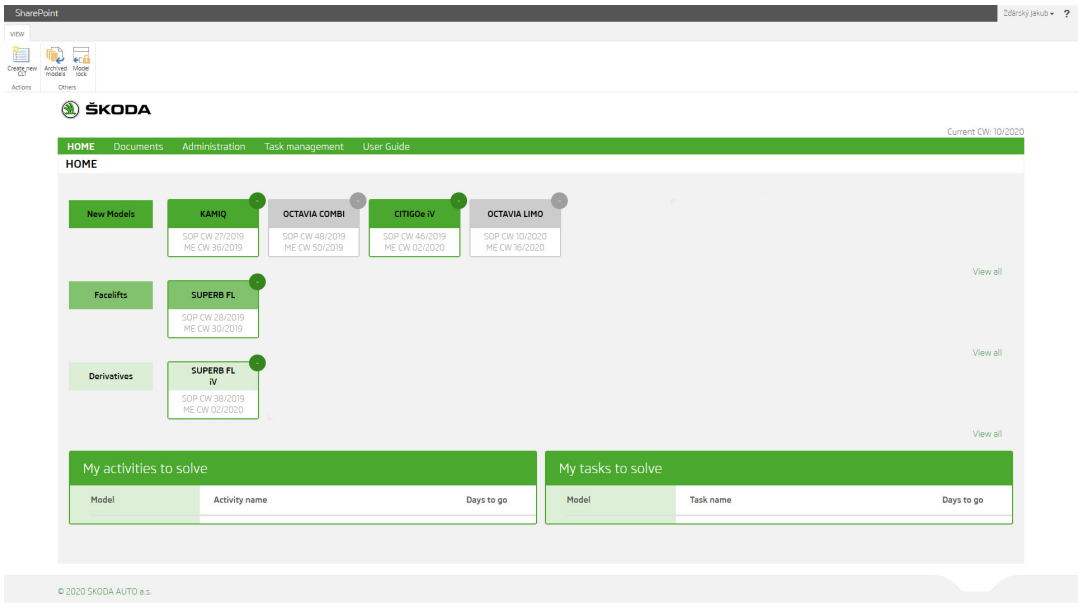
3.1 Představení aplikace LaunchTRACK 1.0

Aplikace LaunchTRACK vstupuje do vývoje v roce 2012. Zadavatelem aplikace se stává ŠKODA AUTO a.s. společně s bývalým vedením kanceláře Launch Management, které působí na oddělení VMP, neboli oddělení produktového marketingu. Aplikace si prošla všemi standardními kroky developmentu, které zmiňuji v teoretické části mé bakalářské práce. Okolnosti vývoje dovedli aplikaci do současného stavu, ve kterém slouží jenom několika málo trhům v různých koutech světa. Momentálně je nasazena na produkčním prostředí ŠA, kde nachází své největší využití pro trhy jako je například Indie. Z pohledu hodnotitele stávající podoby jsem přesvědčený o možnosti nárůstu kvality zpracování, optimalizaci, možnosti vývoje nových funkcionalit a přidání analytických nástrojů.

3.1.1 Dashboard

Dashboard, neboli domovská stránka aplikace nám zobrazuje UI, na kterém vzniká první interakce s uživatelem přistupujícím do aplikace. Slouží jako rozcestník pro vstoupení do konkrétního připravovaného modelu. Můžeme zde nalézt sekce:

- Akční ribbon v horním levém roku.
- Navigaci (Home / Documents / Administration / Task manager / User Guide).
- Centrální struktura s modely (New models / Facelifts / Derivatives).
- My activities to solve.
- My tasks to solve.

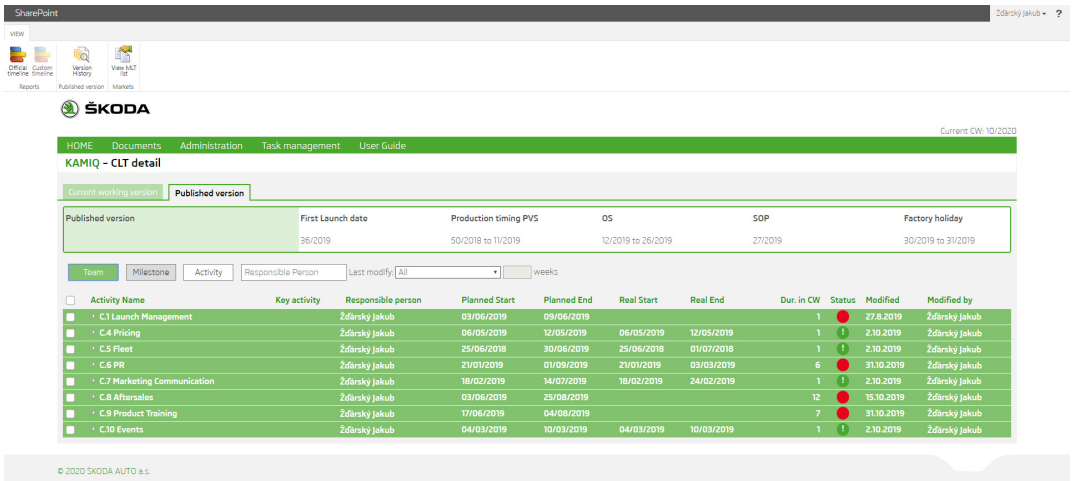


Obrázek 25 – Dashboard aplikace LT 1.0

Zdroj: (Materiály ŠA, LaunchTRACK 1.0, 2020)

3.1.2 CLT (Central Launch Plan)

Cental Launch Plan, neboli centrální plán vypouštění modelů na trh reprezentuje strukturu, která se dělí na Published version (Publikovanou verzi) a Current working version (Aktuální pracovní verzi). Current working version umožňuje administrátorovi aplikace předpřipravit průběh launche, který následně zveřejní do publikované verze, kdy vidí strukturu všichni HQ uživatelé. Po zpracování týmů, milníků a aktivit se vypouští model na trhy. Úkolem HQ a IMP editorů je vyplnit data k předem definovaným milníků, tak, aby měly všechny strany aktuální informace.

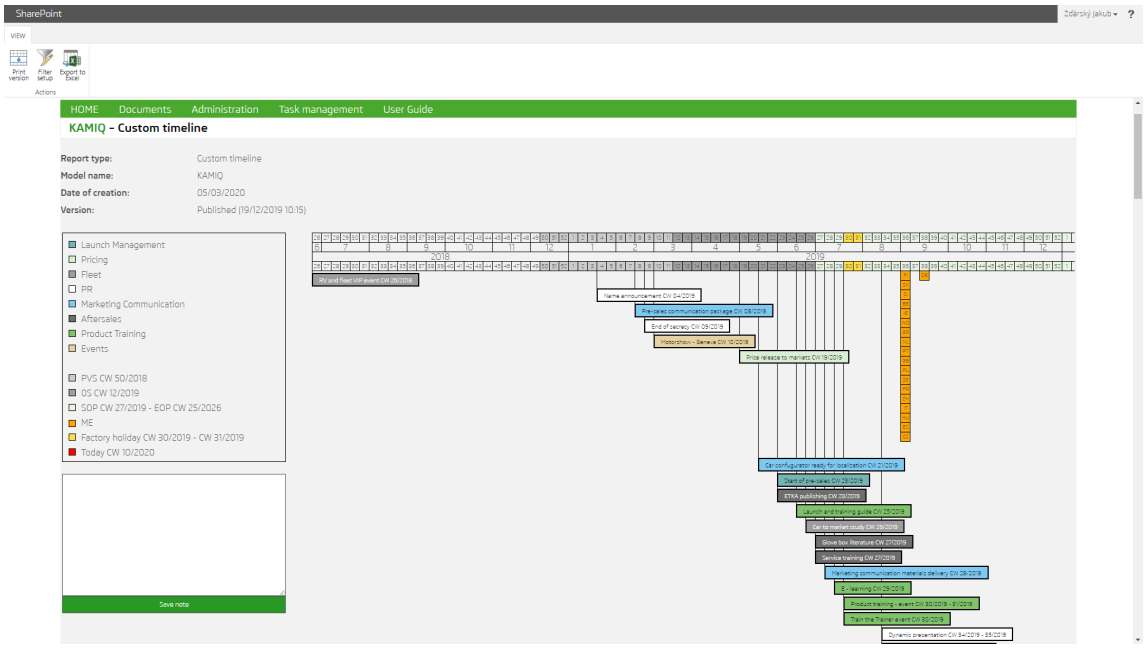


Obrázek 26 – CLT aplikace LT 1.0

Zdroj: (Materiály ŠA, LaunchTRACK 1.0, 2020)

3.1.3 Custom Timeline

Aplikace umožňuje vygenerovat časovou osu definovanou z milníků, které si mohl uživatel vytvořit anebo požadoval vygenerovat oficiální časovou osu připravenou a schválenou týmem administrátorů z oddělení Launch Managementu.



