TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Řízení a zavádění nástrojů automatizace v softwarovém vývoji

Diplomová práce

Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Autor práce:	**Bc. Dominik Švarc**
Vedoucí práce:	Mgr. Tomáš Žižka, Ph.D. Katedra informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Zadání diplomové práce

Řízení a zavádění nástrojů automatizace v softwarovém vývoji

Jméno a příjmení: **Bc. Dominik Švarc**
Osobní číslo: E18000357
Studijní program: N6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika
Zadávající katedra: Katedra informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Agilní způsoby řízení vývoje ERP systému a jeho metodiky.
2. Informační systém ABRA Gen a jeho moduly.
3. Proces vývoje a vydání verze ERP systému ABRA Gen.
4. Inovace automatizace testování.
5. Zhodnocení nových přístupů k řízení a přínosu automatizace.

Rozsah grafických prací:
Rozsah pracovní zprávy: 65 normostran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

- *AutoIt v3: Your Quick Guide*, 2007. Sebastopol, California: O'Reilly Media. ISBN 978-0-596-51512-6.
- CANTU, Marco, 2016. *Object Pascal Handbook*. Piacenza, Italy: Marco Cantu. ISBN 978-1514349946.
- MYSLÍN, Josef, 2016. *Scrum: průvodce agilním vývojem softwaru*. Brno: Computer Press. ISBN 978-80-251-4650-7.
- PROQUEST. 2019 *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. Dostupné z: http://knihovna.tul.cz
- ŠOCHOVÁ, Zuzana a Eduard KUNCE, 2019. *Agilní metody řízení projektů*. 2. vydání. Brno: Computer Press. ISBN 978-80-251-4961-4.

Konzultant: Mgr. Jan Rychlík

Vedoucí práce: Mgr. Tomáš Žižka, Ph.D.
Katedra informatiky

Datum zadání práce: 31. října 2019
Předpokládaný termín odevzdání: 31. srpna 2022

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 diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou 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 diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou 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 diplomová 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. srpna 2022 Bc. Dominik Švarc

Řízení a zavádění nástrojů automatizace v softwarovém vývoji

Anotace

Práce se obecně zabývá tématem způsobu řízení vývoje softwaru. Primárně je zaměřena na agilní způsoby, zejména metodiku Scrum a její nástroje. Dále se zabývá možnostmi využití automatizace testování v procesu vývoje. Veškeré praktické ukázky pramení ze zkušeností a práce s produktem Abra Gen společnosti Abra Software, na jehož tvorbě se autor mohl v rámci své pracovní pozice spolupodílet.

Účelem práce je přiblížení problematiky agilního způsobu řízení s využitím metodiky Scrum. Poskytnutí základního náhledu na proces vývoje komplexního informačního systému a změn, kterými může projít, seznámení s využitím základních nástrojů zpřehledňujících vývoj softwaru. Ukázka možností automatizace testování a v neposlední řadě vyhodnocení získaných poznatků pramenících z praktických zkušeností zkoumané problematiky.

Vlastním přínosem autora je spolupráce na zavádění skriptingové a vizuální GUI automatizace testování, zejména programování prvního skriptingového testu a následného sledování průběhu vývoje. Zavedení evidence rychlostních testů a sledování a kontrola průběhu vývoje předzveřejňovacích vizuálních GUI testů.

Klíčová slova

agilní, Agilní manifest, AutoIt, automatizace, Backlog, Definition of Done, Delphi, ERP, informační systém, plánování, Product Owner, retrospektiva, Scrum, Scrum Master, Sprint, testování

Management and implementation of automation tools in software development

Annotation

The works subject deals with the ways of managing software development. It primarily focuses on agile ways of management, mainly the methodology of Scrum and its tools. Furthermore, it deals with the possibilities regarding the automatization of testing during the process of development. Every practical example is coming from a work experience with the product Abra Gen of Abra Software, on which the author of this thesis participated in its development.

The aim of this work is to také a closer look at the issue of agile management with the use of the Scrum methodology. Further, to provide a basic overview of the process during the development of a complex information system and its possible changes it might go through. Additionally, the familiarization with the basic tools for clarifying software development. Next an example of choices for automatization of testing and lastly the evaluation of the insight collected from practical experience with the examined issue.

Author's own contribution lays in the cooperation during the implementation of scripting and visual GUI automatization of testing. Mainly programming the first scripting test and its consequential observation during advancements. Implementing register of speed tests and finally the inspection of the development processes for pre-release visual GUI tests.

Key Words

agile, AutoIt, automation, Backlog, Definition of Done, Delphi, ERP, Information System, Manifesto for Agile Software Development, planning, Product Owner, retrospective, Scrum, Scrum Master, Sprint, testing

Obsah

Seznam zkratek

ARES	Administrativní registr ekonomických subjektů
EET	Elektronická evidence tržeb
ERP	Enterprise Resource Planning
GUI	Graphic User Interface
MSSQL	Microsoft SQL Server
OLE	Microsoft Object Linking and Embedding
R&D	Research and Development
RTTI	Run-Time Type Information
SQL	Structured Query Language
SVN	Subversion/Apache Subversion

Seznam tabulek

Seznam obrázků

Úvod

Téma této diplomové práce bylo zvoleno na základě autorem nabytých zkušeností, v průběhu spolupráce na vývojovém oddělení. Autor měl možnost se velkou mírou podílet na zavádění automatizace testování ve vývoji komplexního informačního systému Abra Gen. Během svého výkonu měl také možnost absolvovat změnu způsobu řízení vývojového oddělení na agilní. Náplní jeho práce byla příprava prvního automatického skriptingového testu. Jako další byla zavedena evidence rychlostních testů a fungoval také na pozici Scrum Mastera. Cílem této diplomové práce je provedení rozboru problematiky agilního způsobu řízení, zejména metodiky Scrum a vyhodnocení přínosu automatizace procesů testování.

Diplomová práce je rozdělena do dvou dílčích částí. První část se zabývá teoretickým rozborem problematiky agilního způsobu řízení vývojového oddělení. Druhá část je věnována počátkům zavádění automatických testů.

V první části práce se čtenář může seznámit se základními pojmy týkající se dané problematiky. První kapitola je věnována představení a definici agilního způsobu vedení. S manifestem agilního řízení vývoje softwaru je zde představeno jeho dvanáct principů a naznačeno co obnáší a přináší agilní transformace. Druhá kapitola se týká konkrétně metodiky Scrum a jejích nástrojů. Jsou zde představeny základní nástroje pro bezproblémový chod vývojového týmu a zobrazeno schéma procesů metodiky Scrum. Ve třetí a čtvrté kapitole je představena společnost Abra Software a.s. a její produkt Abra Gen.

Druhá část práce se zabývá možnostmi automatického testování komplexního informačního systému. V páté kapitole jsou popsány jednotlivé možnosti využití automatizace testování. V poslední kapitole jsou získané poznatky a zkušenosti, společně s vypozorovanými přínosy, vyhodnoceny.

Závěrečná práce si klade za cíl prokázat schopnosti studentů aplikovat teoretické a praktické dovednosti, které získali během svého studia.

1. Agilní způsoby řízení vývoje softwarových produktů

V souvislosti s agilními způsoby řízení projektů je potřeba ujasnit a vysvětlit význam slova agilní. Za ním se ukrývá schopnost rychle, pružně, efektivně a přehledně reagovat na změny. Agilní může znamenat být dynamický, hbitý, rychlý, přizpůsobivý, hravý nebo rychle reagující na změnu. (Šochová, 2019)

Agilní přístup podporuje, vymezuje a mění způsob řešení projektů, problémů a myšlení. Dalo by se říct, že je to jiný způsob života, upřednostňující jiné hodnoty. Jedná se o jiný možný smysl, respektive filozofii přístupu k řešení vedení projektů. (Šochová, 2019)

Agilní přístup spočívá zejména ve spolupráci, komunikaci a připravenosti a ochotě reagovat na změny. Ve své podstatě se také snaží o omezení nadbytečné byrokracie, složitých předčasných analýz, které bez komunikace se zákazníkem mohou postrádat smysl nebo nadbytečné dokumentace méně významných událostí. Snaží se o zjednodušení procesů s možností jednoduchých, efektivních a rychlých změn v závislosti na požadavcích zákazníka. Jde o to, že v danou chvíli se realizuje jen to, co má v konkrétní moment smysl. Základním principem je dodání kvalitního a funkčního softwaru. Vše ostatní by mělo být podřízeno této myšlence. Vše ostatní je pouhý nástroj. (Šochová, 2019), (Myslín, 2016)

Nejedná se o žádný striktní způsob řízení procesů, ale také se nejedná o žádný chaos a nepořádek. Má jasně specifikovaná pravidla a metodiky, kterými je možné se řídit. Tento přístup pouze vymezuje hranice a zmenšuje pole „hry“ tak, aby si projektové týmy sami mohly určovat svá vlastní pravidla a postupy. Tak, aby se jim pracovalo co nejlépe, byly produktivní, efektivní a dodaly kvalitní produkt v co nejkratším čase. Takové týmy jsou zaměřené na hodnoty, které mohou přinést zákazníkovi. Snaží se mu vyjít vstříc tak, aby byl maximálně spokojený a za své vynaložené prostředky dostal to, co opravdu potřebuje a to, co mu může pomoci. Být agilní znamená žít těmito myšlenkami, jedná se o zcela jinou firemní kulturu a náladu. (Šochová, 2019), (Myslín, 2016)

Agilní přístupy vývoje softwaru zaštiťují různé metodiky, pravidla a frameworky. Základním stavebním prvkem je však agilní manifest uvedený v roce 2001 skupinou vývojářů, který ve čtyřech základních pravidlech a dvanácti principech shrnuje to, co znamená být agilní. (Bednárová, Bogdanovská, Mojžišová, Pócsová, 2020)

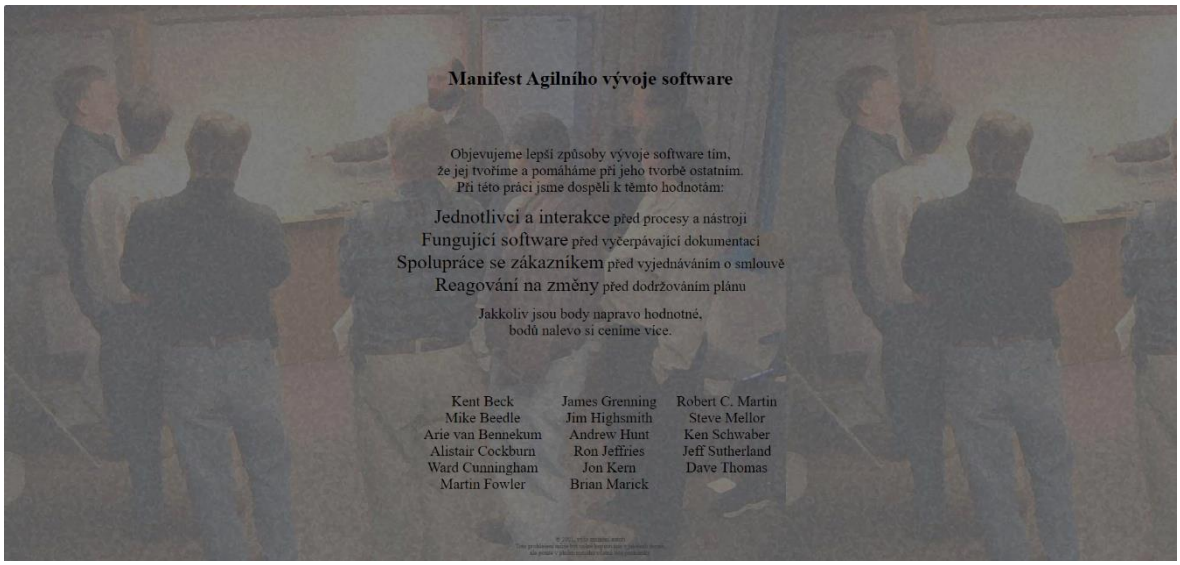
1.1 Manifest agilního vývoje softwaru

Manifest Agilního vývoje softwaru definuje nejzákladnější pravidla agilních přístupů. Dostupný je ve spoustě jazykových mutacích na adrese https://agilemanifesto.org/.

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

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

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



Obrázek 1: Manifest agilního vývoje softwaru
Zdroj: https://agilemanifesto.org

1.1.1 Jednotlivci a interakce před procesy a nástroji

Procesy a nástroje jsou důležité, nicméně pro agilní metodiky jsou důležití jednotlivci, jejich potřeby a jejich vlastní přínos projektu. Úspěch je založen na spolupráci a komunikaci. V obecné rovině lze tvrdit, že spolupracující týmy mají lepší výsledky než skupiny samostatně pracujících jednotlivců. Nástroje a procesy jim mají ve skutečnosti jen sloužit

a pomáhat v dosažení výsledků. Nejsou pro jejich úspěch tolik klíčové. Toto pravidlo říká, že tým by si měl sám vybrat takové nástroje, které mu opravdu pomohou k dosažení kvalitního výsledku. K tomu je ale zapotřebí komunikace a vlastní přínos členů. (Šochová, 2019), (Myslín, 2016)

1.1.2 Fungující software před vyčerpávající dokumentací

Dokumentace je určitě důležitá součást procesu vývoje, ale rozhodně by neměla převážit nad výsledným produktem. Primárně by měla sloužit jako pomůcka pro uživatele a tvůrce pro popis a vysvětlení částí, které nejsou tolik intuitivní a snadno pochopitelné. Vyčerpávající dokumentace o tisíce stranách nejsou z dlouhodobého hlediska snadno udržovatelné. V některých případech dokumentace po letech zdaleka neodpovídá aktuálnímu stavu funkcionalit. Zpravidla se dokumentace vytvoří se vznikem funkcionality, ale při dalších úpravách už se příliš neřeší. „Po nás ať přijde potopa.“ Často platí, že dobrý programátor je velice lenivý člověk a čemu se může vyhnout tomu se vyhne. Pro vývojáře je tvorba dokumentace často velice obtěžující a zbytečná část práce. Dokumentace by měla být stručná a poskytovat klíčové informace. S odkazem na první pravidlo „Jednotlivci a interakce před procesy a nástroji“ lze tvrdit, že složité, vyčerpávající a nadbytečné dokumentaci lze předejít dobrou komunikací v týmu. V případě dobré komunikace se analytici, testeři a vývojáři dokážou sami nejlépe domluvit co a v jaké podobě je potřeba dokumentovat pro „budoucí generace“. Úroveň a míra dokumentace by měla být taková, aby čas, který byl investován do psaní dokumentace odpovídal hodnotě, kterou z ní čtenáři načerpají. Agilní přístupy prosazují přirozenou dokumentaci, kdy vývojáři dokumentují průběžně to, co má smysl. Odmítají hodiny strávené nad dokumentací, kdy se chudák vývojář potí u toho, že musí využívat předepsaný formát, jazykové prostředky a musí psát pomocí standardizovaných šablon a formulářů. (Šochová, 2019), (Myslín, 2016)

1.1.3 Spolupráce se zákazníkem před vyjednáváním o smlouvě

Smlouvy jsou důležité, ale neměly by nahrazovat komunikaci a spolupráci. Smluvní jednání bývají často velice komplikovaná, účastní se jich většinou vedoucí pracovníci, management, právníci, ladí se nejmenší detaily. Výstupem ale často bývá jen precizně sepsaná smlouva nikoliv dobrý software. Proto agilní přístup prosazuje spolupráci před vyjednáváním

dokonalých smluv. Ústní komunikace je považována za standard. Pouze spokojený zákazník, který dostal to, co potřebuje a pomůže mu, může poskytnout doporučení dalším potenciálním zákazníkům. Naproti tomu zákazník, který sice má to, co si objednal a ve smlouvě podepsal, ale nijak mu to nepomohlo, se příště podívá po jiném dodavateli. (Šochová, 2019), (Myslín, 2016)

1.1.4 Reagování na změny před dodržováním plánu

Časové harmonogramy mohou sloužit jako vodítko, ale neměly by ve výsledku výrazněji ovlivňovat naplňování potřeb zákazníků. Agilní přístupy neprosazují striktní dodržování plánů, protože změny jsou u vývoje softwarových produktů nevyhnutelné. Pro naplnění potřeb zákazníka je potřeba být vždy připraven na změnu, která zaručeně přijde. Zákazníci často nemají dokonalou představu o tom, co chtějí a neumějí si prakticky představit co potřebují, proto je tolik potřebná ta stále dokola omílaná komunikace. Při vývoji je potřeba udržovat kontakt se zákazníkem a analyzovat a řešit jeho měnící se požadavky. Striktní dodržování plánu bez průběžné komunikace se zákazníkem může vést k větší katastrofě než průběžné řešení jeho měnících se požadavků. (Šochová, 2019), (Myslín, 2016)

1.2 Dvanáct principů agilního vývoje softwaru

Vedle čtyř základních pravidel vymezujících hrubý rámec agilního způsobu řízení existuje seznam dvanácti principů agilního vývoje softwaru, které jej hlouběji definují.

1. Nejvyšší prioritou je vyhovět zákazníkovi včasným a průběžným dodáváním softwaru s přidanou hodnotou.
2. Požadavky na změnu jsou vítané, a to i v průběhu a pozdějších fázích vývoje. Agilní přístupy je využijí tak, aby zákazníkovi přinesly konkurenční výhodu.
3. Funkční software dodávat co nejčastěji, v řádu týdnů až měsíců. Preferovat kratší intervaly dodání.
4. Lidé ze strany obchodu musí spolupracovat s vývojáři na denní bázi během celého projektu.
5. Na projektech by měli pracovat motivovaní jedinci. Je potřeba poskytnout jim prostředí a podporu, kterou potřebují. Důvěřovat, že svou práci dokončí.