Obrázek 27 – Custom timeline LT 1.0

Zdroj: (Materiály ŠA, LaunchTRACK 1.0, 2020)

3.2 Business logika aplikace

Business logika určuje záměr vývoje aplikace, který napomáhá definovat vizi celého projektu a budovat logické celky, ze kterých se aplikace skládá. Jedná se o jakýsi záměr vývoje spojený s primárními body, které musí software splňovat. Aplikace LaunchTRACK je určena pro sběr a sdílení informací ohledně launchů (=zavedení nových modelů na trh). Zaváděné modely automobilů mohou vznikat pod kategoriemi New Models (=Nové modely), Facelifts (=Facelifty), anebo Derivatives (=Deriváty). Sběr a sdílení dat probíhá jak na úrovni centrálního marketingu (HQ ŠKODA AUTO a.s.), tak i ze strany importérů, kteří potřebují informace pro přizpůsobení marketingové kampaně a prodejních plánů. Dalším z neopomenutelných bodů je komunikace mezi HQ a trhy. Doposud probíhala veškerá komunikace v rámci emailové korespondence a telefonátů. Cílem je centralizovat data na jedno místo, aby k nim měli přístup všechny zúčastněné strany a šetřil se tedy čas i peníze. S vývojem aplikace se zvažují i analytické nástroje, které umožní zpětné vyhodnocení nebo návrh nových plánů vycházejících z úspěšně provedeného vypuštění

modelu na trh. Pro úplné pochopení business logiky je zásadní vnímat pojem launching proces, který doplňuje fungování aplikace.

3.2.1 Launching proces

Launching proces, neboli vypuštění nového modelu automobilu na trh umožňuje definovat a analyzovat jednotlivé fáze od založení modelu po shrnutí feedbacků, které se na vypuštěný model váží. Je to cyklus, kterým projde každý komerčně známý model automobilu ŠKODA AUTO a.s. Konkrétní fáze launching procesu jsou následující:

1. Vydefinování modelu a příprava dat administrátorem
2. Sdílení informací s jednotlivými HQ týmy ŠKODA AUTO a.s.
3. Doplnění dat HQ týmů a společná evaluace s administrátory (kanceláří LM)
4. Vypuštění informací o modelu na importérské trhy
5. Vydefinování ME (=Market Entry – Den vypuštění modelu na konkrétním trhu)
6. Sběr feedbacků z trhů po uplynutí časového intervalu od ME
7. Evaluace feedbacků
8. Uzavření procesu vypouštění modelu na trh

3.3 SWOT analýza aplikace LaunchTRACK 1.0

SWOT analýza je technika, která pomáhá analyzovat vnitřní a vnější faktory ovlivňující úspěšnost aplikace, produktu nebo služby. Strukturu SWOT analýzy dělíme na definici vnitřních faktorů, do kterých náleží silné a slabé stránky záměru. Odpovídáme si na otázky, kde má produkt své kvality a kde je naopak nedostatečný. Vnější faktory ovlivňující produkt jsou příležitosti a hrozby, které přicházejí z okolního prostředí. Akronym SWOT vychází ze spojení následujících slov z anglického jazyka:

- **S – Strengths – Silné stránky**
- **W – Weaknesses – Slabé stránky**
- **O – Opportunities – Příležitosti**
- **T – Threats – Hrozby**

Úkolem a primárním cílem SWOT analýzy je zajistit silné stránky, které je vhodné následně podporovat a stavět na nich. Zanalyzovat slabé stránky a potlačit je, případně zcela odstranit. Nacházet nové příležitosti a pracovat s nimi tak, aby byly užitečné. V poslední fázi je důležité znát hrozby, které mohou záměr, produkt, či službu ohrozit. Ať už jen lehce, tak i existenčně. Je důležité těmto hrozbám předcházet a snažit se je eliminovat. Aby byla SWOT analýza relevantní a přinesla smysluplné výstupy, tak je podstatné postupovat dle

definovaných pravidel. Je důležité udělit prioritu všem ovlivňujícím faktorům. Objektivně přistupovat k celé problematice a nebát se řešit otázky v rámci týmu. Úsloví, které známe jako „víc hlav, víc ví“ je zde na místě. Je vhodné se vyvarovat pouhým domněnkám. Poslední krok SWOT analýzy je vyhodnocení a připravení opatření, pro další naplnění strategie a realizačního plánu. (Managementmania, 2017)

Tabulka 6 – SWOT analýza aplikace LT 1.0

	VNITŘNÍ	VNĚJŠÍ
POZITIVNÍ	**Silné stránky**	**Příležitosti**
	- Urychlení procesu vypuštění modelu automobilu na trh - Zpracování informací o procesu uvedení na trh - Snaha o komunikaci mezi HQ ŠA a importérskou stranou - Možnost archivace modelů	- Potenciál vytvořit fungující aplikaci, která bude zcela uživatelsky přívětivá - Zrychlení procesů koordinace vypouštění modelů na trh - Možnost přinést komunikační nástroj, který ušetří čas a finance - Jednotný systém užívaný napříč koncernovou politikou ŠA - Uchování znalostí
NEGATIVNÍ	**Slabé stránky**	**Hrozby**
	- Špatná distribuce mezi uživatele - Nekonzistentní UI aktivity - Průchod aplikací je zmatený - Systém není vhodně optimalizovaný (nestabilita) - Pomalá odezva systému - Není patrné co uživatel změnil - Absence možnosti in-place editace při vkládání dat	- Změna chování uživatelů - Nedostatečné využití aplikace - Možnost ztráty dat - Neúplná informovanost o užití aplikace (nedostatečná školení) - Nařízení pro využití jiného softwaru vedením ŠA

Zdroj: vlastní zpracování

Z vytvořené tabulky SWOT analýzy vyplývá, že aplikace LaunchTRACK 1.0 zaznamenává celkem 9 pozitivních faktorů, z toho jsou čtyři silné stránky a pět příležitostí. Z těchto hodnot je možno odvodit, že aplikace vzniká se smysluplným záměrem a má velký potenciál k budoucímu rozvoji. Pro Launch management tým se může stát silným optimalizačním nástrojem s vizí zrychlení procesů. Umožní zautomatizovat úkony, které stále musí provádět lidský zdroj.

Pokud si rozebereme silné stránky, tak můžeme konstatovat, že aplikace LaunchTRACK umožňuje ukládání a archivaci dat souvisejících s procesem uvedení jednotlivých modelů a modelových řad automobilů na trh. Uživatel získá přehled o procesech, které proběhly v minulosti a má jedinečnou možnost analyzovat jejich kvalitu a případně z nich čerpat při tvorbě dalších launchů. Rád bych konstatoval, že aplikace je zajímavá spíše svým potenciálem a příležitostmi, které nabízí. Business logika stojící za aplikací má maximální smysl, který může přinést optimalizaci a zrychlení procesů. Zefektivní koordinaci, která bude téměř automatizovaná. Od příležitostí si tedy můžeme slibovat novou verzi aplikace, která přinese kvalitní uživatelské rozhraní a bude splňovat podmínky uživatelského prožitku.

Naopak negativních faktorů se nám podařilo zmapovat celkem 13. Slabé stránky kladou důraz na ne zcela vhodně zpracované UI. V aplikaci se nachází nekonzistence a celkový průchod není jednoznačný. Mezi hrozby můžeme zařadit faktory jako například ztráta dat. Neúplné školení a informovanost uživatelů. SWOT analýzu stávající verze aplikace lze vyhodnotit tak, že momentální kvalita zpracování neodpovídá požadavkům uživatelsky přívětivého prostředí, které bychom jako uživatel daného softwaru očekávali. Proto můžeme apelovat na zlepšení uživatelského prostředí. V nejlepším případě jeho redesignu a další možnosti refactoringu struktury. Tyto inovace by mohli přinést zrychlení systému a zlepšení celkové přívětivosti. Tento posun by motivoval uživatele aplikaci využívat a napomohl by tak k dalšímu rozvoji nových funkcionalit. Popřípadě zajistit využívání napříč koncernovou politikou ŠKODA AUTO a.s.

3.4 Heuristická analýza aplikace LaunchTRACK 1.0

Heuristická analýza je jednou z mnoha metod hledání chyb v rámci použitelnosti softwarových řešení. Testování může probíhat jedním nebo více testery, kteří mají za úkol zhodnotit prostředí za pomocí heuristických principů použitelnosti. Tyto principy byly nadefinovány Jackobem Nielsenem a Rolfem Molichem. Nejprve vzniklo v roce 1990 principů devět, které byly následně doplněny v roce 1994 o princip desátý. (Nielsen, 1994)

Deset principů použitelnosti dle J. Nielsena a R. Molicha:

1. Viditelnost stavu systému – Systém by měl dát uživateli jasnou zpětnou vazbu o tom, co se právě v ten daný okamžik odehrává.

2. Propojení mezi systémem a reálným světem – Systém by měl být navržený tak, aby používal jazyk, kterému uživatel bezpečně rozumí. Komunikace tedy musí probíhat bez odborných termínů.

3. Uživatelská kontrola a svoboda – Uživatel musí mít možnost opustit místo takzvaným „nouzovým východem" a nebo operaci vrátit zpět. Lidé dělají chyby a tato možnost je oprostí od stresu a umožní chyby napravit.

4. Konzistence a standardizace – Prvky v rámci systému by měly být zcela konzistentní napříč platformou. Jako příklad si můžeme uvést panely záhlaví aplikací Word, Excel a PowerPoint od společnosti Microsoft.

5. Prevence chyb – Systém by měl být navržený tak, aby omezoval chyby na minimum. Tomu zabráníme kvalitním a pečlivým návrhem systému, který bude mít ošetřený všechny logické celky, vstupy a výstupy. Tím zamezíme tomu, aby uživatel systém rozbil popřípadě, aby nevznikalo neočekávané chování systému. Zásadní je chyby ze systému eliminovat.