6. Nejúčinnější a nejefektivnější metoda sdílení informací ve vývojovém týmu je osobní komunikace z očí do očí.
7. Hlavním ukazatelem pokroku vývoje je funkční software.
8. Agilní přístupy prosazují a podporují udržitelný vývoj. Sponzoři, vývojáři a uživatelé by měli být schopni udržet vzájemně stejné tempo.
9. Stálá průběžná pozornost věnovaná technické dokonalosti a dobrému návrhu posiluje agilní možnosti.
10. Zásadní je jednoduchost. Největším uměním je maximalizovat množství neprováděné práce.
11. Nejlepší architektury, požadavky a návrhy vznikají v samostatně se řídících týmech.
12. Tým se pravidelně zamýšlí nad tím, jak být efektivnější a podle toho mění svůj směr a chování.

1.3 Lean – štíhlá produkce

V souvislosti s agilním řízením projektů je potřeba vysvětlit také pojem „lean“, se kterým se vzájemně prolíná. Lean neboli štíhlá produkce je využívána zejména v prostředí tovární výroby. Jedná se o přístup, kdy se dělají věci, které jsou zrovna potřeba. Výroba je řízena tak, že příslušný díl se vyrábí až když je potřeba. Jedná se o přístup se snahou o omezení vykonávané práce, která nepřináší přidanou hodnotu. Jak již bylo zmíněno, jedním z principů agilního myšlení je zaměření na to, co je v daný moment důležité. Agilní přístup se také snaží omezovat nadbytečnou práci, která by nemusela přinést žádnou hodnotu navíc. (Šochová, 2019)

Lean Software development je založen na principech:

- odstranění všeho, co nepřináší hodnotu – zbavení se „odpadu“, práce na něčem, co se ve finále vyhodí nebo nepoužije je škoda, čas strávený takovou prací se může využít na věcech co mají smysl, to může vést k vyšší efektivitě,
- zlepšování a zdokonalování v průběhu procesu – procesy je potřeba sledovat neustále, aby se předešlo opakovaným chybám,
- rozhodování co nejpozději – je faktem, že čím později dochází k rozhodování, tím více informací je již k dispozici,

- dodání práce, co nejrychleji – čím dříve je něco dokončeno, tím dříve může být získána zpětná vazba, kterou je možné využít při dalším vývoji,
- poskytnutí důvěry a odpovědnosti týmu – v případě poskytnutí důvěry a vlastní odpovědnosti týmu se tým může stát daleko více motivovaným a „hravým“,
- soustředění na celkový výsledek – při vývoji softwaru je potřeba zaměřit se na celkový dojem, který produkt může vyvolat, je potřeba dbát na kvalitu a celkovou udržitelnost systému. (Šochová, 2019)

1.4 Co může agilní přístup přinést?

Agilní přístup je iterativní proces, který je založen na týmové spolupráci, otevřené komunikaci, zapojení zákazníka do procesu vývoje a připravenosti a ochotě ke změně. (Bednárová, Bogdanovská, Mojžišová, Pócsová, 2020)

Agilní metody očekávají změny požadavků ze strany zákazníků a umožňují jejich zohlednění tak, aby nebylo ohroženo úspěšné dokončení projektu. Zapojením zákazníků do procesu vývoje umožňují průběžně nasazovat funkcionalitu a tím rychle odhalovat případnou potřebu změny a pružně na ní reagovat. (Šochová, 2019)

Se zaměřením na týmovou spolupráci, zákazníka a s rychlým šířením informací mohou přinést vyšší efektivitu. (Šochová, 2019)

Průběžné nasazování funkcionalit u zákazníků, větší otevřenost a lepší přehled mohou zlepšit kvalitu výsledného produktu. Svou podstatou poskytují větší otevřenost, „průhlednost“ a přehlednost, díky tomu je možné lépe předvídat a úspěšně a včas ukončovat projekty. (Šochová, 2019)

V neposlední řadě agilní metodiky svým zaměřením na zákazníka a jednotlivé členy týmu mohou výrazně ovlivnit i celkovou spokojenost všech zúčastněných. (Šochová, 2019)

1.5 Agilní transformace

Transformace způsobu řízení procesu vývoje softwaru na agilní řízení je náročné. Nejedná se pouze o formální změnu, kdy se něco napíše nebo poví a jede se dál ve starých kolejích. Jedná se o změnu myšlení a přístupu. Jde o změnu kultury v celé organizaci. Změna na agilní řízení procesů by měla být nikdy nekončící cestou s nacházením nových metod, jak zlepšit styl práce, jak najít nové inovativní řešení, jak se zdokonalit a podobně. Agilní přístup je jen cesta, jak dosáhnout cílů. (Šochová, 2019)

1.5.1 Důvody pro přechod na agilní řízení

Zavádění agilních metodik je velice složitý proces. Jedná se o celkovou změnu firemní kultury. Z hierarchicky řízené organizace by se měla stát organizace orientovaná na problém. Cílem je dosažení samoorganizovaných týmů. Práce nebude založena na jednotlivcích řízených „někým shora“, ale na spolupráci lidí v týmu, jejich vzájemné komunikaci, vlastním rozhodování a přijetí více vlastní odpovědnosti. (Šochová, 2019)

Nejčastějšími důvody pro přechod na agilní řízení jsou:

- potřeba pružnosti – svět a doba se mění, vše funguje rychleji, dochází k rychlejšímu přenosu informací a všichni potřebují reagovat pružněji, rychleji, téměř okamžitě, všichni chtějí najednou vše hned,
- zvýšení efektivity – je faktem, že spolupráce více lidí je výrazně efektivnější než práce jednotlivců, k dispozici jsou například metodiky párového programování nebo metodika Scrum postavená na spolupracujících a samořídících se týmech,
- přehlednost, transparentnost a předvídatelnost – agilní metody mohou pomoc k včasnému dokončování projektů, jsou postavené na odhadování náročnosti prací, do těchto odhadů je zapojený celý tým, postupem času se odhady díky zkušenostem týmu zpřesňují, agilní metody dávají týmu do rukou možnost projekt rozdělit do menších dílčích částí, a tak mají všichni lepší přehled na čem aktuálně pracují a co je hotové,
- zlepšení kvality – do procesu je zapojen zákazník, jsou známé jeho požadavky a očekávání, výsledky jsou mu prezentovány průběžně, celý tým je zodpovědný za kvalitu výsledku, snižuje se počet chyb a zlepšuje se udržitelnost kódu. (Šochová, 2019)

1.6 Vybrané agilní metodiky

Jako podpora a nástroje agilního způsobu řízení slouží takzvané metodiky neboli frameworky. Existuje několik různých metodik pro agilní přístup, například:

- Lean Software Development – „štíhlý vývoj“ se zaměřením na vývoj toho co je potřeba, snaží se omezit nadbytečnou práci, je spíše souhrnem pravidel než metodikou, jejichž používání by mělo zefektivnit a zrychlit vývojový proces,
- Scrum – aktuálně jeden z nejpoužívanějších a nejúspěšnějších frameworků, využívá se v komplexních prostředích, kde je složité plánování a produkt se postupně vyvíjí v iteracích, kde je potřeba pružně reagovat na změny, je zaměřen na tým a spolupráci, jedná se o jednoduchou metodiku na pochopení vyžadující zásadní změnu přístupu a myšlení,
- Kanban – Kanban má velice blízko ke Scrumu, je volnější a nevyžaduje tak zásadní změny, specifikují ho tři základní principy: omezení rozpracované práce, minimalizace stráveného času a vizualizace průběhu
- Extrémní programování (XP – Extreme Programming) – metodika vhodná pro „extrémní“ prostředí, kde je zapotřebí velice rychle reagovat, iterace jsou zde nastaveny na dny v některých případech na jeden jediný den, každý den tak může být ukončena iterace a do produkčního prostředí se nasazují nové funkcionality, s testováním se nečeká a testuje se již v průběhu, vývoj probíhá v rychlých iteracích, návrh, implementace, testování, nasazení, často je zákazník součástí vývojového týmu, dochází k častým změnám, může se tak stát, že několikrát za den se požadavek změní a práce se „zahazuje“,
- Vývoj řízený vlastnostmi (FDD – Feature Driven Development) – metodika podobná Scrumu s tím rozdílem, že práce je jednotlivým programátorům přidělována, iterace většinou trvají dva týdny, zákazník průběžně dostává verze, proces začíná vytvořením modelu popisující celý systém, ten se následně převede do seznamu vlastností, na kterých se postupně pracuje,
- Vývoj řízený testy (TDD – Test Driven Development) – metodika prosazující jako první návrh testů před samotným psaním kódu, implementuje se takový kód, který projde testem, programátor plní předem definované testy, vývoj je úspěšný, pokud výsledky projdou testy. (Šochová, 2019), (Myslín, 2016)

1.7 Agilní slovníček

Tabulka 1: Agilní slovníček – základní pojmy

Pojmy	**Vysvětlení**
Definice hotovo Definition of Done	Domluvená pravidla pro předání, specifikují podobu hotové funkcionality. Akceptační kritéria.
Extrémní programování Extreme Programming	Agilní metodika postavená na spolupráci a praktikách jako review, průběžná integrace (continuous integration), vývoj řízený testy (TDD – Test Driven Development), párové programování…
Kanban	Metodika spojující principy agilního řízení a "štíhlého vývoje" fungující na principech vizualizace průběhu procesu, minimalizace stráveného času a omezení rozpracované práce.
Plánování Planning	Při využívání Scrumu plánuje tým. Tým vybírá prioritní úkoly, které je schopen dokončit v následujícím Sprintu a zařazuje je do Sprint Backlogu.
Položka Backlogu	Úkol definuje funkcionalitu, která přináší hodnotu. Často je zapsán formou User story.
Product Backlog	Seznam úkolů / funkcionalit (user stories) setřídění podle priorit, které jsou plánované dodat zákazníkovi. O stanovení priorit se stará Product Owner. Na jeho vzniku se podílí celý tým a často i vedení a zákazníci. Je přístupný celému týmu.
Product Owner Vlastník produktu	Product . Na starosti má vizi, Product Backlog a je zodpovědný za celkovou úspěšnost produktu. Je součástí týmu. Navrhuje, co je potřeba v daném produktu či oblasti udělat. Na základě komunikace se zákazníky určuje prioritu úkolů. V podstatě se jedná o vedoucího týmu.
Retrospektiva	Týmová schůzka / ohlédnutí, kde tým hodnotí, co se mu při poslední iteraci dařilo, v čem chtějí pokračovat a co chtějí naopak změnit, vylepšit nebo přestat dělat.
Review meeting Sprint Review	Schůzka, kde se na konci každého Sprintu prezentují výsledky, kterých bylo dosaženo během Sprintu s cílem získání zpětné vazby.
Scrum	Jedna z nejúspěšnějších a nejpoužívanějších agilních metodik, proces založený na týmové spolupráci, zapojení zákazníka a získávání zpětné vazby v krátkých Sprintech (iteracích).
Scrum Master	Kouč a moderátor týmu. Stará se o rozvoj, fungování a potřeby týmu, odstraňuje překážky. Udržuje Scrum proces v chodu. Není vedoucí týmu.
Scrum Tabule	Zobrazuje stav aktuálního Sprintu.
Samoorganizovaný tým	Tým, který se sám organizuje a sám rozhoduje, na čem a jakým způsobem bude pracovat, bez vnějšího vlivu tak, aby maximalizoval efektivitu, pružnost a motivaci jednotlivých členů.
Sprint	Krátká iterace s pevně danou délkou trvání, ve které se tým zaváže dodat funkcionalitu, která má pro zákazníka hodnotu.
Sprint Backlog	Seznam úkolů z Product Backlogu utvářející výslednou funkcionalitu pro daný Sprint.
Cíl Sprintu Sprint Goal	Cíl Sprintu může být považován za vizi na jeden Sprint. Správný cíl shrnuje potřeby zákazníka a je zaměřený na hodnotu. Přináší Sprintu smysl a hodnotu.
Standup Scrum Meeting Daily Scrum	Pravidelná každodenní krátká týmová schůzka, obvykle ráno, kde jednotliví členové mezi sebou sdílí aktuální stav své práce a zmiňují případné problémy. Moderována Scrum Masterem.
Popis funkcionality User Story	Popsaná funkcionalita, kterou chce uživatel. Položka Backlogu, měla by být dostatečně stručná a srozumitelná, aby ji tým mohl naplánovat na další Sprint.
Zákazník	Zahrnuje všechny, kteří od produktu něco očekávají jak zevnitř, tak i z venku firmy, například manažeři, obchodní oddělení, podpora nebo výslední uživatelé.

Zdroj: vlastní tvorba na základě použitých knih od Šochové a Myslína

2. Scrum

Scrum je jednou z nejúspěšnějších a nejpoužívanějších metodik agilního řízení. Je zaměřen na tým a spolupráci. Byl vyvinut počátkem roku 1990. Za jeho vznikem stojí pánové Ken Schwaber a Jeff Sutherland.

První počátky agilního řízení prostřednictvím Scrumu tedy sahají téměř deset let před událost podpisu agilního manifestu, který mimochodem podepsali i pánové Schwaber a Sutherland.

Metodika je oficiálně popsána v příručce s názvem: „Průvodce Scrum, Oficiální průvodce pro Scrum: Pravidla hry“, v originále „The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game“. Tento oficiální průvodce byl poprvé publikován v roce 2010 a je stále aktualizován a dále rozvíjen. Dostupný je na adrese https://Scrumguides.org/Scrum-guide.html. Aktuální a poslední verze byla vydána v listopadu 2020. Za cíl si klade pomoci lidem po celém světě pochopit, co Scrum znamená. Obsahuje definici Scrumu.

Každý prvek tohoto frameworku slouží konkrétnímu účelu, který je nezbytný pro celkovou hodnotu a výsledky dosažené pomocí Scrumu. Nepochopení, změna základního principu nebo myšlenek Scrumu, vynechání jednotlivých prvků nebo nedodržování pravidel za účelem zakrytí problémů omezuje přínos Scrumu a může jej učinit zcela zbytečným. Scrum má kořeny v softwarovém vývoji. Za dobu svojí existence si však vybudoval pověst užitečného nástroje v mnoha odvětvích. Pracují s ním vývojáři, výzkumní pracovníci, analytici, vědci a spousta dalších. Označení, že je někdo vývojář ve Scrumu je už jen pouze symbolický odkaz na historii a slouží pouze pro zjednodušení a rozhodně nemá význam určování toho pro koho je Scrum vhodný. (Schwaber, Sutherland, 2020)

V roce 2009 Ken Schwaber založil Scrum.org světovou organizaci pro zvyšování kvalifikace těm, kteří s metodikou pracují. Scrum.org poskytuje komplexní školení, zdroje, hodnocení a certifikace, které mají lidem pomáhat s řešením jejich složitých problémů.

Dalším významným uskupením odborníků na poli Scrum je americká Scrum Aliance. Členkou, jehož představenstva je také nejzářnější česká hvězda tohoto frameworku Zuzana „Zuzi“ Šochová.

2.1 Definice

Metodika Scrum je soubor základních pravidel, pojmů a nástrojů, základní rámec, který má pomáhat lidem, týmům a organizacím budovat přidanou hodnotu při řešení komplexních a složitých problémů s využitím možností pružnosti. Definuje zásadní činnosti zaměřené především na komunikaci a spolupráci, které se následně neustále opakují. (Schwaber, Sutherland, 2020)

Princip metodiky Scrum je jednoduchý. Dle tvůrců je potřeba ho jen vyzkoušet a zjistit, jestli je účinný a jestli jeho filozofie, teorie a struktura může pomoci dosáhnout stanovených cílů a vytvořit tak přidanou hodnotu. Rámec Scrumu není záměrně příliš detailní, definuje pouze části, které jsou potřebné k jeho implementaci. Scrum je založen na spolupráci, komunikaci, inteligenci a vlastním přístupu lidí, kteří jej používají. Stojí na ideji, že je lepší lidem poskytnout volnost a podporu jejich vztahů než detailní pokyny. Při používání Scrumu můžou být využity různé procesy, techniky a metody. Scrum dokáže využít existující postupy nebo je identifikuje jako nadbytečné. Použití Scrumu přináší velkou míru transparentnosti. U využívaných technik řízení, prováděných prací a daném prostředí zviditelňuje jejich účinnost a užitečnost. (Schwaber, Sutherland, 2020)

V základu jde o to, že existuje samořídící se tým, jehož podporu a moderaci má na starosti člověk s rolí Scrum Master, kde:

1. Product Owner zadá práci na komplexním složitém problému a vytvoří Product Backlog,
2. tým během Sprintu převádí vybranou práci a úkoly na přidanou hodnotu,
3. na konci Sprintu tým a další zúčastněné osoby kontrolují výsledky a navrhují postup zejména pro zlepšení a zvýšení efektivity pro následující Sprint,
4. proces se opakuje. (Schwaber, Sutherland, 2020)

Cyklus a průběh procesu Scrumu je také možné rozdělit do tří etap:

1. Zahájení – ujasnění formy spolupráce, definice cíle, rozdělení rolí a kompetencí, seznámení týmu,

2. Vývoj – iterativní proces, kdy probíhá samotný vývoj softwaru. Iterace probíhají ve Sprintech,
3. Ukončení – výsledný produkt je finálně otestován, přijat a nasazen, projekt je oficiálně ukončen. (Myslín, 2016)

Po zahajovací fázi se vstupuje do fáze vývoje. V této fázi tým setrvává až do závěru projektu. Ve vývojové fázi probíhají jednotlivé Sprinty, které mají po formální stránce vždy stejný průběh. To znamená, že práce týmu se liší, ale její průběh probíhá ve stále stejně nastaveném prostředí. Podoba jednotlivých Sprintů tak zůstává v podstatě neměnná – zahájení, plánování, kontrola, měření a tak dále. Od metodiky v procesu očekáváme jistotu a stabilitu, obsah práce je vždy závislý na tom, co je chtěné. (Myslín, 2016)

2.2 Teorie Scrumu – základní pilíře

Scrum je založen na principech myšlení a rozhodování pramenícího ze zkušeností a „štíhlém“ myšlení, kdy se realizuje to, co je zrovna potřeba. Znalosti pramení ze zkušeností a rozhodování je prováděno na základě pozorování. Snižuje „plýtvání“ a zaměřuje se na to podstatné. (Schwaber, Sutherland, 2020)