6. Rozpoznání místo vzpomínání – Systém by měl být nastavený pro zcela intuitivní zacházení. Uživatel tedy nebude musel hledat informaci, tam kde by jí neočekával.

7. Flexibilita a efektivita použití – Systém slouží uživateli pro zrychlení procesů. Pro lepší práci se systémem je vhodné uživatelům nabízet personalizaci, popřípadě klávesové zkratky, které umožní rychlejší práci.

8. Estetický a minimalistický design – Systém by neměl být zahlcený nepodstatnými informacemi a prvky, které nejsou zcela nezbytné.

9. Pomoc uživatelům rozpoznat, diagnostikovat a vzpamatovat se z chyb – Chybové zprávy by měly informovat uživatele v uživatelsky přívětivé formě a navrhovat přístupy k řešení.

10. Nápověda a dokumentace – I přesto, že by zcela intuitivní systém neměl potřebovat dokumentaci, tak by měla být v případě potřeby uživateli snadno dostupná.

(Nielsen, 1994)

Desatero použitelnosti je velmi univerzálním nástrojem a lze aplikovat na širokou škálu softwarových produktů. S ohledem na ne zcela jasně specifikované body bohužel tuto analýzu musí provádět vyškolený specialista. Z tohoto důvodu začaly vznikat další konkrétnější body heuristiky, které navádějí testera na přesněji vydefinované části softwarového rozhraní. Cílem heuristické analýzy je vydefinovat nedostatky již vzniklého softwarového řešení.

Hodnocení heuristických tabulek probíhá na 3 úrovních:

- **ANO** – prostředí aplikace je na úrovni shody se zvolenou heuristikou.
- **N/A** – prostředí aplikace je na úrovni shody s jistými nedostatky.
- **NE** – prostředí aplikace se neshoduje se zvolenou heuristikou.

Každá z odpovědí nabývá na jiné hodnotě a to: **(ANO = 1, N/A = 0, NE = -1)** Díky těmto hodnotám budeme schopni vyhodnotit procentuální úspěšnost jednotlivých heuristik a vytvořit tak graf zachycující výsledky naší analýzy ve vizuální podobě.

3.4.1 Heuristiky přístupnosti

Tabulka 7 – Odpovědi respondentů na otázky Heuristiky – Přístupnost

1	Heuristiky – Přístupnost	Hodnotitel č.1	Hodnotitel č.2	Hodnotitel č.3
1.1	Má webová aplikace snadno zapamatovatelnou URL adresu?	ANO	ANO	N/A
1.2	Obsahují objekty na stránce alt atribut?	N/A	N/A	NE
1.3	Jsou objekty na domovské stránce vykresleny, tak, aby šly všechny důležité akce provést na jedno kliknutí?	N/A	ANO	ANO
1.4	Je možné aplikaci přenastavit do několika světových jazyků?	NE	NE	NE
1.5	Lze aplikaci spustit a využít na standardně dostupných prohlížečích (Google Chrome, Mozilla Firefox, Safari, IE)	ANO	N/A	ANO

Zdroj: vlastní zpracování

Hodnocení přístupnosti dle heuristické analýzy přineslo výstupy od 3 uživatelů a to například absence jazykových mutací. Jedná se sice o interní systém, který budou využívat zaměstnanci ŠA, ale i tak se domnívám, že by v budoucích iteracích vývoje ocenili jisté jazykové verze. Dashboard, neboli hlavní stránka je navřena tak, aby z hlediska přístupnosti

umožňovala rychlý náhled do modelů a uživatel, tak může téměř okamžitě vkročit do míst, jež vyhledává. Výhodou webové aplikace je, že má snadno zapamatovatelnou doménu, a proto bude přístup na stránku pro uživatele příjemnější. Problém přístupnosti by mohlo narušit ne zcela vhodně optimalizované prostředí, co se rychlosti načítání týče.

3.4.2 Heuristiky layoutu a vizuálního designu

Tabulka 8 – Odpovědi respondentů na otázky Heuristiky – Layout a Vizuální design

2	Heuristiky – Layout a Visuální design	Hodnotitel č.1	Hodnotitel č.2	Hodnotitel č.3
2.1	Jsou od sebe jednotlivé sekce rozložení stránky vizuálně odděleny?	N/A	NE	NE
2.2	Je stránka vyvážená v poměru obsahu informací a textu?	N/A	NE	ANO
2.3	Odkazuje logo stránky na úvodní dashboard (hlavní stránku)?	ANO	ANO	ANO
2.4	Je aplikace zcela responzivní a přizpůsobena pro mobilní zařízení?	NE	NE	NE
2.5	Nesplývá pozadí aplikace s textem? Je dostatečně kontrastní?	ANO	ANO	N/A
2.6	Neobjevují se v aplikaci více jak 2 fonty? Je využívaný font dobře viditelný ve všech sekcích?	ANO	N/A	ANO
2.7	Vykreslují se prvky na stránce jako například komponenta Custom timeline čistě?	NE	N/A	NE
2.8	Lze aplikaci spustit a využít na standardně dostupných prohlížečích (Google Chrome, Mozilla Firefox, Safari, IE)	N/A	N/A	NE

Zdroj: vlastní zpracování

Názor na vizuální design je velmi často individuální, ale existují různé premisy, které by měly fungovat napříč celým rozvržením stránky. Bylo zaznamenáno, že uživatelé se často ztrácí v rozdělení jednotlivých sekcí. Podle výsledků nejsou sekce s komponenty správně odděleny, a proto se může stát, že celkový dojem stránky působí jednotvárně.

Vykreslení některých komponent není po vizuální stránce korektní. Konkrétně to může být custom timeline, ve které zaznamenáváme přetečení textu anebo naopak moc velký prázdný prostor. Jednu z věcí, které je zapotřebí vyzdvihnout je přesměrování klikem na logo na domovskou stránku. Je to funkcionalita, na kterou jsou uživatelé poměrně zvyklý a rádi jí využívají.

3.4.3 Heuristiky informační architektury a navigace

Tabulka 9 – Odpovědi respondentů na otázky Heuristiky – IA a Navigace

3	Heuristiky – IA a Navigace	Hodnotitel č.1	Hodnotitel č.2	Hodnotitel č.3
3.1	Je zřejmé kde se navigace nachází?	ANO	ANO	ANO
3.2	Název jednotlivých polí navigace definuje kam bude uživatel přesměrován?	ANO	N/A	ANO
3.3	Nachází se v aplikaci zpětná vazba o struktuře? Kde se nacházíme pomocí drobečkové navigace?	NE	NE	N/A
3.4	Zásadní informace se zobrazují co nejblíže pod navigací?	ANO	ANO	N/A
3.5	Je zakomponován hover efekt (změna šipky, pozadí tlačítka) při najetí myší na položku navigace?	ANO	ANO	ANO
3.6	Je stránka konzistentní z pohledu zachování logo a layoutu?	N/A	NE	NE
3.7	Je stránka konzistentní s ohledem na informační strukturu?	NE	N/A	NE
3.8	Je k nahlédnutí mapa stránek?	NE	NE	NE
3.9	Odpovídá název menu a další prvky aplikace názvu URL adresy?	N/A	NE	N/A

Zdroj: vlastní zpracování

3.4.4 Heuristiky chyb a zpětné vazby

Tabulka 10 – Odpovědi respondentů na otázky Heuristiky – Chyby a Zpětná vazba

4	Heuristiky – Chyby a Zpětná vazba	Hodnotitel č.1	Hodnotitel č.2	Hodnotitel č.3
4.1	Je aplikace připravena na chybovou hlášku 404? Umožní uživateli pochopit, že zvolený požadavek nebyl nalezen?	N/A	N/A	N/A
4.2	Pokud nastane chyba, tak je uživatel obeznámen jak pokračovat dále?	NE	NE	NE
4.3	Zobrazují se chybové hlášky uživatelsky přívětivým způsobem (např. pomocí Pop-Up okna)?	NE	NE	N/A
4.4	Je načítání jednotlivých stránek rychlé? (Nejdéle s prodlevou 3 vteřin?)	N/A	ANO	N/A
4.5	Je snadné se vrátit zpět nebo ukončit v případě potřeby prováděnou akci?	ANO	ANO	ANO
4.6	Je na stránce k dispozici uživatelský manuál? Popřípadě kontakt na podporu?	ANO	ANO	ANO

3.5 Shrnutí heuristické analýzy

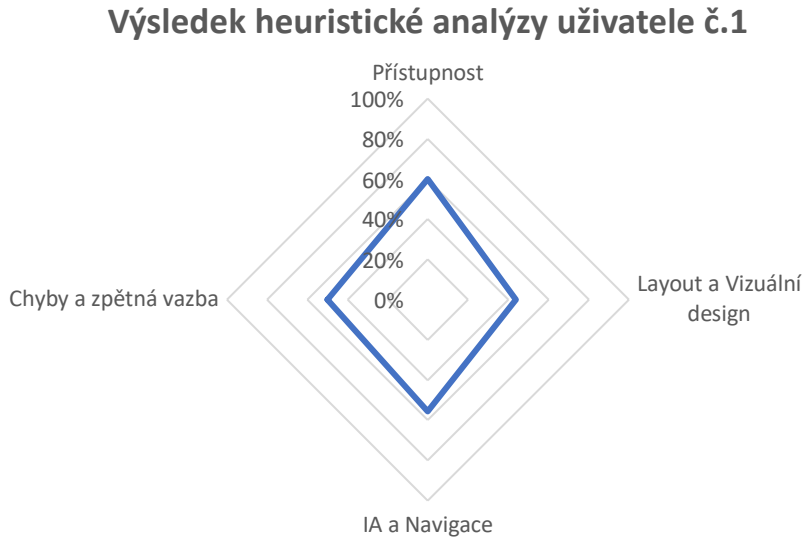
Výstupem heuristické analýzy aplikace LaunchTRACK 1.0 se stává hodnocení tří uživatelů, kteří nezávisle na sobě provedli test na čtyři heuristické kategorie. Byla vyhodnocena přístupnost, layout a vizuální design, informační architektura a navigace, chyby a zpětná vazba. Toto zhodnocení vytvořilo strukturu správně a špatně zpracovaných celků, které v rámci dalšího vývoje podstoupí re-design a pomohou, tak vytvořit, lépe uživatelsky a vizuálně přívětivou aplikaci. Zajímavým poznatkem je, že tři na sobě nezávislí uživatele přistupovali k hodnocení podobně a na většinu konkrétních otázek odpovídali po rychlém přístupu k aplikaci zcela intuitivně.

3.5.1 Hodnotitel č.1

Tabulka 11 – Vyhodnocení heuristik uživatele č.1

Vyhodnocení heuristik uživatele č.1	Součet bodů	Procentuální vyjádření
Přístupnost	**1**	**60 %**
Layout a Vizuální design	**-1**	**43,75 %**
IA a Navigace	**1**	**55,6 %**
Chyby a zpětná vazba	**0**	**50 %**

Zdroj: vlastní zpracování



Obrázek 28 – Graf s výsledky heuristické analýzy uživatele č.1 aplikace LT 1.0

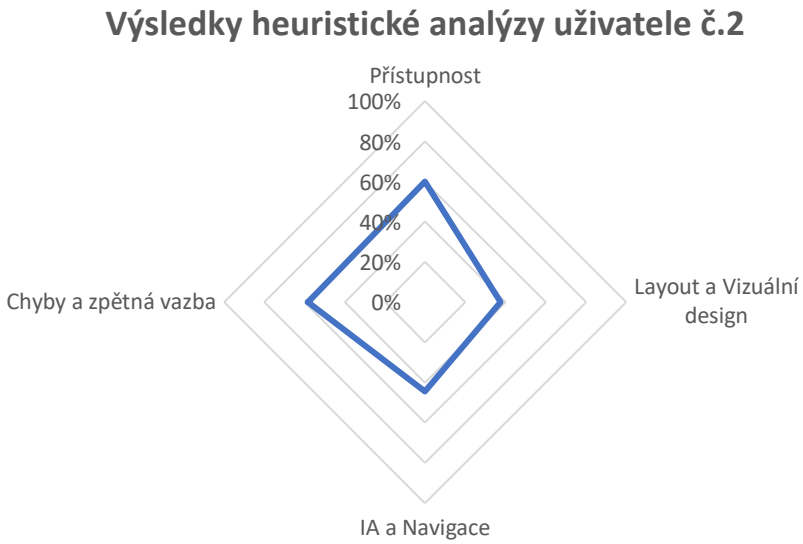
Zdroj: vlastní zpracování

3.5.2 Hodnotitel č.2

Tabulka 12 – Vyhodnocení heuristik uživatele č.2

Vyhodnocení heuristik uživatele č.2	Součet bodů	Procentuální vyjádření
Přístupnost	**1**	**60 %**
Layout a Vizuální design	**-2**	**37,5 %**
IA a Navigace	**-1**	**44,4 %**
Chyby a zpětná vazba	**1**	**58,3 %**

Zdroj: vlastní zpracování



Obrázek 29 – Graf s výsledky heuristické analýzy uživatele č.2 aplikace LT 1.0

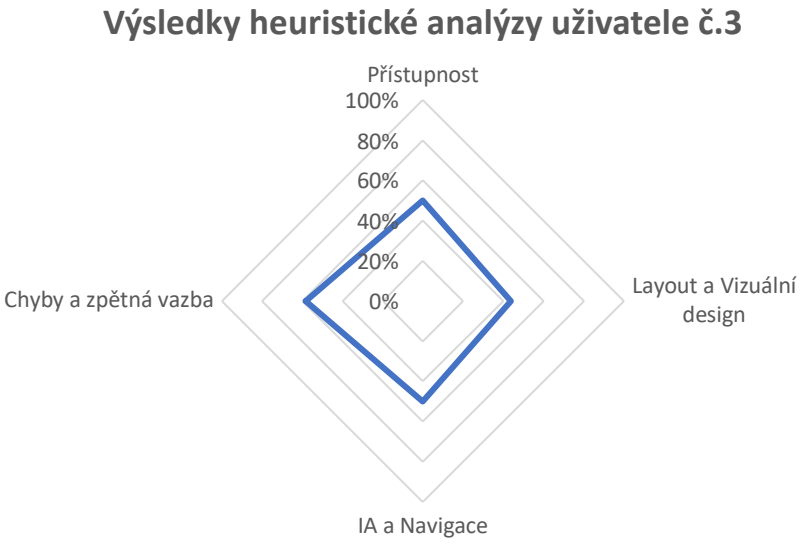
Zdroj: vlastní zpracování

3.5.3 Hodnotitel č.3

Tabulka 13 – Vyhodnocení heuristik uživatele č.3

Vyhodnocení heuristik uživatele č.3	Součet bodů	Procentuální vyjádření
Přístupnost	**0**	**50 %**
Layout a Vizuální design	**-1**	**43,75 %**
IA a Navigace	**0**	**50 %**
Chyby a zpětná vazba	**1**	**58,3 %**

Zdroj: vlastní zpracování



Obrázek 30 – Graf s výsledky heuristické analýzy uživatele č.2 aplikace LT 1.0

Zdroj: vlastní zpracování

3.6 Definice cílového stavu aplikace LaunchTRACK 2.0

Cílem pro následující vývoj bude vytvořit čistou informační architekturu, která podpoří konzistenci a logičnost napříč celou aplikací. Očekávaným zlepšením je rychlost a stabilita aplikace, která umožní rychlejší práci s načítáním a čtením dat. Aplikace bude vytvořena jako komunikační nástroj mezi HQ – centrálou ŠA a IMP – importérskou stranou, neboli trhy. Uživatelské prostředí by mělo být připraveno pro re-design na jiný branding v případě rozšíření aplikace pro jiné značky. Aplikace by také měla splňovat podmínky responzivního designu pro přizpůsobení na různé zobrazovací zařízení. Jako například pro 4K monitor, tak i pro mobilní přístroje. Od aplikace se očekává podpora všech standartních prohlížečů, kterými jsou například Google Chrome, Mozzila Firefox, Safari, Opera a dokonce i Internet Explorer, který je interním prohlížečem pro zaměstnance ŠKODA AUTO a.s. Závěrečným shrnutím definice cílového stavu aplikace LT je postupná implementace nových inovačních komponent a uživatelských funkcionalit, které pomohou zvyšovat uživatelský prožitek.

4 Řízení a optimalizace procesů vývoje aplikace

Kapitola řízení a optimalizace procesů vývoje se zaměří na návrh přechodu na efektivnější metodiku vývoje softwaru. V teoretické části mé bakalářské práce byly popsány jednotlivé fáze vývoje a metodické přístupy k softwarovému developmentu. Také byly zohledněny pozitivní a negativní stránky, se kterými je vhodné pracovat a hledat nejvhodnější cestu pro konkrétní projekt. Osobně jsem byl součástí vývoje aplikace LaunchTRACK v rámci dlouhodobé stáže ve ŠKODA AUTO a.s. Mým úkolem bylo společně s kolegy v týmu zadávat a navrhovat komponenty vývojářské firmě. Podílel jsem se tedy na celkovém procesu vývoje aplikace z pohledu business zadavatele.

4.1 Aktuální stav vývoje LT

Vývoj aplikace LaunchTRACK probíhá přístupem waterfall metodiky. Je stanovený jasný plán vývoje, který ovšem neměl definované konkrétní datum uvedení aplikace na produkční prostředí. Datum uvedení na produkci vykrystalizovalo až v rámci vývoje. Současně však vývoj vykazuje známky metodiky SCRUM, jelikož jsou jednotlivé úkoly zadávány v pravidelném intervalu dvou týdnů. Toto období je definováno jako sprint. Po každém uplynulém cyklu probíhá revize vyvinutých komponent a zaplánování dalších vývojových úkolů pro následující období 14 dnů. Definice nových funkcionalit je komunikována s IT analytikem vývojářské firmy, který následně v logické souvislosti rozepisuje zadání pro vývojářský tým. Zadání se revidují a řeší se detailní požadavky pro jednotlivé uživatelské role s ohledem na restrikční politiku aplikace. Po vyvinutí zadaných úkolů a nasazení na testové prostředí je úkolem testera, aby identifikoval všechny nedostatky a případné chyby předal zpátky na vývojářský tým FE a BE specialistů. Jednou z kompetencí našeho týmu je přistupovat k aplikaci z role testera a hodnotit, zda je vše vyvinuto podle nadefinovaných user stories. Pokud vše odpovídá, komponenta se úspěšně uzavře a je proplacena dodavateli. Pokud není zadání vypracováno korektně, tak vzniká požadavek na přepracování a vývojový tým musí chybnou část komponenty opravit, či doplnit.