Pro lepší předvídatelnost a pro možnost vyhodnocování rizik Scrum využívá postupný, iterativní přístup. Pro usnadnění práce zapojuje skupiny lidí, kteří mají jako celek všechny potřebné dovednosti, odborné znalosti a podle potřeby mohou tyto své znalosti sdílet anebo naopak získávat tak, aby dokázali danou práci vykonávat. (Schwaber, Sutherland, 2020)

Scrum využívá a kombinuje čtyři základní a formální události, díky kterým může být zajištěna kontrola a možnost pružného přizpůsobení během Sprintů. To vše může fungovat, protože na základě zkušenostní prosazují základní pilíře Scrumu:

- transparentnost,
- kontrola/vyhodnocení
- možnost přizpůsobení. (Schwaber, Sutherland, 2020)

2.2.1 Transparentnost/přehlednost

Vznikající a vyvíjející se proces a práce musejí být přehledné, jak pro ty, co jí vykonávají tak pro ty, kteří jí přijímají. Ve Scrumu jsou důležitá rozhodnutí založena na vnímaném stavu a úrovni základních třech „Scrum artefaktů“, respektive naplnění cílů. Artefakty, které mají špatnou průhlednost, mohou vést k rozhodnutím, která mohou snížit přidanou hodnotu a zvýšit riziko. (Schwaber, Sutherland, 2020)

Transparentnost umožňuje kontrolu. Kontrola bez přehlednosti může být zavádějící a zbytečná. (Schwaber, Sutherland, 2020)

2.2.2 Kontrola/ohlédnutí zpět

Scrum artefakty, cíle Scrumu a jejich pokrok a průběh směrem k dohodnutým cílům musejí být často a důkladně kontrolovány, aby byly potenciálně nežádoucí odchylky detekovány včas a předešlo se tak problémům. Metodika Scrum nabízí možnost konání pěti pravidelných událostí, které napomáhají zpětnému ohlédnutí a kontrole. (Schwaber, Sutherland, 2020)

Kontrola umožňuje přizpůsobení a změnu. Kontrola bez přizpůsobení a změny je považována za zbytečnou. Události Scrumu jsou navrženy tak, aby dokázaly identifikovat problém a vyvolat změnu. (Schwaber, Sutherland, 2020)

2.2.3 Přizpůsobení/změna

Pokud se některé vlastnosti nebo části procesu v průběhu odchylují od přijatelných hodnot nebo je výsledný produkt nepřijatelný, musí být využívaný proces nebo základní postupy upraveny. Z důvodu minimalizace negativních dopadů a dalších odchylek je potřeba změnu provést co nejdříve. (Schwaber, Sutherland, 2020)

Přizpůsobení se stává náročným, pokud lidé v týmu nejsou dostatečně kompetentní nebo se neumí sami organizovat. Očekává se, že se tým dokáže přizpůsobit v momentě, kdy se učí něco nového, například ze svých chyb při kontrolách a zpětném ohlédnutí. (Schwaber, Sutherland, 2020)

2.3 Hodnoty Scrumu

Úspěch použití Scrumu závisí na tom, jak lidé dokážou přijmout a dodržovat základních pět hodnot:

- smysl pro povinnost a závazek,
- pozornost,
- otevřenost,
- respekt,
- odvaha. (Schwaber, Sutherland, 2020)

Tým se ve Scrumu zavazuje k dosažení svých cílů a ke vzájemné podpoře. Jejich hlavním záměrem je soustředění a směřování pozornosti na vybranou práci Sprintu tak, aby dosáhl co největšího pokroku při plnění těchto cílů. Při práci na projektu je tým a všichni zúčastnění otevřen změnám a výzvám. Potvrzením a ukázkou schopností a nezávislosti jednotlivců je vzájemný respekt členů týmu. Samozřejmostí je také poskytovaná úcta i vůči všem ostatním zúčastněným. Členové týmu mají odvahu činit správná rozhodnutí, správnou věc a pracovat na náročných úkolech. (Schwaber, Sutherland, 2020)

Tyto hodnoty udávají směr týmu s ohledem na jejich práci, jednání a chování. Rozhodnutí, která jsou činěna, přijatá opatření, další kroky a způsob a cesta, kterou Scrum používá by měla v samotných členech tyto hodnoty ještě více posílit, nikoliv je ponížit nebo podkopat. Členové týmu se učí a objevují tyto hodnoty průběžně s výkonem jejich práce, prováděním Scrum událostí a naplňováním cílů. Naplňování těchto hodnot a prosazování tří základních pilířů týmem a ostatními zúčastněnými posiluje a buduje důvěru. (Schwaber, Sutherland, 2020)

2.4 Scrum Tým

Základní jednotkou Scrumu jsou malé týmy lidí neboli Scrum týmy. Součástí Scrum týmu je jeden Scrum Master, jeden Product Owner (vlastník produktu) a vývojáři. Ve Scrum týmu neexistují žádné dílčí pod-týmy ani není definována žádná hierarchie. Jedná se o soudržnou

jednotku profesionálů zaměřenou v danou dobu na jediný cíl, takzvaný produktový cíl. (Schwaber, Sutherland, 2020)

Scrum Týmy jsou multifunkční, to znamená, že jednotliví členové jsou dostatečně kompetentní a mají všechny potřebné dovednosti pro vytvoření hodnoty každého Sprintu. Tým je také samostatně se řídící jednotkou, což znamená, že členové interně rozhodují o tom, kdo, co, kdy a jak dělá. (Schwaber, Sutherland, 2020)

Důležitou vlastností týmu a jeho členů je vzájemná zastupitelnost a multifunkčnost. To, že je v týmu expert na konkrétní problematiku neznamená, že se jí musí věnovat ze sta procent času své práce. Tým se vždy sám dohaduje o tom, kdo na čem bude pracovat nebo kdo komu pomůže. Lidé si sami určují, na čem chtějí pracovat a v čem se chtějí rozvíjet. Je to dlouhodobý proces a investice do celkové pružnosti týmu. (Šochová, 2019)

Nutnou podmínkou pro vnik samostatně se řídícího týmu je společný cíl. Tým musí rozumět zákazníkovým potřebám, znát a chápat jeho prostředí, vědět, jakým způsobem bude produkt využívat. Všichni musejí jít za stejným cílem, mít stejnou vizi. Důležitým základním kamenem týmu je také důvěra. A to nejen mezi jeho členy, ale i vůči všem ostatním zúčastněným. V neposlední řadě je potřeba týmu poskytnout možnost samostatně se rozhodovat a měnit způsob práce. Umožněním týmu podílet se na tvorbě Product Backlogu a dovolit vlastní přínos může přispět k tomu, že zákazník dostane to nejlepší možné. Tým musí táhnout za jeden provaz. Pokud selže jeden člen týmu, nesmí to být považováno za jeho selhání, ale znamená to, že selhal celý tým. (Šochová, 2019)

Tím, že všichni pracují na stejné věci nemusí docházek k neustálému přepínání kontextu, kdy se vývojář ptá, jak to analytik myslel, nebo k pocitům, že to tester pořád rozbíjí, když už to bylo tolikrát opravené. (Šochová, 2019)

Tým by měl být sestaven na plný úvazek. Pro co nejjednodušší a nerychlejší sdílení znalostí, zkušeností, pomoc a spolupráci by členové týmu měli sedět v jedné místnosti. (Šochová, 2019)

Tým musí být dostatečně malý, aby si zachoval svou hbitost a pohotovost a být dostatečně velký, aby dokázal dokončit práci v rámci Sprintu. Obvykle se skládá z počtu deseti a méně lidí. Obecně lze tvrdit, že uvnitř menších týmů probíhá komunikace snáze a jsou efektivnější.

Pokud se Scrum tým stane příliš velkým, mělo by se zvážit přeskupení nebo rozdělení do několika menších soudržných týmů, každý se zaměřením na stejný produkt. Měly by sdílet stejný produktový cíl, Product Backlog a měly by mít stejného Product Ownera. (Schwaber, Sutherland, 2020)

Odpovědností týmu jsou všechny činnosti související s produktem, od spolupráce a komunikace se všemi zúčastněnými, přes ověřování, údržbu, provoz, experimentování, výzkum, vývoj a vše ostatní, co by mohlo být požadováno. Scrum týmy jsou strukturovány a zmocněny organizací k řízení vlastní práce. Práce ve Sprintech udržitelným tempem zlepšuje zaměření a pevnost Scrum týmu. (Schwaber, Sutherland, 2020)

Pro každý Sprint je celý tým zodpovědný za vytváření užitečného přírůstku, který má hodnotu. Scrum definuje tři základní specifické odpovědnostní role ve Scrum týmu:

- Vývojáři,
- Product Owner neboli vlastník produktu,
- Scrum Master. (Schwaber, Sutherland, 2020)

2.4.1 Vývojáři

Vývojáři jsou členové Scrum týmu, kteří se zavázali vytvářet jakoukoliv použitelnou část přírůstku Sprintu. (Schwaber, Sutherland, 2020)

Konkrétní dovednosti, které vývojáři potřebují, jsou často široké a liší se v závislosti na oblasti práce. (Schwaber, Sutherland, 2020)

Vývojáři jsou však vždy odpovědní za:

- sestavení plánu pro Sprint, takzvaného Sprint Backlogu,
- přínos kvality dodržováním dohodnutého plánu a pravidel předání produktu, takzvaného Definiton of Done, definice provedeného a cíle,
- každodenní přizpůsobení vlastního plánu se zaměřením na cíl Sprintu,
- vzájemnou podporu a profesionalitu týmu. (Schwaber, Sutherland, 2020)

Vývojář, označován také jako Developer nebo Scrum Team Member je „řadovým“ členem týmu. Jedná se o pracovníka s velkou volností v plnění vlastních úkolů. Nikdo mu neřídí, jak má své úkoly plnit s ohledem na dané technologie a technické parametry. Tento řadový člen je v podstatě nejdůležitějším členem týmu, protože to je on, kdo je zodpovědný za probíhající vývoj. Často platí, že člen týmu je vším a nerozlišují se role jako architekt, analytik, tester nebo kodér. Má možnost ale i povinnost plně ovlivnit výslednou podobu a fungování vyvíjeného produktu. (Myslín, 2016)

2.4.2 Product Owner – vlastník produktu

Vlastník produktu je odpovědný za maximalizaci hodnoty produktu vyplývající z práce týmu. Způsob, jakým se to provádí, se může v různých organizacích, týmech, ale i u jednotlivců lišit. (Schwaber, Sutherland, 2020)

Vlastník produktu také odpovídá za správu Product Backlogu, která zahrnuje:

- rozvoj a výslovnou komunikaci produktového cíle,
- vytváření a zřetelnou a jasnou komunikaci položek Product Backlogu,
- zadávání položek/požadavků Product Backlogu,
- zajištění dostupnosti, transparentnosti, viditelnosti a srozumitelnosti Product Backlogu. (Schwaber, Sutherland, 2020)

Vlastník produktu může výše uvedenou práci delegovat na další členy týmu. Přesto mu, ale zůstává plná odpovědnost za tyto úkony. (Schwaber, Sutherland, 2020)

Má na starosti definování produktové vize a její srozumitelnou komunikaci týmu, zákazníkům a organizaci. Definuje priority, rozhoduje, na které funkcionalitě se bude pracovat dříve, na které později a na které vůbec. Je zodpovědný za Product Backlog. Pomáhá mu celý tým a všichni zúčastnění. Měl by trávit čas se zákazníky, vstřebávat jejich prostředí a potřeby, aby dokázal vždy správně rozhodnout, kde je pravá hodnota pro zákazníka. Posláním vlastníka produktu není ani tak řízení pracovníků, ale spíše stanovení priorit a toho, co se má dělat a v jakém pořadí. Hlavním cílem vlastníka produktu je úspěšný produkt. Role vlastníka produktu by neměl být kombinována s rolí Scrum Mastera. Měl by jím být komunikativní člověk se silnými znalostmi produktu. (Šochová, 2019)

Pro úspěch vlastníka produktu je důležité, aby organizace respektovala a důvěřovala jeho rozhodnutím. Možné dopady těchto rozhodnutí jsou čitelné z obsahu a zadání požadavků Product Backlogu a následně prostřednictvím kontrol přírůstků plynoucích ze Sprintu vyhodnocovaných na události s názvem Sprint Review. (Schwaber, Sutherland, 2020)

Vlastníkem produktu je jedna osoba. Vlastník produktu zastupuje potřeby většiny zúčastněných stran prostřednictvím Product Backlogu. Pokud někdo chce ovlivnit podobu Product Backlogu musí přesvědčit vlastníka produktu. (Schwaber, Sutherland, 2020)

2.4.3 Scrum Master

Scrum Master je koučem, moderátorem a takzvaným servant leaderem. Jeho hlavním cílem je vytvoření samostatného, efektivního a spokojeného samořídícího se týmu. Jeho cíle a povinnosti by se daly shrnout do následujících bodů:

- pomáhá týmu, aby dobře fungoval, stal se vysoce výkonným a dosahoval tak lépe svých cílů,
- dobrým moderováním pomáhá týmu odstraňovat překážky,
- motivuje tým k lepším výsledkům
- koučuje a vede tým, stará se o jeho rozvoj. (Šochová, 2019)

Stará se o to, aby byl proces Scrumu efektivní a fungoval. Má na starosti jeho dodržování, ale zároveň má možnost iniciovat změnu, pokud je to potřeba. Scrum Master by měl podporovat rozvoj týmu a jednotlivců, být komunikativní, vnímavý a řešit a utlumovat případné konflikty v týmu. Mělo by se jednat o člověka, který dokáže poradit a obvykle zná odpovědi na otázky lépe než ostatní. Musí být vnímavý, klást otázky a podporovat samostatnost týmu, který si na řešení problému musí přijít sám i za cenu dočasné neefektivity. Je to ten, kdo týmu „zametá cestičku“, aby se mu dobře šlo a pracovalo. Scrum Master je důležitou součástí týmu a měl by mu být neustále plně k dispozici. Proto by měl s týmem sedět v jedné místnosti. Scrum master by měl být empatický, komunikativní, být dobrý moderátor, kouč a v neposlední řadě agilní nadšenec. (Šochová, 2019)

Scrum Master je odpovědný za zavedení Scrumu tak, jak je definován v oficiálním Průvodci Scrumu. Měl by pomoci každému členovi týmu s pochopením teorie a praxe principů metodiky Scrum a to, jak v rámci týmu, tak v rámci celé organizace. (Schwaber, Sutherland, 2020)

Odpovídá za účinnost a efektivitu týmu prostřednictvím poskytnutí metod a nástrojů pro zlepšení postupů v rámci Scrumu. (Schwaber, Sutherland, 2020)

Scrum Master je skutečný vůdce a pomáhá, dalo by se říct slouží, týmu a větší organizaci. (Schwaber, Sutherland, 2020)

Scrum Master pomáhá týmu vícero způsoby, například:

- koučováním a vedením členů týmu v samostatném řízení a zastupitelnosti,
- pomoci týmu soustředit se na vytvoření vysoké hodnoty přírůstku, který splňují definici cíle,
- odstraňováním překážek pokroku v postupu týmu,
- zajištěním správného průběhu Scrum událostí, aby se odehrály všechny, byly pozitivní, produktivní a byly udrženy v rozumném časovém rámci. (Schwaber, Sutherland, 2020)

Vlastníkovi produktu pomáhá například s:

- nalezením správných technik, metod a nástrojů pro efektivní definici cílů produktu a Product Backlogu,
- pochopením týmové potřeby jasných a stručných položek Product Backlogu,
- vybudováním a zavedením produktového plánování na základě zkušeností ve složitém prostředí,
- pokud si to situace žádá nebo je potřeba, usnadněním a moderováním spolupráce se zúčastněnými stranami. (Schwaber, Sutherland, 2020)

Scrum Master může organizaci pomáhat několika způsoby, například:

- vedením, školením a koučováním organizace při přijetí Scrumu,
- plánováním a radami se zaváděním Scrumu,

- pomoc zaměstnancům a zúčastněným stranám s pochopením a využíváním znalostního přístupu ve složitém prostředí,
- odstraňováním překážek mezi týmem a zúčastněnými stranami. (Schwaber, Sutherland, 2020)

2.5 Události Scrumu

Sprint obaluje všechny události Scrumu. Každá událost ve Scrumu je formální možností pro kontrolu a přizpůsobení a změnu artefaktů a cílů. Tyto události jsou navrženy tak, aby umožňovaly a poskytovaly požadovanou transparentnost. Neprovádění předepsaných událostí vede ke ztrátě příležitosti kontroly a přizpůsobení. Události používané ve Scrumu vytvářejí pravidelný režim a předcházejí tak potřebě schůzek a meetingů, které nejsou součástí Scrumu. (Schwaber, Sutherland, 2020)

Pro snížení složitosti je vhodné, aby se všechny události konaly ve stejnou dobu a na stejném místě. (Schwaber, Sutherland, 2020)

2.5.1 Sprint

Sprinty jsou srdcem a motorem Scrumu. Místem, kde se nápady proměňují ve skutečnou hodnotu. (Schwaber, Sutherland, 2020)

Sprint je iterace, jedná se tedy o jeden z mnoha opakujících se cyklů při vývoji softwaru. Vychází z pozorování, že pravidelně se opakující věci jsou pro lidi z psychologického hlediska příjemnější a snáze si na ně zvyknou. Scrum rozděluje celý vývojový proces do pravidelných cyklů, Sprintů, ve kterých tým dodává hotové funkcionality. Díky pravidelnosti těchto cyklů jsou výsledky prezentovány vždy včas a odpovídají zadání. (Myslín, 2016), (Šochová, 2019)

Jedná se o události s pevně stanovenou dobou trvání, měsíc a méně, pro vytvoření dojmu a pocitu soudržnosti a konzistence. Nový Sprint začíná vždy okamžitě po uzavření předchozího. (Schwaber, Sutherland, 2020)

Obecně platí, že čím kratší je Sprint, tím lépe. Čím kratší je Sprint, tím je i rychlejší zpětná vazba a přizpůsobení na změnu. (Šochová, 2019)

Sprint zaštiťuje veškerou práci, která je nezbytná pro dosažení produktového cíle, od plánování Sprintu, denních meetingů, Sprint Review po retrospektivu. (Schwaber, Sutherland, 2020)

Během Sprintu:

- nedochází ke změnám, které by mohly ohrozit cíl Sprintu,
- nedochází ke snížení kvality,
- je v případě potřeby upravován Product Backlog,
- jakmile je k dispozici více informací mohou se objasňovat a znovu vyjednávat podmínky projektu s vlastníkem produktu. (Schwaber, Sutherland, 2020)

Sprinty přinášejí předvídatelnost díky poskytovaným nástrojům kontroly, možností přizpůsobení a změny v průběhu projektu směrem k produktovému cíli minimálně jednou měsíčně. Pokud je doba trvání Sprintu příliš dlouhá, mohou se procesy stát příliš složitými, může dojít ke zvýšení rizika a v krajních případech až k znepřehlednění cíle, kdy už se ani neví na čem se pracuje. Kratší Sprinty lze použít pro vytvoření více užitečných cyklů pro poučení a omezení rizik vyšších nákladů. Každý Sprint může být považován za krátký projekt. (Schwaber, Sutherland, 2020)

Pro předvídání průběhu projektu existuje spousta nástrojů, jako jsou například grafy „vypálení“ (burn-downs), „vyhoření“ (burn-ups) nebo kumulativní toky. I když jsou považovány za užitečné, přesto nedokážou nahradit význam myšlení na základě zkušeností. Ve složitých prostředích není nikdy možné přesně předpovědět co se stane. Pro budoucí rozhodování lze využít pouze znalostí a zkušeností z přechozích událostí.