Momentální vývoj softwaru funguje v hybridním spojení metodiky Waterfall a SCRUM. Výhodou může být zvyk celého týmu na tento přístup. Ovšem pro větší flexibilitu, efektivitu a otevřenou komunikaci v týmu by bylo vhodné přistoupit k metodice, která by rozvolnila ruce jak zadavateli, tak dodavateli při vývoji aplikace.

4.2 Optimalizační návrh vývoje LT

Optimalizační návrh vzniká na základě vědomostí nabytých v teoretické části mé bakalářské práce. Měl jsem možnost systematicky prostudovat jednotlivé metodiky vývoje softwaru a osvětlit si výhody a nevýhody s nimi spjaty. S ohledem na dynamicky se vyvíjející živý organismus aplikace LaunchTRACK jsem přesvědčený o možnosti zefektivnění. Nechci tím napadat momentální přístup a kvalitu odváděné práce týmu, ale spíš hledám cestu k nárůstu spokojenosti na obou stranách. Jak týmu zadavatele, tak i dodavatele. V rámci vývojových cyklů se jako celek dostáváme všichni do značného tlaku s ohledem na návrh komponent. Vznikají sice logické celky, které jsou vyvíjeny, ale ne s dostatečným předstihem. Další faktor ovlivňující tlak v týmu, který jsem zaznamenal, je soustředěná komunikace na IT analytika. Je to zkušený profesionál, ale soubor našich požadavků přechází vždy přes jeho osobu a v takovém množství může vznikat jistý komunikační šum. Bylo by tedy vhodné soustředit komunikaci a požadavky přímo na vývojáře s ověřením správnosti u IT analytika.

Z toho vychází přístup k několika metodám vývoje softwaru. Mou osobní preferencí by, ale bylo zvolit agilní metodiku vývoje softwaru, která umožňuje přímou komunikaci se všemi členy týmu. Je možné dynamicky diskutovat o zadání a měnit ho v čase. Úkolem agilní metodiky je nadefinovat přibližná, či konkrétnější zadání do jakéhosi Backlogu, což je vyhrazený prostor například v optimalizačním nástroji JIRA, kde se nacházejí připravená nadefinovaná zadání. Úkolem business týmu, potažmo Product ownera je jednotlivým taskům přiřadit prioritu a získat tak strukturu vývoje. Výhodou agilní metodiky je, že zadání lze dynamicky měnit bez ohledu na navýšení platby za změnu chování nebo vizuální podobu komponenty. Agilní přístup umožňuje zadavateli vidět vývojovému týmu více „do karet“ a stává se tedy tak pomyslnou součástí vývojového týmu. V tomto smyslu jednou týdně, případně jednou za 14 dní probíhá společná schůzka mířená na hodnocení pracnosti nových vývojových úkolů. Po společném vydefinování pracnosti, má pak business tým možnost určit priority úkolů a následně je zařazovat do sprintů. Efektivita v agilní metodice přichází s nárůstem zkušeností komunikace mezi zadavatelem a vývojářem. Dále se také zvyšuje efektivita vývojového týmu, který zná lépe chování aplikace a v čase tak dokáže předvídat potřeby zadavatele. Úkolem vývojářů je v této metodice, aby byli angažovaní a přinášeli návrhy na zlepšení sami, případně po brainstromingu s vlastníkem produktu. Navrhuji tedy přechod na agilní metodiku vývoje aplikace LaunchTRACK. V následující kapitole vytvořím SWOT analýzu přibližující možné silné/slabé stránky, příležitosti a hrozby.

4.3 SWOT analýza přechodu na agilní metodiku vývoje

Tabulka 14 – SWOT analýza přechodu na agilní metodiku vývoje softwaru

	VNITŘNÍ	VNĚJŠÍ
POZITIVNÍ	**Silné stránky**	**Příležitosti**
	- Přínos větší flexibility pro vývojový tým při zadávání komponent k vývoji - Požadavky je možné dynamicky měnit po domluvě s vývojářem - Rychle viditelný výstup - Proces je zadavateli zcela transparentní - Jednoduchý na pochopení - Týmová práce s vývojáři	- Rychlé zpětné vazby při přímé komunikaci mezi businessem a vývojem - S ohledem na iterativní proces se vývojový tým v čase zrychluje a je schopný dodávat stejně náročné komponenty rychleji - Agilní vývoj stojí na týmu, který je motivovaný přinést projektu maximum (Částečný úkol PM)
NEGATIVNÍ	**Slabé stránky**	**Hrozby**
	- Zadavatel musí být plně angažovaný do vývoje a komunikace s týmem (Může být i silnou stránkou) - Není zcela vhodný pro velké projekty - Není kladen důraz na přílišnou dokumentaci procesu vývoje	- Možný nedostatek dokumentace - Tým musí být nastavený na stejnou vlnu. Pokud tomu tak není, může být ohrožena kvalita dodávky nebo efektivita vývoje

Zdroj: vlastní zpracování

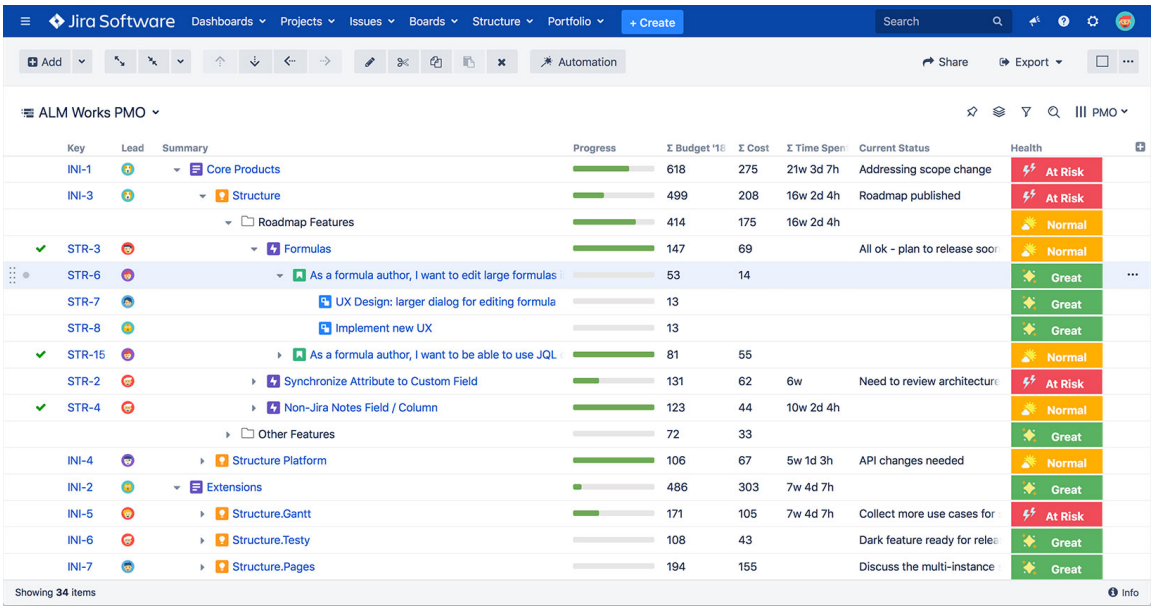
Ze SWOT analýzy o přechodu na agilní metodiku vývoje softwaru vychází celkem 6 silných stránek, 3 stránky slabé, 3 příležitosti a 2 hrozby. Hrozby jsou téměř zanedbatelné, nicméně musíme vzít potaz určité riziko při volbě týmu. Slabé stránky jsou směřovány převážně na správnou motivaci týmu a riziko ne zcela podrobně zpracované dokumentace. V analýze převažují silné stránky a příležitosti, které převyšují zanedbatelné negativní faktory. Proto je mým doporučením zvážit možnost přechodu na agilní vývoj softwaru.

4.4 Optimalizační nástroje vývoje softwaru

Mezi optimalizační nástroje vývoje softwaru lze řadit vše co zrychluje proces a napomáhá ke komunikaci s týmem. Proto jsem se rozhodl vypsat seznam aplikací využívaných na denní bázi umožňujících tvořit zadání a komunikovat se skupinou vývojářů.

4.4.1 JIRA

JIRA je softwarový nástroj pro projektové řízení využívaný napříč IT obory. Umožňuje definici jednotlivých vývojových úkolů, jejich editaci jak ze strany businessu, tak i vývoje. Slouží k oboustranému schvalování. Je databází sprintů, které již proběhly a slouží k přípravě dalšího vývojového období. Následné vyhodnocení správnosti provedení a uzavření vývojového cyklu. JIRA je nástrojem, který je z naší strany využíván na denní bázi a umožňuje nám řídit projekt. Skvělým bonusem je možnost vyhodnocovat úspěšnost a zaznamenávat workflow celého procesu vývoje aplikace LT. (ALM Works, 2020)



Obrázek 31 – JIRA

Zdroj: (ALM Works, 2020)

4.4.2 TFS

TFS, neboli z anglického Team Foundation Server je optimalizačním nástrojem, který přináší společnost Microsoft. Umožňuje spravovat na sebe souvisle navazující zdrojové kódy projektu, reportovat nedostatky kódu a projektově řídit tým vývojářů. Je také využíván testerem, který zde definuje jaká úloha proběhla úspěšně a jaká nikoliv. Pokrývá tak celý životní cyklus vyvíjeného softwaru. My tento software používáme převážně k reportování

vzniklých BUGů (=chybných úkolů) k následnému zapracování. Software disponuje skvěle ovladatelnými analytickými prvky, napomáhajícími určit stav nových, řízených a opravených chyb. Je to nástroj, který šetří velké množství času, protože je centralizovaný a reporty vidí všechny zúčastněné strany.