Sprint může být zrušen v případě zastarání cíle Sprintu. Pouze vlastník produktu může Sprint zrušit a ukončit.

2.5.2 Sprint Planning – Plánování Sprintu

Plánování Sprintu je v podstatě jeho zahájením. Během této události se stanovuje práce, která má být během Sprintu dokončena. Na plánování se podílí celý tým společně. (Schwaber, Sutherland, 2020)

Vlastník produktu je odpovědný za připravenost účastníků k diskusi o nejdůležitějších položkách Product Backlogu a za to, jak rozumí a jak se připravují na produktový cíl. Tým si může také dovolit pozvat další lidi, aby se zúčastnili plánování a poskytli rady. (Schwaber, Sutherland, 2020)

Plánování Sprintu se věnuje následujícím tématům:

- Proč je Sprint cenný?
  Vlastník produktu navrhuje jak a čím by mohl produkt nabýt na hodnotě a užitečnosti během aktuálního Sprintu. Tým následně spolupracuje na definici cíle Sprintu, který by vyjádřil proč je Sprint užitečný i pro ostatní zúčastněné. Cíl Sprintu musí být definován během plánování.
- Co je možné tento Sprint zrealizovat?
  Skrze diskusi s vlastníkem produktu vývojáři vybírají položky Product Backlogu pro realizaci během nadcházejícího plánovaného Sprintu. Během tohoto procesu může tým položky vylepšit a zlepšit tak jejich srozumitelnost.
  Predikce množství práce, kterou je možné během Sprintu stihnout dokončit je náročná. Nicméně s rostoucími zkušenostmi týmu se zlepšuje i jeho schopnost předpovědi.
- Jak bude vybraná práce provedena?
  Pro každou vybranou položku Product Backlogu vývojáři naplánují práci nezbytnou pro vytvoření přírůstku, který splňuje definovaný cíl. Často se to provádí rozložením položky ještě do menších dílčích položek, které je možné zpracovat během jednoho pracovního dne. Realizace je plně v rukou vývojářů. Nikdo jim neřekne, jak mají položky Product Backlogu převést na přírůstek hodnoty.
  Cíl Sprintu, vybrané položky Product Backlogu pro nadcházející Sprint a plán jejich dodání jsou souhrnně označeny, jako Sprint Backlog. (Schwaber, Sutherland, 2020)

Plánování začíná stanovením cíle Sprintu. Tým následně vybírá položky z Product Backlogu, které mu nejlépe pomůžou dosáhnout stanoveného cíle. Při výběru zvažuje nejen přírůstkovou hodnotu vzhledem ke stanovenému cíli, ale i své kapacity. Vybrané položky se stávají Sprint Backlogem. Scrum Master schůzku moderuje tak, aby si tým uvědomoval, kolik úkolů se již zavázal dokončit a jestli je takový plán realistický. Zároveň motivuje a pomáhá týmu ujasnit si cíl Sprintu s vlastníkem produktu. Součástí plánování může být také rozdělení vybraných položek na úkoly pro jeden pracovní den. (Šochová, 2019)

2.5.3 Daily Scrum / Standup / Scrum meeting

Jednou z nejznámějších praktik agilního řízení je Daily Scrum meeting. Říká se mu také Stanudup nebo denní Scrum. Je snadný na implementaci, a tak s ním většina týmů začíná. Teorie říká, že tým se pravidelně schází každý den, obvykle ráno, na jiném než pracovním místě. Členové nad seznamem úkolů sdílejí informace o tom, na čem pracovali včera, na čem budou pracovat dnes a zda nemají se svým úkolem nějaké problémy, o kterých by tým měl vědět. Cílem je také posílení vnímání týmového závazku. (Šochová, 2019)

Účelem Daily Scrum schůzek je kontrola postupu směrem k naplnění cíle Sprintu a podle potřeby případně úprava Sprint Backlogu pro následující plánovanou práci. (Schwaber, Sutherland, 2020)

Standupy jsou přibližně patnáctiminutové meetingy určené vývojářům týmu. Pro udržení režimu a snížení složitosti se koná každý pracovní den ve stejný čas, pokud možno na stejném místě. Pokud se vlastník produktu nebo Scrum Master aktivně podílejí na vývoji, účastní se ho i tito jako „řadoví“ vývojáři. (Schwaber, Sutherland, 2020)

Vývojáři mohou schůzce nastavit libovolnou podobu, pokud je schůzka přínosná, má smysl a je zaměřená na postup směrem k cíli Sprintu a specifikuje plán pro následující pracovní den. Vytváří pocit soustředění a zlepšuje samo organizaci. (Schwaber, Sutherland, 2020)

Tyto denní schůzky zlepšují komunikaci, dokáží včas identifikovat překážky, podporují rychlé rozhodování a eliminují tak následnou potřebu dalších schůzek. (Schwaber, Sutherland, 2020)

Denní Scrum, ale není jediným okamžikem, kdy může tým upravovat plán. Tím, že je Scrum zaměřen na spolupráci tak se tým často setkává po celý den a může tak téměř nepřetržitě diskutovat o přizpůsobení nebo přeplánování zbytku práce Sprintu. (Schwaber, Sutherland, 2020)

2.5.4 Sprint Review

Účelem Sprint Review je kontrola a prezentace dosažených výsledků za Sprint a určení dalších očekávaných změn. Tým prezentuje výsledky své práce klíčovým zúčastněným stranám a diskutuje se o dalším průběhu směrem k produktovému cíli. (Schwaber, Sutherland, 2020)

Během události tým a ostatní zúčastněné strany kontrolují čeho bylo během Sprintu dosaženo a co se změnilo. Na základě těchto informací spolupracují na návrhu dalšího postupu. Product Backlog je možné následně upravit tak aby odpovídal zjištěným novým skutečnostem. Tým by se zde neměl omezovat pouze na prezentaci svých výsledků, ale měla by to pro něj být příležitost ke sběru zpětné vazby. (Schwaber, Sutherland, 2020)

Sprint Review je předposlední událostí před koncem Sprintu. (Schwaber, Sutherland, 2020)

2.5.5 Retrospektiva Sprintu

Jednou z agilních praktik, která je přínosná i v neagilních prostředích a týmech, je retrospektiva, zpětné ohlédnutí. Je to efektivní nástroj pro jednoduché získávání zpětné vazby. (Šochová, 2019)

Retrospektiva se běžně skládá z několika fází:

- úvod,
- sběr dat – co funguje, co ne a co by šlo zlepšit,
- hlubší porozumění informacím – identifikace příčin, snaha o pochopení podstaty identifikovaného problému nebo prostoru pro zlepšení,
- návrh možností – bez stanovení konkrétních kroků by se problémy mohly opakovat stále dokola,

- souhrn konkrétních akcí – seznam konkrétních kroků vedoucích ke změně nebo zlepšení. (Šochová, 2019)

Účelem retrospektivy je naplánování dalších kroků a cesty, jak zvýšit kvalitu a efektivitu. (Schwaber, Sutherland, 2020)

Tým kontroluje a analyzuje průběh posledního Sprintu s ohledem na jednotlivce, interakce, procesy, nástroje a cíle. Kontrolované prvky se často liší na základě jejich oblasti. Identifikují se neúspěchy a zkoumá se jejich původ. Tým diskutuje o tom, co šlo a fungovalo dobře, na jaké problémy narazil a jak byly nebo nebyly tyto problémy vyřešeny. (Schwaber, Sutherland, 2020)

Tým identifikuje nejužitečnější změny, které vedly ke zvýšení jeho efektivity. Nejvýznamnější zlepšení budou řešena co nejdříve. Často bývají přidány do Sprint Backlogu pro další Sprint. (Schwaber, Sutherland, 2020)

Retrospektiva uzavírá Sprint. (Schwaber, Sutherland, 2020)

2.6 Artefakty Scrumu

Artefakty Scrumu reprezentují práci nebo její hodnotu. Jsou navrženy tak, aby maximalizovaly přehlednost a transparentnost klíčových informací. Každý, kdo se jimi zabývá, má tedy stejný základ pro přizpůsobení. (Schwaber, Sutherland, 2020)

Každý artefakt nese závazek pro zajištění takových informací, které přinesou a zvýší přehlednost a transparentnost:

- pro Product Backlog je to Product Goal neboli produktový cíl,
- pro Sprint Backlog je to Sprint Goal neboli cíl Sprintu,
- pro Inkrement neboli přírůstek je to Definition of Done neboli definice hotovo. (Schwaber, Sutherland, 2020)

Tyto závazky existují pro posílení myšlení na základě získaných zkušeností a hodnot Scrumu pro tým a všechny zúčastněné. (Schwaber, Sutherland, 2020)

2.6.1 Product Backlog

Product Backlog je seznam všech plánovaných úkolů. Tento seznam může být vytvářen vývojovým týmem, manažery nebo přímo zákazníkem. Na starosti ho má vlastník produktu, ale měl by být přístupný všem. Jednotlivé položky by měly odpovídat požadovaným funkcionalitám produktu, tedy tomu, co přináší zákazníkovi hodnotu. S udržováním Product Backlogu vlastníkovi produktu pomáhá tým a ostatní zúčastnění. Vlastník produktu je zodpovědný za nastavení priorit a smysluplnost požadavků od zákazníka, v Product Backlogu má poslední slovo. Prodcut Backlog obvykle obsahuje i odhad náročnosti v relativních jednotkách ohodnocený týmem a prioritu. (Šochová, 2019)

Product Backlog je seřazený seznam úkolů s přiřazenými prioritami, které jsou nutné pro zlepšení produktu. Hlavní a jediný celkový seznam a zdroj práce týmu. (Schwaber, Sutherland, 2020)

Položky Product Backlogu, které může tým zpracovat v rámci jednoho Sprintu jsou považovány za připravené k výběru při plánování Sprintu. Obvykle se stávají transparentními po přesnějším definování činností. Upřesnění Product Backlogu je akt rozdělení a další definice položek Product Backlogu na menší přesněji a srozumitelněji definované položky. Jedná se proces, kdy se specifikují podrobnosti, jako je popis, zadání a velikost. Vlastnosti se často liší podle oblasti práce. (Schwaber, Sutherland, 2020)

Za celkovou velikost, strukturu a roztřídění tohoto seznamu jsou odpovědní vývojáři, kteří budou práci vykonávat. Vlastník produktu jim pomáhá porozumět a vybrat a stanovit kompromisy. (Schwaber, Sutherland, 2020)

- Závazek: Produktový cíl
  Produktový cíl popisuje očekávaný budoucí stav produktu, který může sloužit týmu jako cíl vůči kterému se bude plánovat. Produktový cíl je součástí Product Backlogu. Ostatní položky Product Backlogu definují, co a jaké kroky povedou ke splnění produktového cíle. (Schwaber, Sutherland, 2020)

- Produkt je prostředek, který přináší hodnotu. Má jasné vymezení hranic, známé zúčastněné strany, správně definované uživatele nebo zákazníky. Za produkt je možné považovat službu, fyzický produkt, ale i něco více abstraktního. (Schwaber, Sutherland, 2020)

Produktový cíl je dlouhodobým cílem týmu. (Schwaber, Sutherland, 2020)

2.6.2 Sprint Backlog

Sprint Backlog je výňatkem, užší vybranou částí Product Backlogu. Obsahuje prioritní funkcionality, které tým plánuje dodat během aktuálního Sprintu. Sprint Backlog vytváří tým na začátku Sprintu podle priorit vlastníka produktu tak, aby mohl dosáhnout cíle Sprintu. (Šochová, 2019)

Sprint Backlog definuje „co?“, „jak?“ a „proč?“ bude realizováno během jednoho Sprintu. Skládá se z cíle Sprintu (proč?), vybraných položek z Product Backlogu (co?) a plánu, popisu, „cesty“ pro dodání přírůstku (jak?). (Schwaber, Sutherland, 2020)

Sprint Backlog je zúžený seznam úkolů pro aktuální Sprint, který navrhují vývojáři sami pro sebe. Jedná se o velice zřetelně viditelný obraz práce v reálném čase, kterou se vývojáři zavázali dodat během Sprintu s ohledem na dosažení cíle Sprintu. Sprint Backlog je průběžně během Sprintu v souvislosti s přísunem nových informací stále aktualizován. Měl by poskytovat dostatek informací, aby tým mohl průběžně kontrolovat pokrok na denních Scrumech. (Schwaber, Sutherland, 2020)

- Závazek: Cíl Sprintu
  Cíl Sprintu je závazkem vývojářů. Pokud jde o přesnou práci potřebnou k jeho dosažení, poskytuje velikou míru pružnosti. Cíl Sprintu v týmu také vytváří pocit soudržnosti a soustředění, povzbuzuje tým ke spolupráci. (Schwaber, Sutherland, 2020)

Cíl Sprintu je definován během události plánování a následně přidán do Sprint Backlogu. (Schwaber, Sutherland, 2020)

2.6.3 Inkrement – přírůstek

Inkrement neboli přírůstek je odrazovým můstkem pro dosažení produktového cíle. Každý přírůstek je aditivní ke všem předchozím, je důkladně ověřen, což zajišťuje, že všechny přírůstky spolu dokáží fungovat. Za účelem poskytnutí hodnoty musí být přírůstek použitelný. (Schwaber, Sutherland, 2020)

Během Sprintu může být vytvořeno více přírůstků. Souhrn přírůstků je prezentován na události Sprint Review a podporuje myšlení a další rozhodování na základě zkušeností. Přírůstek může být zúčastněným stranám dodáván ještě před koncem Sprintu. (Schwaber, Sutherland, 2020)

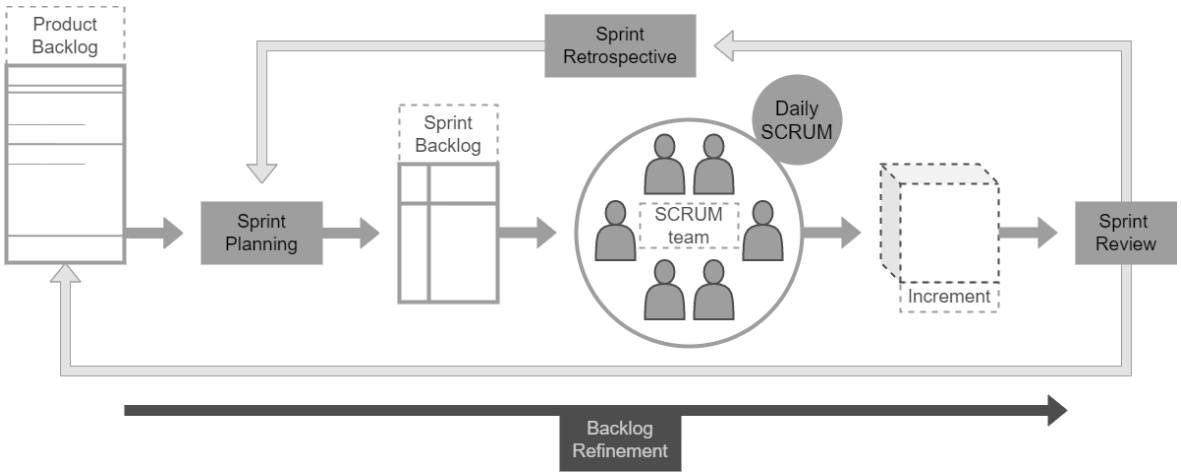
Práci nelze považovat za přírůstek, pokud nesplňuje „definici hotovo“. (Schwaber, Sutherland, 2020)

- Závazek: Definice hotovo

  Definice hotovo je formální zápis stavu přírůstku, který splňuje kritéria kvality produktu. V momentu, kdy položka Product Backlogu splňuje definice hotovo zrodí se přírůstek. Definice hotovo vytváří transparentnost tím, že poskytuje každému sdílené pochopení toho, jaká práce byla dokončena jako část přírůstku. Pokud položka Product Backlogu nesplňuje definici hotovo, nemůže být uvolněna ani prezentována na události Sprint Review. Místo toho se vrátí do Product Backlogu k možnému budoucímu zvážení. Pokud je definice hotovo součástí standardů organizace, musí ji dodržovat všechny týmy. Pokud se nejedná o standard, musí tým vytvořit vlastní definici hotovo odpovídající produktu. Vývojáři jsou povinni držet se definice hotovo. Pokud na produktu spolupracuje více týmů, musejí vzájemně definovat a dodržovat stejnou definici hotovo. (Schwaber, Sutherland, 2020)

2.7 Scrum schéma

Scrum začíná událostí – plánování sprintu (Sprint Planning), kde se na základě kompletního seznamu plánovaných úkolů (Product Backlog) sestavuje seznam požadavků pro nadcházející Sprint (Sprint Backlog). Každý den Sprintu probíhá krátká denní porada (Daily Scrum), na které se analyzuje průběžné dění. Výsledkem Sprintu vzniká posun ve vývoji (Increment), který se prezentuje na finální prezentační události (Sprint Review). Na závěr Sprintu tým vyhodnocuje svou dosavadní práci na události zpětného ohlédnutí (Sprint Retrospektive). Během průběhu celého Sprintu tým pracuje také na ohodnocení náročnosti jednotlivých požadavků (Backlog Refinement). Následně se celý proces opakuje v podobě dalšího Sprintu se zohledněním získaných zkušeností. Na obrázku číslo dvě je graficky znázorněno schéma průběhu Sprintu s využitím metodiky Scrum a jejích nástrojů.