4.4.3 Skype for Business

Skype for Business, neboli Skype pro firmy, je využívaný napříč celým koncernem ŠKODA AUTO a.s. Umožňuje zrychlovat tok informací a eliminuje čas strávený na cestě na místo konání. Je to tedy efektivní cesta komunikace a řízení vývoje. Optimalizačním prvkem mohou být předpokádány každodenní ranní schůzky, na téma stavu vývoje. Tyto schůzky nesou název Stand Up a umožňují vytvořit v týmu přehled společně s větším kontaktem s vývojáři. Tyto schůzky doplňují plánování a retrospektivu 14 denního cyklu, neboli Sprintu, kde vzniká osobní kontakt pro vydefinování uživatelských úkolů.

4.4.4 Grafický editor

Grafické návrhy pomáhají definovat přibližnou nebo přesnou vizáž vyvíjené komponenty. Námi využívaný software je Malování, Inkscape a případně i Adobe Illustrator CC. Všechny tyto softwary jsou ve ŠKODA AUTO a.s. licencovány, a proto využíváme jejich funce ke znázornění požadavků. Použití nástroje jako je malování může být až komické, ale zcela upřímně nám kombinace aplikace Výstřižky a Malování šetří nepřeberné množství času.

5 Představení LT 2.0 a návrh inovačních komponent

Kapitola návrhu inovačních komponent je zrcadlením náplně práce v rámci mé dlouhodobé stáže ve společnosti ŠKODA AUTO a.s. pod vedení kanceláře Launch Management týmu na oddělení VMP, neboli produktovém marketingu. Zodpovědností Launch Managementu jako takového je koordinace procesu vypouštění modelů automobilů na trh. Pro zlepšení produktivity a efektivního zpracování dat zde před lety započal vývoj optimalizační aplikace, která má za úkol centralizovat data a usnadnit jejich následnou distribuci mezi angažované strany.

5.1 Představení aplikace LaunchTRACK 2.0

Za dobu působení na dlouhodobé stáži aplikace podstoupila velké vizuální změny. Prošla z prostředí SharePoint až k nově navržené aplikaci postavené na platformě .Net Core. Byla formována i nová informační architektura společně s uživatelsky přívětivým designem. Díky vypracované analýze bylo naplněno velké množství předem stanovených cílů a požadavků, které byly kladeny na vývoj nově fungující verze. Aplikace je díky dobře optimalizovanému a udržitelnému kódu rychlá s téměř nulovou odezvou. Vzniklý požadavek pro možnost přizpůsobit UI je zde také splněn a v případě potřeby bude umožňovat rebranding. LaunchTRACK tedy s postupem času „vstává z popela“ a je připraven na první zátěžové testy spokojených uživatelů. Je doopravdy skvělý pocit podílet se na něčem co roste a má to smysluplný směr. Mým úkolem bude v následujících krocích představit vizuální stránku aplikace a následně prezentovat navržené inovační komponenty.

5.1.1 Barevná paleta

Barevná paleta respektuje pravidla a korporátní identitu barev CI/CD. Barevné schéma je zaměřeno na veselejší tóny zelené barvy, které jsou doplněny o jemný bílý podklad společně s černým písmem. Kontrast mezi písmem a pozadím je ideálně volený a všechny informace jsou dobře čitelné. Aplikaci doplňují okrajové barvy jako je například oranžová (nově přidaný launch), červená (varování) anebo jemná modrá (informace).

Obrázek 32 – Základní barevná paleta aplikace LaunchTRACK 2.0
Zdroj: vlastní zpracování

5.1.2 Dashboard

Dashboard, neboli hlavní stránka je tváří celé aplikace. Tvoří první dojem po přihlášení a obsahuje všechny klíčové informace, případně odkazuje na místa, kam můžeme jít pro detailní přehled. Pokud si rozebereme layout stránky, tak jí můžeme rozdělit do několika sekcí. V levém horním rohu vidíme logo aplikace, které je podpořené funkcionalitou odkazování na domovskou stránku. Uprostřed horního panelu získáváme informaci o aktuálním kalendářním týdnu v roce. Je to datový typ, s kterým je pracováno napříč celou aplikací. V pravém horním rohu jsme schopni identifikovat čtyři ikony. Ta, která se nachází nejvíce vpravo je personalizační ikonou umožňující sledovat přehled o vašem uživatelském účtu, anebo případně přistoupit k odhlášení z aplikace. Nalevo od ní se nachází rychlé menu s odkazy na relevantní stránky. Další ikonou je notifikační zvoneček reprezentující změny vázané na váš uživatelský učet. Poslední položkou je administrativní ikona schvalování Workflow centra. Pokud proběhne nějaká změna například data u milníku, tak musí administrátor ověřit a schválit jeho správnost. Pod horní lištou vidíme sekci Model launches, která prezentuje všechny aktivní Launche, které jsou aktuálně v procesu. Pod konkrétním modelem jsou zaznamenány nejzásadnější informace a tři ikony rychlých odkazů. Výhodou nové verze je kompletní responzivita, proto můžeme pod sekcí Model launches zaznamenat horizontální scrollbar umožňující pohyb mezi modely. Dále se na stránce objevuje rozčlenění na NEW Models, Facelifts a Derivatives. Tato sekce nám umožňuje rychlé filtrování mezi modely. Ve spodní části stránky se nacházejí rychlé odkazy, které pomáhají přistoupit na jiné stránky v rámci aplikace. V pravém sloupci se ukazují HQ oznámení předávající ty nejzásadnější informace, například pro možné ohrožení launche.

Obrázek 33 – Dashboard aplikace LaunchTRACK 2.0

Zdroj: (Materiály ŠA, LaunchTRACK 1.0, 2020)

5.1.3 Launch Plan Detail

Stránka Launch Plan Detailu zaznamenává strukturu milníků nadefinovaných administrátory aplikace LaunchTRACK. Jde o jakýsi plán prezentující podstatné body provázející průběh vypouštění modelu automobilu na trh. Jako uživatel dostávám přehled týmů, milníků a aktivit na launch vázaných. Některé milníky spadají pod správu administrátorů – ty musí být vyplněny a některé si každý uživatel může nadefinovat sám v případě zájmu vytvoření jiného záchytného bodu pro postupné vypouštění na trh. Ve stromové tabulce vidíme název milníku, odpovědnou osobu, status aktuálního milníku, jeho začátek a konec, datum modifikace a osobu, která provedla poslední změnu. Pokud je zapotřebí detailnější informace o milníku nebo aktivitě, tak klikneme na jeho název a následně se zobrazí Pop-Up okno s bližšími informacemi. Pomocí kontextového selektoru v pravém horním rohu pod ikonami jsme schopni přepínat mezi jednotlivými trhy. Dokážeme se tedy podívat jak na strukturu České republiky, tak i Rakouska. V záhlaví stránky můžeme identifikovat infopanel, který přibližuje informace k danému procesu.

Obrázek 34 – Launch Plan Detail

Zdroj: (Materiály ŠA, LaunchTRACK 2.0, 2020)

5.1.4 Administrace

Ukázka administrace nám prezentuje možnosti administrátora spravovat jednotlivé sekce informačního systému LaunchTRACK 2.0. Na obrázku pod textem, také můžeme zaznamenat fungující responzivní design, který umožňuje zalamovat komponenty navigace pod sebe a stránka je tedy stavěna i pro zobrazení na mobilních zařízeních. Přehled stránky administrace je klíčovým rozcestníkem pro administrátora, který má přehled o uživatelích využívajících aplikaci. Má možnost tvořit nové týmy a zakládat markety. Sbírat feedbacky od importérů, schvalovat data spadající do workflow a vytvářet nové Launche. Tato stránka nabízí široké využití, které je běžnému uživateli skryto.

Obrázek 35 – Administrace aplikace LT 2.0

Zdroj: (Materiály ŠA, LaunchTRACK 2.0, 2020)

5.2 Uživatelské role

Uživatelské role lze rozčlenit do tří základních kategorií, kterými je Administrátor, zastupitel centrály (HQ) a uživatelská role importéra konkrétních trhů (IMP). Kancelář Launch Managementu týmu zastává roli administrátorů a schvalovatelů procesu. Mezi HQ Editory patří jednotlivá oddělení ŠA jako například PR, Fleet nebo Marketing Communication. Mezi IMP je zařazeno TOP 25 marketů/trhů, do kterých jsou modely ŠA importovány.

5.2.1 Administrátor

Administrátor je zodpovědný za definici centrálního plánu. Komunikaci a vyladění detailů s trhy. Koordinaci a kontrolu zadávaných dat do aplikace. Je to správce, který udržuje čistý chod aplikace a schvaluje požadavky z trhů a HQ ŠA. Administrátor má možnost nahlížet do struktury všech uživatelských rolí.

5.2.2 HQ Editor

HQ editor je zaměstnanec ŠKODA AUTO a.s., který náleží do jednoho z týmů definovaných strukturou oddělení. Má na starosti vyplňovat v aplikaci potřebné termíny a informace vázané k milníku, za který je zodpovědný. HQ editor má možnost nahlížet do struktury všech importérských uživatelských rolí.