Obrázek 2: Scrum schéma
Zdroj: vlastní tvorba na základě schéma od.Scrum.org

3. Informační systém Abra Gen a jeho moduly

Tato kapitola je věnována stručnému představení společnosti Abra Software a.s. a jejího největšího produktu, komplexního informačního systému Abra Gen.

3.1 Abra software a.s.

Společnost Abra Software, do roku 2006 známá jako AKTIS vznikla v roce 1991 jako sdružení pánů Petra Vacka a Jaroslava Řasy. (abra.eu)

Jejich cílem byla produkce softwaru pro podnikatele. Prvním takovýmto produktem byl informační systém Abra Plus. Po jeho rychlém rozšíření přišel populární Abra Classic. V roce 1994 se přidal produkt Abra Gold a o tři roky později Abra Gold mini. Od roku 2000 se na trhu pohybuje produktová řada Abra Gx. (abra.eu)

Od svého založení společnost zažila několik organizačních a právních změn. V roce 1994 Abra založila svou slovenskou pobočku. Rok 1997 znamenal pro Abru přeměnu v akciovou společnost. V roce 2008 byla založena společnost United Software a.s., která se stala mateřskou organizací Abry Software. United Software zastává funkci platformy pro spolupráci softwarových společností. Tato platforma je mířena převážně na české a slovenské firmy. Díky této skupině Abra získává obchodní podíly ve společnostech Flores, Neotech, slovenské Abra Software s.r.o. a zakládá společnost Systematiq. (abra.eu)

Mezi hlavní aktivity společnosti Abra Software a.s. se řadí vývojářská a dodavatelská činnost moderních informačních systémů. Dále poskytuje možnosti cloudového účetnictví, řeší e-commerce oblast, webové a mobilní zakázkové aplikace. Zabývá se také konzultantskými službami a návrhem optimálních hardwarových řešení. Zaměřena je především na střední a velké firmy. Vzhledem ke své dlouholeté existenci má spoustu zkušeností v oboru. (abra.eu)

Společnost v současné době sídlí v Praze. Za svou dlouholetou působnost se ale také rozrostla o několik poboček. V české republice má pobočky ve městech Brno, Hradec Králové, Chomutov, Olomouc, Ostrava, Písek, Plzeň. Na Slovenku má pobočky v Bratislavě a Žilině. Pobočky a služby Abry doplňuje také rozsáhlá partnerská síť. (abra.eu)

V roce 2016 a 2017, zažila Abra Software obrovské změny. Abra prošla kompletním rebrandigem, změnou loga a tváře společnosti. Společnost nyní daleko více prezentuje své hodnoty široké veřejnosti. Abra je tak svou novou prezentací otevřena širšímu okruhu veřejnosti. Produkty z řady Abra Gx (G1, G2, G3 a G4) byly sjednoceny v jeden komplexní produkt (Abra Gen). Je snaha o zjednodušování a zpřehlednění. V roce 2016 byla také vytvořena kompletní švýcarská verze informačního systému Abra Gen. A v roce 2017 oficiálně založena švýcarská dceřiná společnost. Abra Software tak míří a pomalu vstupuje na švýcarský trh a získává své první švýcarské partnery.

3.2 ERP Abra Gen

Co znamená zkratka ERP? Anglická zkratka ERP zastupuje slovosled Enterprise Resource Planning. To může být volně přeloženo jako plánování podnikových zdrojů. Jedná se tedy o jakýsi prostředek, který si klade za cíl pomoci podnikateli efektivně řídit jeho zdroje. (Basl, Blažíček, 2012), (Bruckner, Buchalcevová, Voříšek, 2012)

Ve většině případů se jedná o konkrétní softwarové řešení/aplikaci, používané pro řízení, sběr a analýzu podnikových dat s cílem zefektivnění podnikových procesů. Takovýto komplexní informační systém při správném využití a optimalizaci dokáže například ušetřit lidské zdroje či zautomatizovat firemní procesy. (Basl, Blažíček, 2012), (Bruckner, Buchalcevová, Voříšek, 2012)

Informační systém Abra Gen je jedním z řady produktů Abra od společnosti Abra Software a.s. Může být ideálním řešením pro podporu řízení firmy a evidenci podnikových procesů pokrývajících oblasti obchodu, ekonomických a účetních agend a manažerské řízení a plánování. Je vhodný pro velké firmy, podnikatele, neziskové organizace i státní správu, ale mohou je využít i střední a menší firmy. (ABRA Help, help.abra.eu)

Systém Abra Gen je modulární systém. Skládá se z jednotlivých modulů, které mohou fungovat i samostatně (a to zcela nebo částečně). V celku ale tvoří integrovaný a provázaný systém. (ABRA Help, help.abra.eu)

Jedná se o systém s třívrstvou architekturou Client/Server, která zajistí optimální provozní vlastnosti i při větším počtu přihlášených uživatelů. Data jsou spravována relační transakční SQL databází. (ABRA Help, help.abra.eu)

V roce 2016 a 2017 zažila Abra Software a.s. obrovské změny. Abra prošla kompletním rebrandigem, změnou loga a tváře společnosti. Původní produkty z řady Abra Gx (G1, G2, G3 a G4) byly sjednoceny v jeden komplexní produkt Abra Gen.

Aktuálně je systém k dispozici v následujících variantách:

- Abra Gen pro Firebird v provedení pro Abra Gen s daňovou evidencí nebo Abra Gen s podvojným účetnictvím (původní G1, G2 a G3); vyžaduje databázový server Firebird,
- Abra Gen pro Oracle pouze s podvojným účetnictvím (původní G4); vyžaduje databázový server ORACLE,
- Abra Gen pro MSSQL pouze s podvojným účetnictvím (původní G4); vyžaduje databázový server MSSQL. (ABRA Help, help.abra.eu)

Aplikační a databázový server systému Abra Gen je možné provozovat jak na Windows, tak i na Linuxu. (ABRA Help, help.abra.eu)

3.3 Moduly informačního systému Abra Gen

Systém Abra Gen je modulární systém, skládá se ze základních modulů, které se pak dále dělí na agendy specifické pro daný modul. Tyto moduly mohou fungovat i samostatně. Některé moduly je možné zakoupit a provozovat zcela samostatně, některé je možné provozovat pouze spolu s jiným modulem.

Prostřednictvím agend lze pracovat se vším potřebným a důležitým ať už je to v modulu Prodej, kde se dají sledovat nabídky a objednávky, pracovat s fakturami a ceníky. Nebo například modul Mzdy, kde je možné pracovat s agendou personalistika, která dovoluje sledovat vše potřebné týkající se zaměstnanců.

Mezi základní a nejdůležitější moduly se řadí:

- Prodej
    - skupina agend umožňující veškeré operace týkající se prodeje,
- Nákup
    - skupina agend umožňující veškeré operace týkající se nákupu,
- Skladové hospodářství
    - skupina agend a číselníků určených pro práci se sklady a logistické operace,
- Adresář firem
    - skupina agend a číselníků umožňující práci s adresáři osob a firem,
- Maloobchodní prodej
    - skupina agend pro podporu pokladního prodeje,
- Pokladna
    - skupina agend umožňují práci s pokladními doklady,
- Účetnictví
    - skupina agend umožňující veškeré operace v účetnictví,
- Mzdy
    - skupina agend týkající se zpracování mezd a personalistiky,
- Majetek,
    - skupina agend a číselníků pro evidenci majetku,
- Banka
    - skupina agend umožňují provádění operací týkajících se bankovních účtů, výpisů a platebních příkazů,
- Administrace
    - skupina agend a číselníků určená pro administraci systému. (ABRA Help, help.abra.eu)

4. Proces vývoje a vydání verze

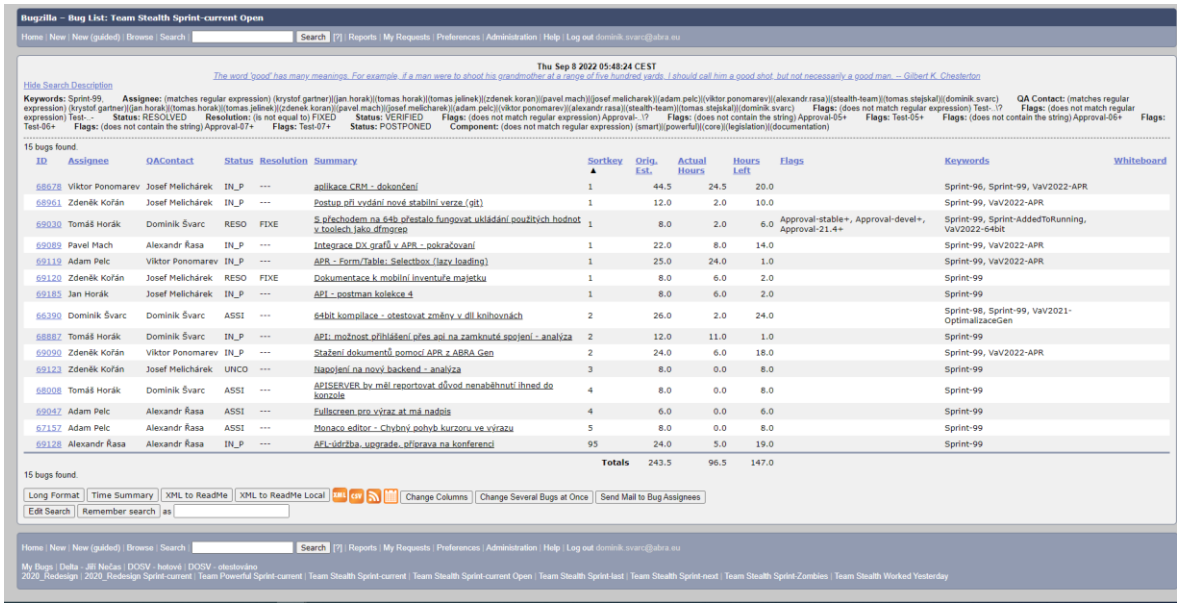
V této kapitole je popsána struktura oddělení R&D, Research and Development. Rozbor struktury je rozdělen do dvou částí. Popis uskupení oddělní je rozdělen na části před a po zavedení metodiky Scrum. Na závěr je popsán proces zavedení metodiky Scrum a jeho využívaných nástrojů.

4.1 Struktura R&D oddělení

Za uplynulých šest let prošlo vývojové oddělení systému Abra Gen spoustou velkých změn. Nejprve zanikl tým Services. Moduly, které měl ve správě připadly zejména týmu Trade. Vevedení oddělení se vystřídalo více lidí. Ovšem největší změny se začaly dít od konce roku 2017, kdy se pomalu začalo přecházet na agilní způsob řízení s využitím metodiky Scrum.

4.2 Evidence požadavků pro vývoj

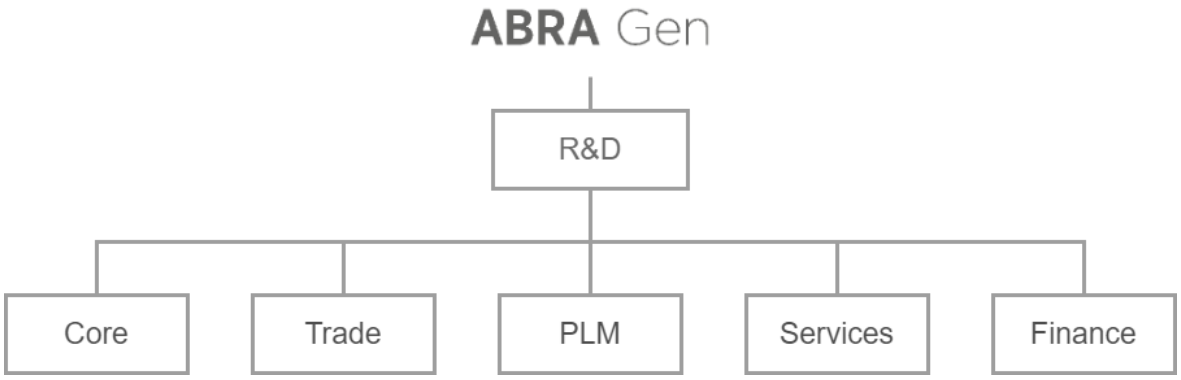
Posledních šest let využívalo vývojové oddělení pro evidenci požadavků webovou aplikaci pro sledování defektů neboli bugů s názvem Bugzilla. Bugzilla je robustní upravitelný systém pro evidenci úkolů určený především pro vývoj softwaru.



Obrázek 3: Snímek obrazovky systému evidence úkolů Bugzilla
Zdroj: vlastní tvorba

4.2.1 Struktura oddělení před zavedením metodiky Scrum

Před zavedením agilního způsobu řízení existovala na vývojovém oddělení pozice Abra Gen R&D Directora (vedoucí divize). Vedoucí divize byl zodpovědný za její činnost a bezproblémový chod. Toto oddělení obsahovalo celkem deset týmů, zabývající se vývojem. Z nichž vývoji produktu Abra Gen se věnovalo celkem pět týmů. Byly to týmy: Core (Jádro), Trade (Prodej), PLM (Výroba), Services (Servis) a Finance (Finance). Každý z týmů měl svého Team Leadera, který zodpovídal za jeho činnost a výsledky. Dále se povětšinou skládal z developerů (vývojářů) a testerů. Jak již bylo zmíněno, produkt je rozdělen do několika modulů, kdy každý tým měl ve správě určité moduly.

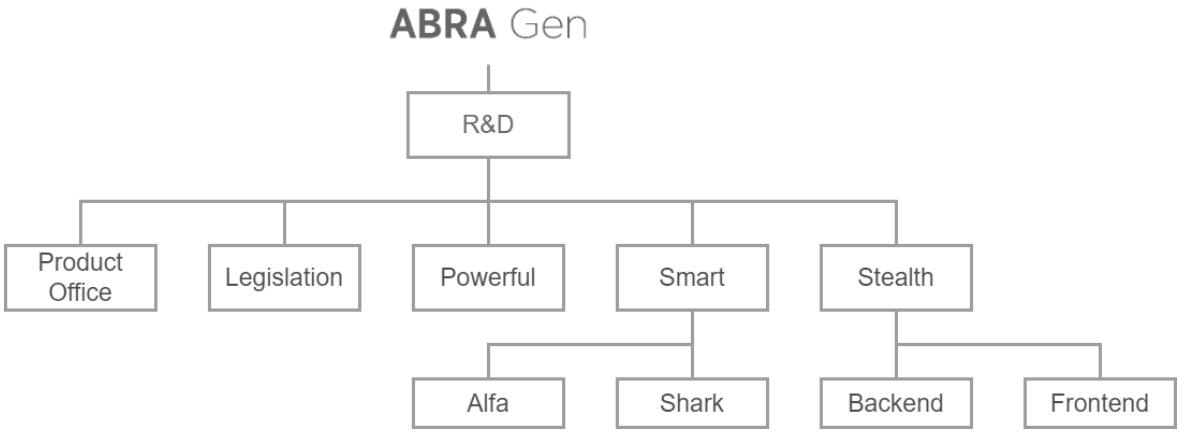


Obrázek 4: Organizační schéma vývojového oddělení před zavedením metodiky Scrum
Zdroj: vlastní tvorba

Před zavedením metodiky Scrum měl každý tým v evidenčním systému Bugzilla filtr pro zobrazení omezeného seznamu úkolů, který spadal pod správu konkrétního týmu. Do tohoto seznamu neustále přibývaly další a další požadavky, a tím se mohl jevit jako nikdy nekončící katalog úkolů. Jednotlivé úkoly přiřazoval konkrétním vývojářům Team Leader. Vývojové oddělení přistupovalo k řešení požadavků vodopádovou metodou. S plynoucím časem bylo nevyhnutelné, že se seznamy úkolů stanou nepřehledné. Ztratil se tak celkový přehled o dokončené práci. Vše vyústilo v přetrvávající problém, nemožnosti vydat certifikovatelnou verzi. Až ke konci roku 2017 nastoupil na vývojové oddělení Michal Ježek, který měl za úkol zmapovat procesy ve vývoji a přinést zkušenosti s agilním řízením. Ještě před nástupem na vývoj Abra Gen zaváděl agilní metody v divizi Abra Flexi.

4.2.2 Struktura oddělení po zavedení metodiky Scrum

Po nějaké době a ustálení procesů se definovala nová struktura vývojového oddělení, která by měla být flexibilnější, co se týče řešení změn v plánech pro vývoj a odpovídat tak nárokům agilních způsobů. Oddělení se nově skládá z týmů Product Office, Legislation, Powerful, Smart a Stealth nesoucí názvy odpovídající firemní strategii pro tvorbu chytrého, rychlého a „neviditelného“ systému. Týmy jsou zaměřením jednotlivých vývojářů vyrovnané, tak aby bylo možné jednoduše nahrazovat a doplňovat kapacity podle aktuálních potřeb. Nově už neexistují role Leaderů, ale každý tým zastupuje takzvaný Product respektive Area Owner, který je zodpovědný za dodanou a odvedenou práci týmu. Každý tým má také svého Scrum Master, který týmu pomáhá zajišťovat jeho bezproblémový chod a moderuje jednotlivé Scrum události.



Obrázek 5: Organizační schéma vývojového oddělení po zavedení metodiky Scrum
Zdroj: vlastní tvorba

4.3 Zavedení metodiky Scrum

Prvním týmem, který začal experimentovat s agilními způsoby byl tým Trade pod vedením Jana Javůrka. Michal Ježek přijal roli Scrum Mastera a začal tým seznamovat s praktikami vedoucími k lepší transparentnosti a flexibilitě.

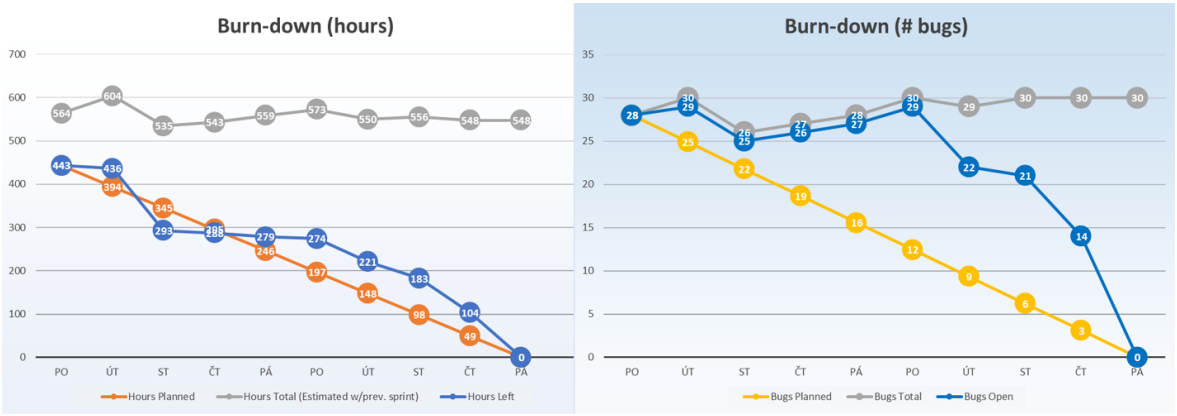
4.3.1 Používané nástroje

Jako první byly nastaveny čtrnácti denní Sprinty s každodenní ranní poradou s názvem Standup, nebo Daily Scrum, kde jednotliví členové shrnují svou práci za uplynulý předchozí den. Dalším krokem bylo zavedení události plánování Sprintu (Sprint Plannig) jehož součástí byla, také stručná restrospektiva s ohlédnutím zpět s otázkami, co se povedlo?, co se nepovedlo?, co zlepšit? a s čím nepokračovat? Seznamy požadavků byly s každým dalším Sprintem přehlednější. Pro sledování náročnosti úkolu byl zaveden systém hodnocení v hodinách. Začal se využívat Burn-Down graf, který pomáhal hlídat průběh Sprintu. Postupem času se standardem stal odhad týmu na přibližně dvě stě padesát hodin odvedené práce, což většinou odpovídalo přibližně dvaceti pěti požadavkům. Vývojáři si práci nyní volí sami a Product Owner má více času řešit podstatné záležitosti.

Postupně se zavedly následující nástroje agilních způsobů:

- čtrnáctidenní Sprinty,
- ranní Daily Scrum porady,
- plánování Sprintu (Sprint Plannig), kdy součástí byla i retrospektiva,
- sledování náročnosti úkolů v hodinách.

Aktuálně je snaha o nastavení i zbývajících Scrum událostí, tak jak mají být. Jedná se o plnohodnotnou retrospektivu sprintu, Sprint Review a pravidelné odhady složitosti úkolů při Refinement schůzkách.



Obrázek 6: Burn-Down evidence vykázané práce v hodinách
Zdroj: vlastní tvorba

5. Automatické testy

Kapitola je věnována možnostem automatizovaného testování. Vychází z nabytých poznatků a zkušeností při práci na vývojovém oddělení systému Abra Gen. Začíná představením jednotkových unit testů. Dále se zabývá systémovými integračními skriptingovými testy. V závěru představuje počátky vizuálního GUI testování s využitím technologie AutoIt a nabízí letmý pohled na další možnosti.

5.1 Unit testy

Unit test je test ucelené dílčí části kódu. V objektově orientovaném programování se tedy jedná o testování jednotlivých metod. Ve své podstatě jde o ověřování správné funkčnosti dílčích částí neboli „jednotek“ zdrojového kódu. U unit testů je důležitá jejich jednoduchost. V případě, že by test narostl do velikosti, mohl by být problém s jeho udržitelností. Méně je někdy více. Je lepší napsat test jen pro testovanou/vyvíjenou funkcionalitu, ostatní funkcionality jsou testované jinými samostatnými testy. Autor unit testu by si měl ověřit nejenom, že test prochází, kdy má, ale také to, že neprochází, kdy procházet nemá.

Tyto testy jsou nezávislé na vnějším prostředí, to znamená například, že test bude funkční mezi všemi vývojáři nehledě na odlišnosti jejich vybavení. Unit test by měl být rychlý. Unit test zpravidla není pouze jeden, ale bývá jich větší sada a neměly by zdržovat vývojáře v práci.

Historicky byl pro systém Abra Gen problém psát unit testy, protože byla programovaná v Delphi 7. Byl problém zejména s odcloněním vnějšího prostředí. Skutečné unit testování systému Abra Gen bylo v počátcích velice nesmělé. Abra Gen hodně využívá INTERFACE objekty, které bylo velice pracné nahrazovat pomocnými objekty (takzvanými mock objekty), které by simulovaly předpokládané chování. V této fázi vývoje tedy příliš unit testů neexistovalo. Před verzí Delphi XE2 byla pro potřeby unit testování do informačního systému ABRA Gen implementována sada testů s agendou Testování. Tyto testy ověřují, zda formálně napsaná „business logika“ neobsahuje chybu, podobu SQL dotazů nebo různé akce proti databázi. Testují se všechny tři platformy. Na tyto testy bylo nahlíženo jako na

jednotkové, ale jelikož běží proti skutečným datům v databázi, jedná se spíše o integrační testy.

Na přelomu let 2014 a 2015 Abra přešla na Delphi XE2. Do této verze byla přidána podpora RTTI umožňující další manipulaci s datovými typy, v tomto případě nahrazování interface metod virtuálními mock objekty. Od této doby se unit testy začaly používat ve větším měřítku. Aktuálně je Abra pokryta přibližně dvanácti sty unit testy.

5.1.1 Příklad použití DUnit

DUnit je open-source testovací framework určený pro ověřování funkčnosti „jednotek“ zdrojového kódu, který je založený na testovacím frameworku JUnit. Framework DUnit umožňuje vytvářet a provádět testy s aplikacemi psaných v Delphi. (DUnit Overview)

Pro objekt, který provádí regulární výrazy je definováno, jak má regulární výraz vypadat, nastaví se proměnná, která se má testovat a ověří se, zda má regulární výraz správnou podobu.

```
procedure TNxRegExTests.TestMatchExeFileName;
begin
// zjištění výchozího klienta pošty (zda se jedná o outlook)
fRegEx.RegEx := '(?i)"?.*?([^\\]*\.exe)"?';
fRegEx.Subject                                                    :=
'"C:\PROGRA~1\MICROS~1\Office14\OUTLOOK.EXE"/recycle';
CheckTrue(fRegExMatch);
CheckEquals('OUTLOOK.EXE', fRegEx.SubExpressions[1]);
end;
```

5.2 Proč unit testy psát?

Pro programátory může být jednodušší, efektivnější a rychlejší ověřit svou práci naprogramováním testu než vizuální prohlídkou v běžící instalaci. S tím souvisí další aspekt pro psaní těchto testů, a to, že jednou napsaný test zůstává a může se automaticky spouštět.

5.3 Automatizace testování při sestavování verzí na Jenkins

Jenkins může sloužit i k automatizaci testování po nějaké větší změně. Mohou se rozlišovat dva typy testů podle doby jejich trvání. A k jejich spuštění není potřeba nic dělat.

První typ testu nastává po commitu do SVN. Po úspěšném commitu, který obsahuje sobory mající vliv na produkt (*.pas, *.dpk, *dpr, …) se Jenkins notifikuje. Po notifikaci Jenkins updatuje pracovní adresář k dané revizi a spustí kompilaci projektu, pokud dojde k chybě, odešle se do konference informace, že je problém s kompilací. Další fází po commitu je spuštění všech testovacích projektů, pokud dojde k chybě, opět se do konference odešle informace o testech, které selhaly. Pokud v žádném kroku nedojde k potížím, odešle se do konference informace, že je stav v SVN v pořádku.

Další typem testů jsou testy noční. Ty trvají déle a probíhají denně, v noci. Tyto testy jsou spouštěny pro jednotlivé verze produktu a testuje se integrita dat. Všechny problémy se reportují do konference, aby o nich byl programátor informován a mohl na ně reagovat.

5.4 Skriptingové systémové/integrační testy

Unit testy otestují jednu konkrétní malou část. ERP Abra Gen je ale masivní informační systém s velkou provázaností, a tak je potřeba a chtěné testovat i kompletní příklady užití. Například, že k vystavené faktuře půjde vystavit skladový doklad, skladový doklad se odešle do logistiky, půjde vystavit bankovní příkaz a tak dále.

Sytém Abra Gen je možné ovládat z vnějšku třemi možnostmi.

Prostřednictvím OLE (Microsoft Object Linking and Embedding), pro který je dostupný kompletní datový model. Otevřené rozhraní AbraOLE je ale pomalu zastarávající funkčnost, která se pouze udržuje a má tak k dispozici jen pouhou část nevizuálních služeb.

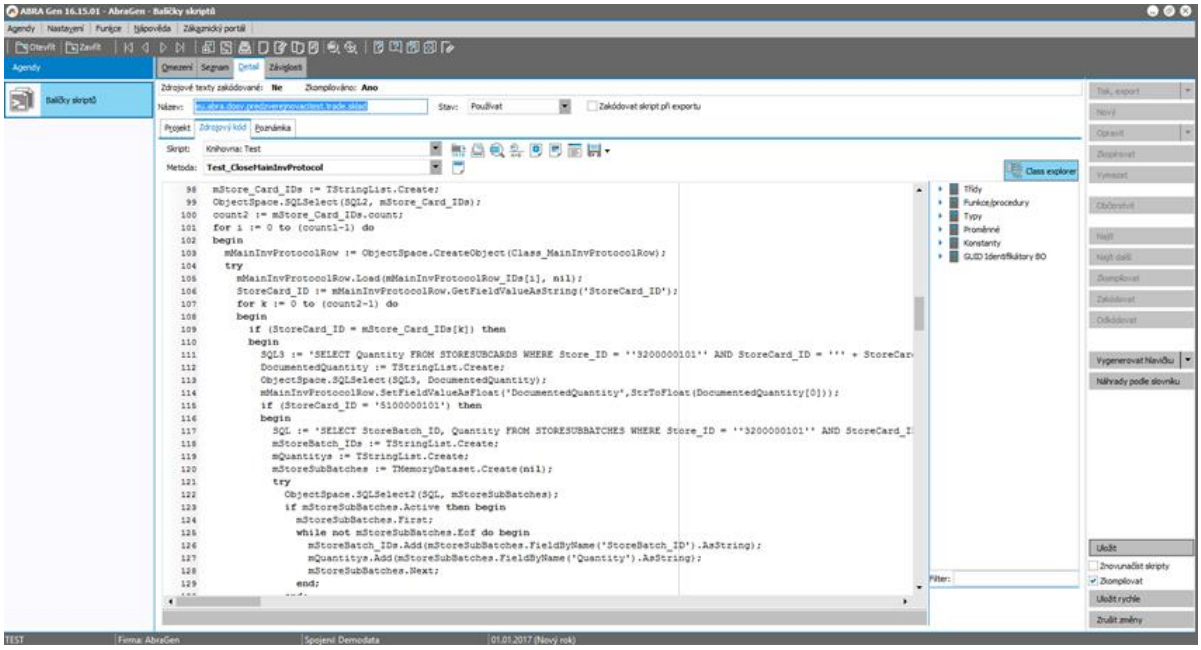
Nejnovějším přírůstkem mezi ovládací rozhraní produktu Abra Gen je Abra WebAPI (https rest api). WebAPI má dostupný kompletní datový model a část nevizuálních služeb. Je tak možné zakládat a editovat doklady, odesílat doklady a dokumenty, komunikovat s jinými systémy, ovládat transakce a tak dále. Abra WebAPI je ve vývoji a součástí systému Abra Gen od roku 2015. Jedná se tak o relativně novou technologii, která je spíše vyhlídkou budoucnosti a na které se bude více a intenzivněji pracovat v následujících letech. Z těchto důvodů není prozatím příliš vhodná pro systémové testování.

Ideálním kandidátem pro psaní systémových integračních testů se jevil skriptovací engine AbraScript. AbraScript je postaven na technologii FastScript. FastScript je více platformní více jazykový skriptovací nástroj, prostřednictvím něhož je možné do vlastních aplikací dodat podporu skriptování. To celé je možné, si představit, jako práci s makry v Excelu, kdy je možné poslat klientu Abra Gen nějaký zdrojový kód, ten si jej přeloží a následně vykoná. Pro Abra Gen skriptování je zveřejněn kompletní datový model, většina nevizuálních služeb (například prostředky EET, eKasy, tvorba dokladu z dokladu, skladové uzávěrky, mzdové uzávěrky, transakce). Do skriptování je vyvedená také velká část vizuálních prvků Abra Gen.

5.4.1 Vývojové prostředí a technologie

Jak již bylo zmíněno Abra skripting je postaven na technologii FastScript, prostřednictvím níž může být do aplikace přidána podpora skriptování. Je více platformní a více jazyková. Vzhledem k tomu, že systém Abra Gen je programován v prostředí Delphi s využitím objektově orientovaného programovacího jazyka Pascal, splňuje FastScript všechny požadavky. Syntaxe skriptů odpovídá jazyku PascalScript, respektive Pascal.

První test byl zpočátku psán přímo v Abra Gen, v jednoduchém editačním prostředí, v agendě Balíčky skriptů, která spadá pod modul Nástroje přizpůsobení.



Obrázek 7: Snímek obrazovky editačního prostředí pro psaní skriptů
Zdroj: vlastní tvorba

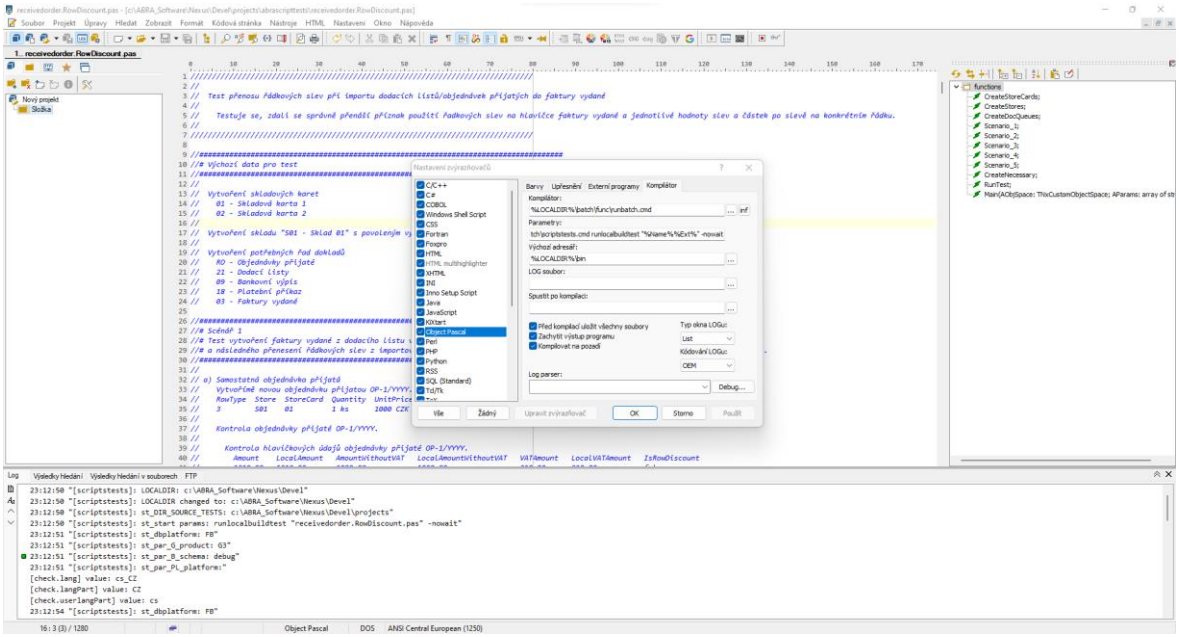
Test byl spouštěn událostí na doskriptované tlačítko v dokladové agendě. Ladění testu bylo prováděné pomocí procedury ShowMessage zobrazením okna se zprávou nebo nástrojem ScriptDebugger.exe. ScriptDebugger umožňuje krokovat běh skriptů, pozastavovat na bodech zastavení (breakpoints), zjišťovat zanoření volání (callstack), vyhodnocovat výrazy, zachytávat ladící hlášení, zjišťovat a měnit hodnoty lokálních proměnných skriptů a další.

Se stupňující se náročností a rostoucími požadavky na výsledný test bylo potřeba strategii psaní, za chodu přehodnotit. V průběhu vytváření prvního testu bylo zjištěno, že v budoucnu bude potřeba skriptingové testy spouštět nevizuálním způsobem, například přes konzoli.

Bylo zjištěno, že Abra Gen nabízí možnost spouštění skriptů prostřednictvím konzolové aplikace AbraScript.exe. Pomocí tohoto nástroje je možné skripty spouštět i z jednoduše editovatelných zdrojových .pas souborů. Více o spouštění testů je uvedeno v následující podkapitole „5.4.4 Spouštění testů“. Takovýto soubor je následně jednoduše editovatelný prostřednictvím téměř jakéhokoliv textového editoru. Vývojáři si postupem času navykli na použití textového editor PSPad nebo Notepad++. Tyto textové editory navíc nabízejí další možnosti vylepšení, jako je například zvýraznění syntaxe odpovídající použitému jazyku nebo usnadnění spouštění výběrem kompilační aplikace a klávesové zkratky pro spuštění. Tyto zjištění a kroky vedly k obrovskému usnadnění vývojářské práce.