Příklady týmu vytvořených dle příslušného oddělení:

- Product marketing
- Sales planning
- Price
- PR

5.2.3 IMP Editor

Importér editor je pověřen vyplňováním struktury milníků, tak, aby předával informaci centrále ŠA. Jeho další kompetencí je definovat ME a informování kanceláře Launch Managementu zda vypuštění modelu proběhlo úspěšně. Po uplynutí nadefinovaného časového intervalu od ME je povinností importéra zaslat feedback marketingové kampaně připravované v rámci daného trhu společně s feedbackem na průběh launche.

5.2.4 HQ/IMP Reader

Role readera je určena pro uživatele, kteří nejsou schopni editovat žádné záznamy v rámci aplikace. Mohou jen nahlížet a číst. Tato role bude vhodná pro vedení a zastupitelstvo ŠA, které si do aplikace bude přicházet pouze pro informace.

Obrázek 36 - Uživatelské role aplikace LT

Zdroj: (Materiály ŠA, LaunchTRACK 2.0, 2020, vlastní zpracování)

5.3 Návrh inovačních komponent

Návrh a vydefinování inovačních komponent patřil mezi jednu z obsáhlejších aktivit, kterou jsem společně s týmem v rámci dlouhodobé praxe vykonával. Proto jsem se rozhodl uzavřít výstup bakalářské práce ukázkou navrhovaných komponent. Proces návrhu byl následovný. Vždy bylo nejdříve provedeno několik brainstormingů, na kterých se probíraly možnosti návrhu nových funkcionalit. Po uzavření těchto kolikrát i čtyřhodinových setkání byla sepsána business logika komponenty, která byla představena IT analytikovi. Společně s ním byla vyhodnocena logika navrhovaných celků. Dále se přistoupilo na vizuální návrh, který byl řešen pomocí nástrojů jako Inkscape, Adobe Illustrator CC, ale i obyčejný program Malování, pro rychlý návrh. Dále byly návrhy porovnány, vybrán byl ten, který se z pohledu vizuální stránky, použitelnosti a uživatelské přívětivosti jevil jako nejlepší. Ten byl následně rozepsán do prostředí JIRA a zadán na konkrétního vývojáře. Po ukončení vývojářovi činnosti a nasazení komponenty na testovací prostředí byla funkce otestována a v případě požadované kvality zpracování podle zadání schválena.

5.3.1 Analytický nástroj Market overview

Analytický nástroj pro market overview bude mít za úkol vytvořit hodnocení průběhu vypouštění jednotlivých modelů na trh. Toto vyhodnocení bude graficky zpracováno v koláčovém grafu a určí tak tedy jakém stavu se launch právě nachází. Kolik je ještě potřeba splnit milníků pro možnost přechodu do fáze evaluace. Dále se k této funkci přidá možnost znázornění milníků, které jsou připsány přímo na konkrétního uživatele. Milníky, u kterých je uživatel nastavený jako zodpovědná osoba budou vykresleny v poli Assigned to me.

Obrázek 37 - Grafický návrh analytického zpracování procesu Market overview

Zdroj: vlastní zpracování

5.3.2 Uživatelská personalizace

Uživatelská personalizace je inteligentní nástroj, který napomáhá přizpůsobit každému uživateli pracovní prostředí. Úkolem tohoto nástroje je ne zcela rozvolnit pravidla návrhu individuálního prostoru a zároveň umožnit uživateli dostatečnou vizuální i logickou úpravu osobního účtu. Prozatím se komponenta bude vztahovat v rámci testování pouze na hlavní stránku, kde bude mít uživatel možnost řadit ascendentně nebo descendentně modely launchů v horní sekci vizualizace karet modelů. Dále bude funkcionalita disponovat možností zvolit si své oblíbené modely, které jsou pro jeho práci zásadní a tudíž je potřebuje prioritizovat a dynamicky připnout na začátek, tak, aby se objevovaly na prvních pozicích.

Pro výběr svého oblíbeného launche zaklikne symbol "hvězdičky". Následně se předřadí vybraný model v karuselu na přední pozice a dále budou následovat neoznačené launche. Hvězdička bude ve výchozím zobrazení prázdná (bílý vnitřek) se žlutým obrysem a po vybrání se zobrazí plně vybarvená žlutě. Hvězdička se bude zobrazovat jak u launche v karuselu, tak v sekci na hlavní stránce current model launches. Tato komponenta je prvním krokem k uživatelsky editovatelnému prostředí, které navyšuje kvality použitelnosti.

- Další z možností uživatelské optimalizace je přístup k nadefinování rychlých odkazů na hlavní stránce.
- Drag&Drop funkcionalita v rámci posouvání milníků, anebo také jednotlivých komponent na stránce mezi sebou. Pro tuto možnost by musela být připravena šablona, do které by byl zasazen kontext manipulace s komponenty.
- Mezi další návrh patří případné skrytí nerelevantního launche z dashboardu.

Obrázek 38 - Rozbalené menu uživatelské personalizace

Zdroj: vlastní zpracování

5.3.3 Komponenta interaktivních feedbacků

Sběr feedbacků je jednou z důležitých činností Launch Management týmu. Tato aktivita je poměrně časově náročná, a proto vznikl v rámci týmu požadavek pro vytvoření sběru feedbacků pomocí systému LaunchTRACK. Komponenta automatického sběru feedbacků byla připravena ve dvou zněních a to:

- **Staticky řešené feedbacky –** Umožňující uživatelům stáhnout předpřipravenou šablonu v program MS PowerPoint, kterou následně vyplní a nahrají jí zpět. Jedná se o řešení, které je méně finančně náročné, z hlediska funkčnosti použitelné, ale s ohledem na uživatelskou přívětivost ne zcela dostačující. Uživatel musí podstoupit značné množství kroků pro to, aby feedback úspěšně stáhl, otevřel v externím programu a následně feedback nahrál zpět.

Obrázek 39 - Struktura nahrávání feedbacků

Zdroj: vlastní zpracování

- **Dynamický formulář feedbacku –** je navrhovaná komponenta, která by podpořila uživatelskou přívětivost společně s rychlejším vyplňování pro importéra a možností využít analytické nástroje k vyhodnocení zaslaných feedbacků. Uživatel ze strany importéra by tedy uspořil svůj čas, stejně jako administrátor, který by nemusel nahrát šablonu feedbacku a následně stahovat vyplněný soubor a analyzovat ho. Tyto tři kroky by za něj vykonal systém, a proto by tato komponenta přinesla optimalizaci času, ale také úsporu uložiště na využívaném zařízení.

5.4 Zhodnocení a doporučení

V rámci zhodnocení je vhodné shrnout celkový postup tvorby praktické části bakalářské práce. Prvním krokem bylo zanalyzovat aplikaci LaunchTRACK 1.0, kde jsme využili SWOT analýzy k získání potřebných informací o silných/slabých stránkách, příležitostech a hrozbách. Bylo tedy zřejmé na čem je vhodné začít stavět, a co naopak potlačit. Dále byla využita heuristická analýza, na které se podíleli tři nezávislí respondenti ze společnosti ŠA. V heuristické analýze byly získány informace o přístupnosti, layoutu a vizuální stránce, IA a navigaci, chybách a zpětné vazbě. Tyto čtyři heuristiky byly následně graficky znázorněny pomocí paprskových diagramů, kde byly bezpečně zachyceny všechny směry vybraných heuristik. Neméně důležitá byla definice návrhu optimalizačního přístupu k vývoji softwaru a následný návrh inovačních komponent.

Návrhy inovačních komponent byly konstruovány tak, aby přinesly do systému další cesty a příležitosti jak optimalizovat celkový chod aplikace společně s procesy fungování Launch Management týmu. Cílem komponent a aplikace jako takové je, bylo a bude zautomatizovat všechny fáze vypouštění modelů na trh. Díky analýze a návrhu řešení je možné rozvíjet aplikaci novým směrem a pro budoucí spolupráci s trhy a centrálou ŠA, tak ušetřit značné množství času, které by bylo místo sdílení informací skrz centralizovaný systém řešeno emailovou korespondencí se značnou časovou prodlevou odpovědí. K úspěšnému fungování aplikace budou ovšem potřeba angažovaní a motivovaní uživatele. Tato motivace bude zajištěna s ohledem na ušetřený čas a řadu nástrojů, které umožní zefektivnit ukládání a práci s dokumenty.

Závěr

Cílem teoretické části bylo poskytnout strukturu pro pochopení základních pojmů vývoje softwaru. Byly rozebrány jednotlivé fáze životního cyklu vývoje společně s konkrétními rolemi vývojářského týmu. V souvislosti s tématem byly definovány metodiky vývoje softwaru, jejich výhody, nevýhody a možné případy využití v praxi. Celkový pohled na problematiku byl čtenáři doplněn o pojmy UX a UI (User eXperience a User Interface), které reprezentují silné spojení umožňující dodávat vizuálně čistý a uživatelsky přívětivý softwarový produkt. Byl popsán UX proces, který přistupuje k návrhu uživatelsky přívětivého prostředí.

Praktická část bakalářské práce využívá nabytých zkušeností z teoretických podkladů, které byly aplikovány na vytvoření analýzy softwarového řešení aplikace LaunchTRACK 1.0. Byla vytvořena SWOT analýza, která umožnila vyhodnotit silné/slabé stránky, příležitosti a hrozby. Dále bylo využito tří na sobě nezávislých respondentů, zaměstnanců ŠKODA AUTO a.s., kteří měli za úkol otestovat čtyři přístupy heuristické analýzy. Výsledky, které vycházely z analýz, byly vyhodnoceny a na jejich základě vznikly definice pro vývoj nové verze.

Primárním přínosem této bakalářské práce je vyhodnocení analýzy, nadefinování cílového stavu nové verze a následný návrh inovačních komponent. Tyto komponenty umožňují posunout aplikaci LaunchTRACK 2.0 na další úroveň. Snaha o optimalizaci procesu vývoje aplikace se stala důležitým bodem diskuse s předpokladem přechodu na agilní metodiku vývoje softwaru.

Přínosy práce jsou podpořeny návrhem optimalizačních procesů Launch Management týmu v rámci oddělení VMP, neboli produktového marketingu. Aplikace LaunchTRACK 2.0 byla navržena, tak aby odpovídala optimalizačním požadavkům a struktuře procesů. Softwarové řešení bude v dohledné době uvedeno na produkční prostředí ŠKODA AUTO a.s. a vstoupí tak znovu do světa reálného využití. Bakalářskou práci by bylo možné rozšířit o zpětnou vazbu uživatelů, ze zahraničních trhů a centrály ŠKODA AUTO a.s. Na základě těchto vyhodnocených feedbacků by bylo možné vytvořit nové inovační nástroje podporující uživatelskou přívětivost.

Seznam použité literatury

Almworks. 2020. JIRA [online]. [cit. 2020-07-18]. Dostupné z: https://almworks.com

ANDERSON, Stephen. 2012. *Přitažlivý interaktivní design: jak vytvářet uživatelsky přívětivé produkty*. Brno: Computer Press. ISBN 978-80-251-3722-2.

BUCHALCEVOVÁ, Alena. 2005. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. Praha: Grada. Management v informační společnosti. ISBN 80-247-1075-7.

BRDA, Jiří. 2016. *Co je UX design a kde se s ním setkáte?* [online]. [cit. 2020-07-21]. Dostupné z: http://www.jiribrda.cz/co-je-ux-design-a-kde-se-s-nim-setkate.html

DAWSON, Neil. 2020. *Eye Tracking: What Is It For And When To Use It* [online]. [cit. 2020-07-17]. Dostupné z: https://usabilitygeek.com/what-is-eye-tracking-when-to-use-it/

Experience UX. 2020. *What is Card Sorting?* [online]. [cit. 2020-07-21]. Dostupné z: https://www.experienceux.co.uk/faqs/what-is-card-sorting/

Fidelity, High. 2019. *6 Factors Influencing Employee Morale and How They Can Strengthen Your Remote Team* [online]. [cit. 2020-07-20]. Dostupné z: https://www.highfidelity.com/blog/factors-influencing-employee-morale-for-remote-teams

FILIPOVA, Olga. 2018. *Software Development From A to Z: A Deep Dive Into All the Roles Involved in the Creation of Software*.1.vyd. Brno: Zoner Press. ISBN 978-14-8423-945-2.

GORDON, Kelley. 2020. 5 Principles of Visual Design in UX. *Nielsen Norman Group* [online]. [cit. 2020-07-18]. Dostupné z: https://www.nngroup.com/articles/principles-visual-design/

Graphic design junction. 2020. *Top 7 UX Topics All Beginners Need to Know* [online]. [cit. 2020-07-19]. Dostupné z: http://graphicdesignjunction.com/2018/12/ux-topics-all-beginners-need-to-know/

HARTSON, Rex. 2012. *The UX book: process and guidelines for ensuring a quality user experience*. 2. vyd. Waltham, MA: Morgan Kaufmann. ISBN 978-0-12-385241-0.

KONÍČEK, Petr. 2016. *Co je to A/B testování?* [online]. [cit. 2020-07-18]. Dostupné z: https://www.objevit.cz/co-je-to-ab-testovani-t176297

KRUG, Steve. 2010. *Nenuťte uživatele přemýšlet!: praktický průvodce testováním a opravou chyb použitelnost* [sic] webu. Brno: Computer Press. ISBN 978-80-251-2923-4.

KUIPERS, Arthur. 2020. UX design vs UI design [online]. [cit. 2020-07-21]. Dostupné z: https://www.interaction-design.org/literature/article/what-is-interaction-design

MACIASZEK, Leszek A. a Bruc Lee LIONG. 2005. *Practical software engineering: a case study approach*. Boston: Addison-Wesley. ISBN 03-212-0465-4.

Managementmania. 2017. SWOT Analýza. *Managementmania.com* [online]. Wilmington (DE) [cit. 2020-07-20]. Dostupné z: https://managementmania.com/cs/swot-analyza

MARSH, Joel. 2019. *UX pro začátečníky: (rychlokurz - 100 lekcí)*. Páté vydání. Brno: Zoner Press. ISBN 978-80-7413-397-8.

MARTINŮ, Jiří a Petr ČERMÁK. 2018. *Metodiky vývoje software, studijní podpora pro kombinované* [online]. [cit. 2020-07-24]. Dostupné z: https://dl1.cuni.cz/pluginfile.php/886974/mod_resource/content/1/Metodiky-vývoje-software-studijn%C3%AD-text.pdf

MCKAY, Everett. 2013. *UI is communication: how to design intuitive, user centered interfaces by focusing on effective communication*. Boston: Elsevier, Morgan Kaufmann. ISBN 978-0123969804.

McKay, Jory. 2019. *Software Development Process* [online]. [cit. 2020-07-21]. Dostupné z: https://plan.io/blog/software-development-process/

MICHL, Jakub. 2020. *Co je to brand, co je to branding?* [online]. [cit. 2020-07-21]. Dostupné z: https://medium.com/@jakubmichl/co-je-to-brand-co-je-to-branding-67e6b633d29

MINHAS, Saadia. 2018. *User Experience Design Process* [online]. [cit. 2020-07-18]. Dostupné z: https://uxplanet.org/user-experience-design-process-d91df1a45916

MORAN, Kate. 2019. Setup of an Eyetracking Study. *Nielsen Norman Group* [online]. [cit. 2020-07-22]. Dostupné z: https://www.nngroup.com/articles/eyetracking-setup/

MORRIS, Scott. 2020. WHAT IS A BACK END DEVELOPER?, *Skillcrush* [online]. [cit. 2020-07-16]. Dostupné z: https://skillcrush.com/blog/what-is-a-back-end-developer/

NIELSEN, Jakob. 1994. 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group* [online]. [cit. 2020-07-20]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

NIELSEN, Jakob. 1997. How Users Read on the Web. *Nielsen Norman Group* [online]. [cit. 2020-07-20]. Dostupné z: https://www.nngroup.com/articles/how-users-read-on-the-web/

NIELSEN, Jakob. 2000. Why You Only Need to Test with 5 Users. *Nielsen Norman Group* [online]. [cit. 2020-07-20]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

NIELSEN, Jakob. 2012. How Many Test Users in a Usability Study? *Nielsen Norman Group* [online]. [cit. 2020-07-20]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/

Přístupnost. 2020. O přístupnosti, *Přístupnost.cz* [online]. [cit. 2020-04-20]. Dostupné z: http://www.pristupnost.cz/o-pristupnosti/

PacktPub. 2020. Usability testing methodologies.[online]. [cit. 2020-07-21]. Dostupné z: https://subscription.packtpub.com/book/web_development/9781788999045/1/ch01lvl1sec10/usability-testing-methodologies

REISS, Eric. 2012. Usable usability: simple steps for making stuff better. Indianapolis, IN: Wiley. ISBN 978-1-118-18547-6.

ROSENFELD, Louis a Peter MORVILLE. 2002. *Information Architecture for the World Wide Web*. 2nd Edition. Sebastopol: O'Reilly. ISBN 0-596-00035-9.

ŘEZÁČ, Jan. 2016. *Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů.* Vydání druhé. [Brno]: House of Řezáč. ISBN 978-80-270-0644-1.

SEMERÁDOVÁ, Tereza a Petr WEINLICH. 2018. Uživatelské parametry webového designu. Liberec: Technická univerzita v Liberci. ISBN 978-80-7494-448-2.

TRAVIS, David. 2016. 247 web usability guidelines. *Userfocus* [online]. London [cit. 2020-07-21]. Dostupné z: https://www.userfocus.co.uk/resources/guidelines.html

Usability. 2020. Interaction Design Basics. *Usability.gov* [online]. [cit. 2020-07-21]. Dostupné z: https://www.usability.gov/what-and-why/interaction-design.html

UNGER, Russ. c[2012]. *A project guide to UX design: for user experience designers in the field or in the making*. Second edititon. Berkeley: New Riders. ISBN 978-032-1815-385.

WEINSCHENK, Susan. 2012. *100 věcí, které by měl každý designér vědět o lidech*. Brno: Computer Press. ISBN 978-80-251-3649-2.

Existek. 2017. *SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral* [online]. Ukraina [cit. 2020-07-28]. Dostupné z: https://existek.com/blog/sdlc-models/

SCHER, Paula. 2020. About Paula Scher [online]. [cit. 2020-07-21]. Dostupné z: https://www.pentagram.com/about/paula-scher

SIANG, Teo. 2020. What is Interaction Design? *The Interaction Design Foundation* [online]. [cit. 2020-07-21]. Dostupné z: https://www.interaction-design.org/literature/article/what-is-interaction-design