Možnosti vylepšení a nastavení spouštění testů v textovém editoru PSPad:

- na kartě „Nastavení“, volba „Nastavení zvýrazňovačů“, vyhledat a zvolit „Object Pascal“,
- na kartě „Kompilátor“ vyplnit důležitá pole: „Kompilátor“, „Parametry“ a „Výchozí adresář“,
- dále se vyplatí zaškrtnout možnosti „Před kompilací uložit všechny soubory“, „zachytit výstup programu“ a „Kompilovat na pozadí“,
- skript je možné spustit klávesovou zkratkou CTRL + F9.



Obrázek 8: Snímek obrazovky textového editoru PSPad upraveného pro potřeby psaní skriptingových testů
Zdroj: vlastní tvorba

5.4.2 Scénář

Před zahájením práce na tvorbě testu je potřeba vymyslet konkrétní scénář, který bude ověřovat požadovanou funkčnost. Obvykle je založen na základě požadavku, který je ve vývoji. Testy se běžně píší po dokončení a naprogramování konkrétního úkolu. Scénář by měl pokrývat požadovanou funkčnost, ale i případy, kdy by procházet neměl.

Pro ilustraci je zde přiložen scénář skriptingového testu receivedorder.rowdiscount.pas, který ověřuje správnost přenosu řádkových slev při importu dodacích listů/objednávek přijatých do faktury vydané v systému Abra Gen.

Test přenosu řádkových slev při importu dodacích listů/objednávek přijatých do faktury vydané. Testuje se, zdali se správně přenáší příznak použití řádkových lev na hlavičce faktury vydané a jednotlivé hodnoty slev a částek po slevě na konkrétním řádku.

Výchozí data pro test:

- vytvoření skladových karet:
    - 01 – Skladová karta 1,

    - 02 – Skladová karta 2,
- vytvoření skladu „S01 – Sklad 01“ s povoleným vyskladněním do minusu bez výstrahy,
- vytvoření potřebných řad dokladů:
    - RO – Objednávky přijaté,
    - 21 – Dodací listy,
    - 09 – Bankovní výpis,
    - 18 – Platební příkaz,
    - 03 – Faktury vydané.

Scénář 1

Test vytvoření faktury vydané z dodacího listu vytvořeného na základě objednávky přijaté bez řádkových slev a následného přenesení řádkových slev z importovaného dodacího listu vytvořeného na základě objednávky přijaté s řádkovými slevami.

a) Samostatná objednávka přijatá

   Vytvoření nové objednávky přijaté OP-1/YYYY.

Tabulka 2: Řádek objednávky přijaté OP-1

Scénář 1_a – nová objednávka OP-1/YYYY				
RowType	Store	StoreCard	Quantity	UnitPrice
3	S01	1	1ks	1000

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

   Kontrola objednávky přijaté OP-1/YYYY.

Tabulka 3: Tabulka kontrolovaných hodnot objednávky přijaté OP-1

Scénář 1_a – kontrola hlavičkových údajů objednávky OP-1/YYYY						
Amount	LocalAmount	AmountWithoutVAT	LocalAmountWithoutVAT	VATAmount	LocalVATAmount	IsRowDiscount
1210,00	1210,00	1000,00	1000,00	210,00	210,00	false

Scénář 1_a – kontrola údajů na řádcích objednávky OP-1/YYYY			
RowType	TAmount	TAmountWithoutVAT	RowDIscount
3	1210,00	1000,00	0,00

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

b) Samostatná objednávka přijatá

Vytvoření nové objednávky přijaté OP-2/YYYY s příznakem použití řádkových slev.

Tabulka 4: Řádky objednávky přijaté OP-2

Scénář 1_b – nová objednávka OP-2/YYYY					
RowType	Store	StoreCard	Quantity	UnitPrice	RowDiscount
3	S02	1	1ks	1000 CZK	20
RowType	Text	TotalPrice	RowDiscount		
1	Text	500 CZK	10		

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

Kontrola objednávky přijaté OP-1/YYYY.

Tabulka 5: Tabulka kontrolovaných hodnot objednávky přijaté OP-2

Scénář 1_b – kontrola hlavičkových údajů objednávky OP-2/YYYY						
Amount	LocalAmount	AmountWithoutVAT	LocalAmountWithoutVAT	VATAmount	LocalVATAmount	IsRowDiscnout
1512,50	1512,50	1250,00	1250,00	262,50	262,50	true

Scénář 1_b – kontrola údajů na řádcích objednávky OP-2/YYYY			
RowType	Tamount	TAmountWithoutVAT	RowDiscount
3	968,00	800,00	20,00
1	544,50	450,00	10,00

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

c) Importovací manager – tvorba dodacího listu DL-1/YYYY z objednávky přijaté OP-1/YYYY z kroku a).

d) Importovací manager – tvorba dodacího listu DL-2/YYYY z objednávky přijaté OP-2/YYYY z kroku b).

e) Vytvoření faktury vydané FV-1/YYYY z dodacího listu DL-1/YYYY.
Importovací manager – tvorba faktury vydané z dodacího listu DL-1/YYYY z objednávky přijaté OP-1/YYYY bez slev.

Kontrola faktury vydané FV-1/YYYY

Tabulka 6: Tabulka kontrolovaných hodnot faktury vydané FV-1

Scénář 1_e – Kontrola hlavičkových údajů faktury FV-1/YYYY						
Amount	LocalAmount	AmountWithoutVAT	LocalAmountWithoutVAT	VATAmount	LocalVATAmount	IsRowDiscount
1210,00	1210,00	1000,00	1000,00	210,00	210,00	false

Scénář 1_e – Kontrola údajů na řádcích faktury FV-1/YYYY			
RowType	Tamount	TAmountWithoutVAT	RowDiscount
3	1210,00	1000,00	0,00

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

f) Import dodacího listu DL-2/YYYY z kroku d) do faktury vydané FV-1/YYYY.
Importovací manager – import dodacího listu DL-2/YYYY z objednávky přijaté OP-2/YYYY se slevami do faktury vydané FV-1/YYYY.

5.4.3 Vývoj testu

Skriptingové testy se při vývoji Abra Gen používají bezmála pět let. Za tuto dobu vzniklo přes tři sta jednotlivých skriptingových testů s více jak čtrnácti sty testovacími scénáři a sedmdesát pět knihoven základních obecných funkcí pro zjednodušení práce vývojářů. Pokryta je kompletní cenotvorba, tvorba většiny dokladů, které je možné v systému Abra Gen vystavit, výsledky reportů i funkčnost napojení na externí služby jako je EET, eKasa, e-maily, datové schránky, ARES a podobně. Testy se vždy píší pro novou funkcionalitu a při změnách ovlivňující datový model. Testy mohou psát jak programátoři, tak i testeři. Skriptingové testy slouží nejenom k ověřování bezchybnosti systému, ale i rychlosti každé vydávané verze systému. Rychlostním testům je věnovaná samostatná podkapitola „5.4.5 Rychlostní testy“.

Při psaní testu je vhodné se inspirovat již vytvořenými testy, zejména těmi posledními a ověřenými.

Testy mají definovanou strukturu názvu, kdy z názvu musí být zřejmé do jaké části systému test spadá, například zda se jedná o jádro, nebo EET a podobně. Název testu by měl být unikátní.

Například: *receivedorder.rowdiscount.pas*

Na úvod testu se popíše testovaná problematika, o co se jedná. Stručně se popíše příklad užití, který je potřeba testovat. Následuje soupis konkrétních testovacích scénářů.

Test přenosu řádkových slev při importu dodacích listů/objednávek přijatých do faktury vydané. Testuje se, zdali se správně přenáší příznak použití řádkových lev na hlavičce faktury vydané a jednotlivé hodnoty slev a částek po slevě na konkrétním řádku.

Při psaní testů se používají již existující knihovny funkcí, které vnikly na základě opakovaných používaných procedur z prvně vzniklých testů, například funkce pro kontrolu hodnot.

Za zmínku stojí tyto základní pomocné knihovny:

- base.lib.pas – základní funkce související s testem, definice globálního ObjectSpace, globální seznam parametrů, výpis varování, výjimek a podobně,
- base.lib.companyparameters.pas – metoda pro zápis nastavení globálních firemních parametrů,
- base.lib_const.pas – seznam konstant (názvy tabulek, položek číselníku a podobné),
- base.lib_businessobject.pas – pomocné metody pro práci s business objektem,
- base.lib_chcekvalue.pas – metody používané pro kontrolu údajů,
- base.lib_necessary.pas – metody pro založení základních dat číselníků,
- base.lib_sql.pas – metody pro SQL vyhledávání v databázi,
- base.lib_stores.pas – metody související se sklady a skladovými řady dokladů, vytváření vazeb mezi sklady a řad dokladů.

```
Uses
  'base.lib_const',
  'base.lib_businessobject',
  'base.lib_necessary',
  'base.lib_stores',
  'base.lib_checkvalue',
  'base.lib_store_basic_const';
```

Skriptingový test má vždy jednu metodu Main, ve které se spouští jednotlivé procedury (založení základních dat, jednotlivé scénáře). Zjednodušeně řečeno je proces následující: vytvoří se databáze, naplní se základními hodnotami, zpracují se testovací scénáře (například vytváření dokladů) a nakonec se testuje výsledek.

```
procedure CreateNecessary;
begin
  FxTest_Start('CreateBasic');             FxNecessary_CreateBasic;
FxTest_Stop;
  FxTest_Start('CreateDocQueues'); CreateDocQueues; FxTest_Stop;
  FxTest_Start('CreateBankAccount');
FxNecessary_CreateBankAccount; FxTest_Stop;
  FxTest_Start('CreateStores'); CreateStores; FxTest_Stop;
  FxTest_Start('CreateStoreCards'); CreateStoreCards; FxTest_Stop;
  FxTest_Start('CreateAllStoresDocQueuesLinks');
FxStores_CreateAllStoresDocQueuesLinks; FxTest_Stop;
end;
```

```
procedure RunTest;
begin
  // Používání původního algoritmu
  FxCompanyParameters_Set([CatParam_SystemVATAlgorithm], ['0']);
  FxTest_Start('Scenario_1'); Scenario_1; FxTest_Stop;
  FxTest_Start('Scenario_2'); Scenario_2; FxTest_Stop;
  FxTest_Start('Scenario_3'); Scenario_3; FxTest_Stop;
  FxTest_Start('Scenario_4'); Scenario_4; FxTest_Stop;
  FxTest_Start('Scenario_5'); Scenario_5; FxTest_Stop;
end;

procedure Main(AObjSpace: TNxCustomObjectSpace; AParams: array of
string);
begin
  FxScriptTest_Begin(AObjSpace, AParams);
  try
    CreateNecessary;
    RunTest;
  finally
    FxScriptTest_End;
  end;
end;
```

5.4.4 Spouštění testů

Pro bezproblémový chod skriptingových testů je potřeba mít systém Abra Gen nainstalovaný nebo zkompilovaný ze zdrojových kódů. Testy se spouštějí nad prázdnou databází, je potřeba mít připravenou prázdnou databázi. Pro zjednodušení, urychlení a opakovaný chod testu nebo sériové spuštění více testů za sebou je vhodné prázdnou databázi zálohovat nativními prostředky databázového serveru. Testy je možné spouštět nad všemi platformami systému Abra Gen, to znamená Firebird (G1 a G3), MSSQL (M4) a Oracle (G4). Testy fungují pouze s platnou licencí. Samotný test se dá spustit na základě nějaké události v Abra

Gen (naskriptované tlačítko nebo vytvoření objektu) nebo konzolovou aplikací AbraScript.exe, které umožňuje.

Základní spuštění přes AbraScript:

```
AbraScript.exe -u uzivatel_prihlasovaci_jmeno -p heslo -c nazev_spojeni soubor_testu.pas
```

S pomalu přibývajícími testy bylo potřeba zdokonalit a zefektivnit spouštění testů, zejména pro potřeby běhu více testů v řadě. Vývojář Jan Javůrek připravil rozsáhlou dávku pro příkazový řádek Windows. Tato dávka dokáže připravit prostředí (zkompilovat systém Abra Gen, vytvořit prázdnou databázi a její zálohu), spustit vybraný test nebo celý seznam testů a obnovit databázi ze zálohy. Spuštění testů funguje i proti nainstalované verzi. Zpočátku bylo možné spouštět testy pouze v lokálním jednouživatelském režimu na jednom konkrétním zařízení. Dávku ve velké míře pomohli rozšířit také vývojáři Jiří David a Alexey Gusev, kteří ji rozšířili o možnost využití síťového provozu. Od této doby je možné spouštět testy také v síťovém režimu klient – server. Dík ovšem náleží i zbývajícím členům původního týmu Trade, kteří se podíleli na dávce běžnou činností a opravou problémů.

Dávka nese název scriptstests.cmd a aktuálně pokrývá následující možnosti spuštění:

- scriptstests.cmd runlocalbuild_prepare_environment
    - příprava prostředí (kompilace Abra Gen ze zdrojových kódů) bez vytvoření a zálohy databáze a bez spuštění testů,
- scriptstests.cmd runlocalbuild_create_source_database
    - vytvoření zdrojového spojení a prázdné databáze na zkompilované verzi ze zdrojových kódů,
- scriptstests.cmd runlocalbuild_backup_source_database
    - vytvoření zálohy databáze ze zdrojového spojení na zkompilované verzi ze zdrojových kódů,
- scriptstests.cmd runlocalbuild_prepare
    - kompletní příprava prostředí (kompilace Abra Gen ze zdrojových kódů, vytvoření zdrojového spojení, prázdné databáze a její záloha) bez spuštění testů,

- scriptstests.cmd runlocalbuild
    - kompletní příprava prostředí (kompilace Abra Gen ze zdrojových kódů, vytvoření zdrojového spojení, prázdné databáze a její záloha) a spuštění všech testů,
- scriptstests.cmd runlocalbuildtests
    - spuštění všech testů proti již kompletně připravenému prostředí zkompilované verze ze zdrojových kódů,
- scriptstests.cmd runlocalbuildtest
    - spuštění jednoho konkrétního testu proti již kompletně připravenému prostředí zkompilované verze ze zdrojových kódů,
- scriptstests.cmd runlocal_create_source_database
    - vytvoření zdrojového spojení a prázdné databáze pro lokální instalaci,
- scriptstests.cmd runlocal_backup_source_database
    - vytvoření zálohy dat ze zdrojového spojení z lokální instalace,
- scriptstesst.cmd runlocal
    - spuštění všech testů proti lokálně nainstalované verzi s kompletní přípravou prostředí (tvorba zdrojového spojení, prázdné databáze a její záloha),

- scriptstests.cmd runlocaltests
    - spuštění všech testů proti lokálně nainstalované verzi bez přípravy prostředí (tvorba zdrojového spojení, prázdné databáze a její záloha),
- scriptstests.cmd runlocaltest
    - spuštění jednoho konkrétního testu proti již kompletně připravenému prostředí lokálně nainstalované verze,
- scriptstests.cmd runremote
    - spuštění všech testů proti nainstalované verzi na vzdáleném testovacím serveru s kompletní přípravou prostředí (tvorba zdrojového spojení, prázdné databáze a její záloha), určený zejména pro testování v síťovém víceuživatelském režimu,

- scriptstests.cmd runremotetests
    - spuštění všech testů proti nainstalované verzi na vzdáleném testovacím serveru bez přípravy prostředí (tvorba zdrojového spojení, prázdné databáze a její záloha), určený zejména pro testování v síťovém víceuživatelském režimu.

Dávku a způsob jejího provedení je možné ovlivnit následujícími parametry:

- pro localbuild (zkompilovaná verze ze zdrojových kódů)
    - /G: typ produktu [G1, G3, G4, M4] – výchozí hodnota je G3,
    - /B: schéma [debug, release], udává kompilační schéma – výchozí hodnota je debug,
    - /L: verze lokalizace [cz_CZ, sk_SK, de_CH],
    - další možné nepovinné parametry: /AS, /K:, /C:, /E:, /D:, /ORAOCI:, /ORAHOME:,
- pro local (lokálně nainstalované verze)
    - /P: cesta k lokálně nainstalované verzi,
    - další možné nepovinné parametry: /AS, /K:, /C:, /E:, /D:, /ORAHOME:,
- pro remote (verze nainstalovaná na vzdáleném testovacím serveru pro běh v síťovém víceuživatelském režimu)
    - /F: adresář, kde se nachází nainstalovaná verze na vzdáleném testovacím serveru,
    - /S: název serveru,
    - /U: přihlašovací jméno uživatele pro vzdálený přístup do Windows na serveru – výchozí hodnota Administrator,
    - /A: heslo uživatele pro vzdálený přístup do Windows na serveru,
    - /N: síťová cesta k adresáři nainstalované verze – výchozí hodnota \\název serveru\adresář, kde se nachází nainstalovaná verze,
    - další možné nepovinné parametry: /K:, /C:, /E.

Popis nepovinných parametrů:

- /AS udává, zda se mají testy spouštět v režimu s aplikačním serverem,
- /K: typ testů, spuštění pouze vybraného typu testů [speed, fast],
- /C: název zdrojového spojení databáze pro spuštění testů – výchozí název SCRIPT_TEST,
- /E: název zdrojového spojení pro zálohovanou databázi – výchozí název je SCRIPT_TEST_SOURCE,
- /D: název serveru, kde běží databáze – výchozí název je localhost,
- /ORAOCI: komunikační knihovna Oracle – výchozí cesta C:\Database\Oracle\product\instantclient_12_1_x86\oci.dll,
- /ORAHOME: cesta k domovskému adresáři Oracle – výchozí cesta C:\Database\Oracle.

Spuštění skriptingových testů běžně probíhá následujícím způsobem:

- příprava prostředí
    - kompilace nebo instalace verze,
    - vytvoření zdrojového spojení a prázdné databáze (Firebird, MSSQL, Oracle),
    - nativní záloha databáze, pro urychlení celého procesu, zejména obnovy, bylo nejlepším řešením využití nativních prostředků sloužících k zálohování daných databázových serverů,
- spuštění a běh testů
    - Test I.
        - zastavení aplikačního serveru,
        - obnova databáze ze zálohy,
        - spuštění aplikačního serveru,
        - spuštění skriptingového testu,
    - Test II.
        - …

- běh testu
    - příprava dat k testu
        - nastavení parametrů systému,
        - založení potřebných záznamů (období, účetní předkontace, řady dokladů a podobné),
    - spouštění a průběh jednotlivých testovacích scénářů
        - Scénář A
            - tvorba záznamů,
            - kontrola výsledných hodnot,
        - Scénář B
            - …

5.4.5 Rychlostní testy

Pomocí skriptingových testů je také možné sledovat rychlost systému. V souvislosti se změnami struktury a způsobu vedení vývoje a tlaku na výkonnostní zlepšení systému Abra Gen tak vzniklo několik rychlostních zátěžových testů. Sleduje se, jak dlouho jednotlivé testy běží. Abra Gen má v sobě také integrovaný profiler, který vyhodnocuje veškeré datové toky, komunikace, prováděné SQL dotazy a podobné. Testy je tak možné spustit i se zapnutým profilováním, kdy se ukládají výsledky. Tyto výsledky je následně možné porovnávat s očekávanými hodnotami a vyhodnotit, zda Abra Gen nevykonává některé procesy neefektivně nebo nadbytečně. Aktuálně je systém Abra Gen pokryt sedmnácti rychlostními testy.

5.5 Vizuální GUI testy

Automatické vizuální GUI testy jsou teprve v počátcích vývoje. Výhoda vizuálního GUI testování je, že není potřeba se příliš zabývat strukturou datového modelu a tím jaký databázový engine funguje na pozadí. U všech platforem (Firebird – G1, G3, MSSQL – M4 a Oracle – G4) je vizuální chování takřka identické. Testy, které byly psané pro Firebird fungují bezproblémově například i na Oraclu.

Při generování verze k certifikaci se provádějí vždy takzvané předzveřejňovací testy. Tyto testy se dříve prováděly pouze manuálně na základě konkrétních a v podstatě neměnných scénářů. Vždy se jednalo v podstatě o uživatelský průchod systémem Abra Gen s přesně daným postupem, kdy se tester zaměřoval zejména na správnost hodnot a korektní vykreslování vizuálních prvků. Tým Trade měl k dispozici šest takovýchto scénářů. Z nichž se nejčastěji pracovalo se scénáři pokrývajícími moduly nákupu a prodeje, skladového hospodářství a maloobchodního prodeje. Manuální provádění těchto předzveřejňovacích testů je ale časově velice náročné. Náročné jsou také na testerovu pozornost. Provedení takového testu může trvat až pět hodin, což se může negativně projevit zejména při přegenerování verze. Některé scénáře nemusejí být sepsány v takové podobě, aby vyhovovaly všem. Například v případě, kdy je potřeba scénář upravovat nebo doplňovat o nové funkcionality se může stát hotový scénář méně přehledným, kdy i zkušený tester může chybovat a následně je nucen předzveřejňovací test vykonat znovu. Tyto testy a jejich scénáře jsou co se týče databáze z principu destruktivní. V případě, kdy dojde k chybě v postupu, je nutné test opakovat od začátku. Z důvodu celkově větší náročnosti může také docházet při opakovaném přegenerovávání k vynechávání testů. Díky této náročnosti se tyto testy používají jen při generování verze. Všechny tyto neduhy mohou vést například k nedetekovanému nabourání funkčnosti jiným týmem, k většímu stresu a komplikacím před certifikací a uvolněním verze. V nejhorším případě může dojít ke zdržení verze.

Prvním pokusem o zahájení automatického vizuálního GUI testování byl první Abra Hackathon. Hackathon je událost, kdy mimo klasický plán práce, který vychází z agilní metodiky Scrum mají vývojáři možnost se věnovat něčemu čemu sami chtějí. Jan Rychlík v rámci Hackathonu analyzoval možnosti automatického GUI testování. Jako nástroj k automatizaci byl zvolen skriptovací jazyk AutoIt, zejména z důvodů, že AutoIt je k dispozici zdarma a i pro testera můžu výsledný kód působit přehledně a čitelně.

Velkou zásluhu na sepsání předzveřejňovacích testů do podoby AutoIt automatických GUI testů má Daniel Šíma. Největším problémem při psaní testů je odezva systému Abra Gen, kdy odezva například načítání kontrolních prvků není vždy stejná a závisí i na konkrétním prostředí, kde Abra běží. Často tak může docházek k neúspěšnému ukončení průběhu testu, kdy Abra Gen nestihne vykreslit prvek a AutoIt skript se snaží pokračovat. Některé prvky také nejsou pro AutoIt viditelné. Za tímto účelem byl vytvořen nástroj AbraGUITool jehož prostřednictvím programátoři dokážou potřebné prvky zviditelnit i pro AutoIt.

5.5.1 Vývojové prostředí a technologie

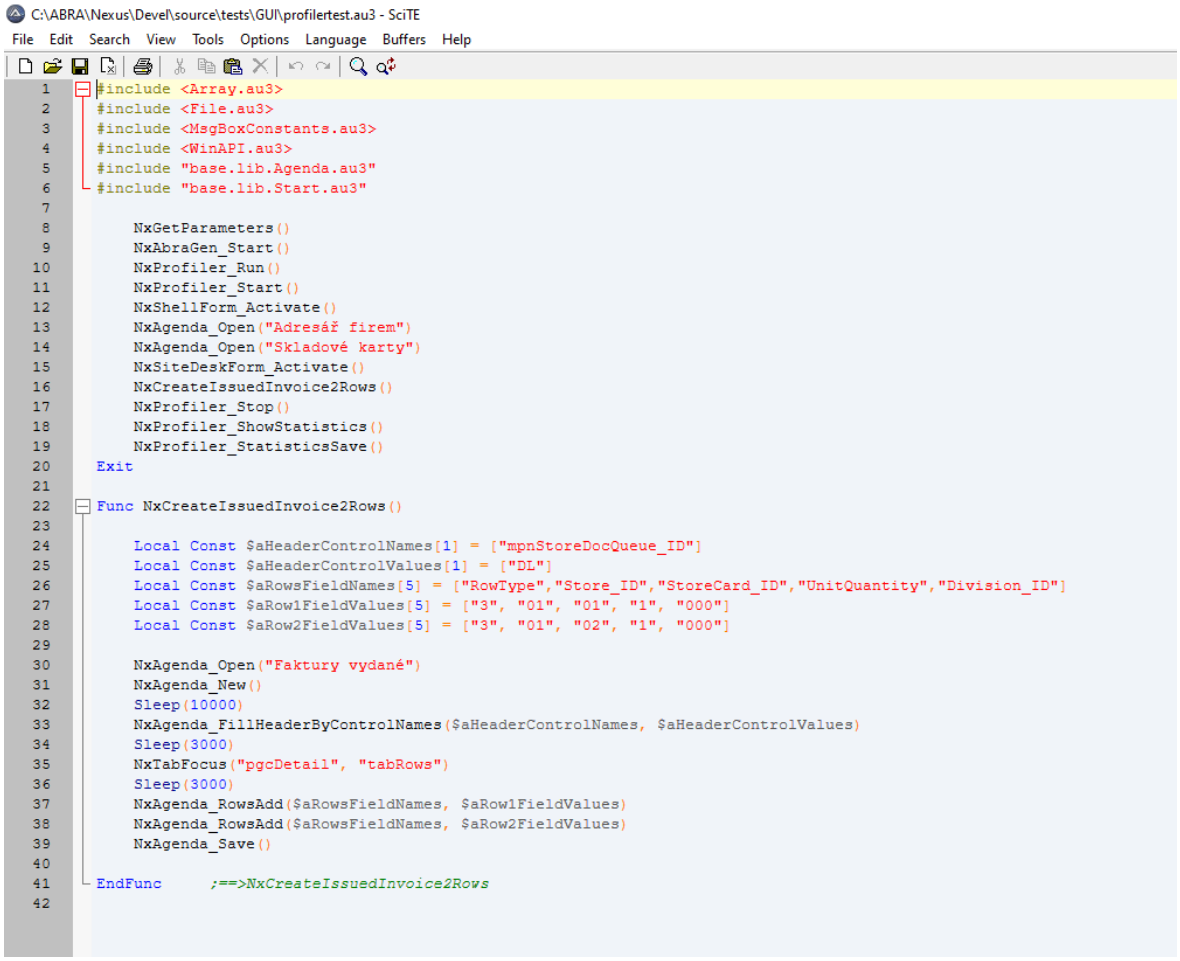
AutoIt je skriptovací jazyk podobný jazyku BASIC, navržený pro automatizaci Windows GUI a obecné skriptování. Využívá kombinaci simulací „mačkání“ tlačítek, pohybů myší a manipulaci s okny, za účelem nekonvexní automatizace. AutoIt je jako program velice malý a dokáže běžet na jakýchkoliv „out-of-the-box“ Windows. Zároveň byl navržen, tak aby fungoval bez jakýchkoliv externích .dll souborů. (AutoIt Scripting Language)

AutoIt začíná být více a více populární v administrativě, kde se využívá k automatizaci administrativních úkonů. I když je nejvíce populární právě v administrativě, dá se AutoIT využít k automatizaci čehokoliv v prostředí Windows. AutoIT dokáže fungovat s jakýmkoliv programem a dokáže manipulovat s klávesnicí a myší. Administrátoři dokáží s pomocí funkce RunAs vytvořit instalace a změny, které nepotřebují dohled pro jejich dokončení. (Flesner, 2007)

AutoIt začal ke konci roku 1998 kde pomáhal automatizovat „mačkat“ klávesy při instalacích softwaru. První verze AutoIt v1 byla vydána v roce 1999. Tato verze obsahovala funkce: Send, Run, RunWait, WinWait, WinWaitClose, WinWaitActive, WinHide, WinActivate, WinClose, WinRestore, Sleep a SetKeyDelay.V roce 2001 byl AutoIt přepsán do C++. První verze s GUI byla vydána v roce 2004 (AutoIt v3.1.0). GUI v AutoIt nabízí možnost instalačních menu, vstupních formulářů a indikátorů postupu. (Flesner, 2007)

SciTE je textový editor na bázi Scintilly, jehož prvotním cílem byla demonstrace právě Scintilly. Rozrostl se však na obecně užitečný editor se schopností tvořit a rozbíhat programy. (SciTE4AutoIt3)

Bylo zjištěno že SciTE funguje dobře jako editor AutoIt a proto vznikl instalační program SciTE4AutoIt3, který obsahuje SciTE a další komplementární programy. Ty dále pomáhají s konfiguracemi a schopnostmi SciTE fungovat společně s AutoIt. (SciTE4AutoIt3)



Obrázek 9: Snímek obrazovky prostředí SciTE pro tvorbu automatických GUI testů
Zdroj: vlastní tvorba

5.5.2 Scénáře

V této kapitole je poskytnuta ukázka scénáře předzveřejňovacího testu. Konkrétně se jedná o test pokrývající moduly nákupu a prodeje jehož průběh je formou videozáznamu přiložen k této diplomové práci. Scénář je psán pro konkrétní data. Před začátkem provádění testu je potřeba obnovit výchozí data.

1. Založit skladovou kartu se sériovými čísly:
   Třida: Se sériovými čísly; Kód: TV; Název: Televize; Typ: Z; DPH: 21%; Dealerské slevy: DealSlevA; Kusové slevy: KusSlevaA; Sortimentní skupina: SortimentA; Struktura sériového čísla: Auromaticky generovat: ANO; Prefix: „TV“; Délka těla: 5; Sufix: 0

2. Přidat skladovou kartu TV do ceníku CenikA:

Tabulka 7: Tabulka cen skladové karty TV

Cena1	1 111,000
Cena2	4 500,000
Cena3	3 150,000

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

3. Vystavit OP:

   Firma: ABC (00002); Typ obchodu Tuzemský; Měna: CZK; DPH: Ano; Částky jsou: Bez DPH;

   Řádky:

   Zaokrouhlení DPH: Nahoru na 1.000; Celkové zaok.: Aritmeticky na 0,010; (karty ze skladu 02 – Hlavní sklad);

Tabulka 8: Tabulka řádků objednávky přijaté

Č.ř.	Typ ř.	Kód skl.k./Text	Počet	J.cena	DPH
1	0	Text 0	-	-	-
2	1	Text 1	-	150550	14
3	2	Text 2	2 hod	500290	21
4	3	TV	5 ks	2 603,306	21
5	3	8	2 ks	1 205,785	21

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

   Celkem: bez daně: 14 765,54; DPH: 3 094,00; Celkem: 17 859,54;

   Doklad vytisknout – sestava „Formulář objednávky přijaté“

4. Na základě OP vystavit poptávkový list:

   Oslovení dodavatelé:

Tabulka 9: Tabulka oslovených dodavatelů

Skladová karta	Firma	Potvrzený počet	Nabídnutá j.cena	Vítěz
TV - Televize	Galenit (00003)	5	2 000	Ano
TV - Televize	Alfa s.r.o. (00011)	5	2 220	Ne
08 - DVD rekordér SONY	Galenit (00003)	2	1 950	Ano

Zdroj: vlastní tvorba na základě scénáře předzveřejňovacího testu

   Poptávkový list uzavřít a vytvořit OV:

   Zaokrouhlení DPH: Nahoru na 1,000; Celkové zaok.: Aritmeticky na 0,010;

   Celkem bez daně: 13 900,00; DPH: 2 919,00; Celkem: 16 819,00

5.6 Další možnosti automatických testů

Další z možností automatizovaného testování jsou testy prováděné prostřednictví Abra WebAPI technologie. Aktuálně se možnosti testování prostřednictvím WebAPI zkoumají. Pro psaní těchto testů bude zřejmě použit jazyk Python.

Další z možností testování systému Abra Gen prostřednictvím WebAPI nebo konkrétně samotného WebAPI jsou možnosti kolekcí v aplikaci Postman. Pro tyto účely byla vytvořena kolekce všech základních dotazů ze stránek nápovědy systému Abra Gen. Součástí jednotlivých požadavků je také jednoduchý test typu odpovědi a její přesné podoby json formátu.

Test může vypadat následovně:

```
pm.test("Status code is: " + pm.response.code, () => {
    pm.expect(pm.response.code).to.eql(200);
});
var expectedJsonResponseBodyStructure =
[{
        "ID": "1400000101",
        "Amount": 8330,
        "DocNumber": "FV-1/0000"
    },
    {
        "ID": "2400000101",
        "Amount": 500,
        "DocNumber": "FV-2/0000"
    }]
pm.environment.unset("expectedJsonBody");
pm.environment.set("expectedJsonBody",
expectedJsonResponseBodyStructure);
pm.test('Response structure is valid', function() {
    var jsonBody = pm.response.json();
  pm.expect(jsonBody).is.to.eql(pm.environment.get("expectedJsonBody"));
});
```

6. Vyhodnocení zavedení automatizace testování a agilních způsobů řízení

Poslední kapitola je věnována vyhodnocení získaných poznatků a zkušeností nabytých při zavádění agilního způsobu řízení a automatizovaného testingu.

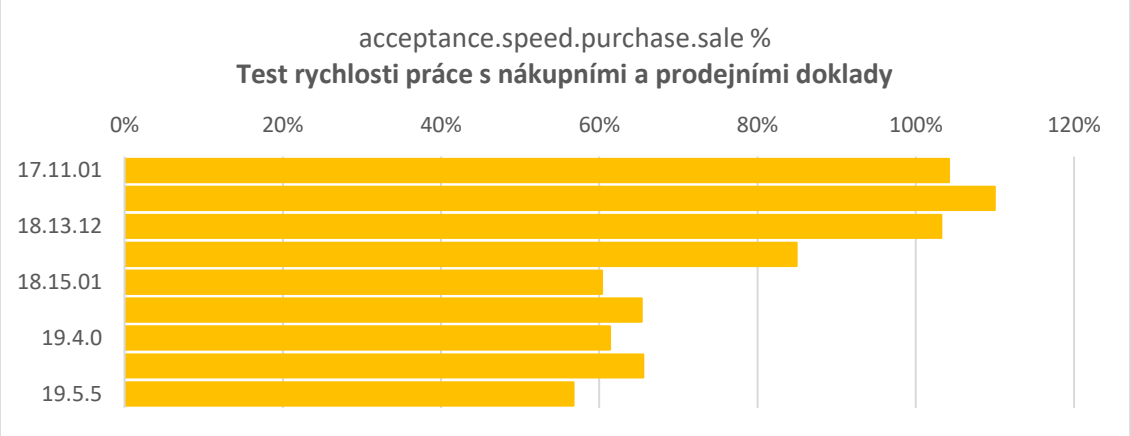
Po přijetí agilních způsobů a metodiky Scrum došlo k obrovskému zpřehlednění procesů vývoje. Všechny týmy mají vždy naprostý přehled o dokončené a nedokončené práci. Nemůže tak dojít k několikaměsíčním odkladům, generování a certifikace verze. Vývojáři vcelku rychle vzali pravidla za svá a práci si volí sami. Návyk denních Daily Scrum porad považují také za přínosný, případné blokace jsou odhaleny daleko dříve. Mezi týmová spolupráce se díky sdílení kapacit a zlepšené flexibilitě také výrazně zlepšila. Vývoj má nyní k dispozici i manipulační prostor pro případné projekty navíc mimo definovaný plán.

6.1 Přínos skriptingových testů

Jak již bylo zmíněno za dobu existence skriptingových testů, vzniklo přes tři sta jednotlivých testů s více jak čtrnácti sty testovacími scénáři a sedmdesát pět knihoven základních obecných funkcí pro zjednodušení práce vývojářů. Pokryta je kompletní cenotvorba, tvorba většiny dokladů, které je možné v systému Abra Gen vystavit, výsledky reportů i funkčnost napojení na externí služby jako je EET, eKasa, e-maily, datové schránky, ARES a podobně.

Skriptingové testy snad absolutním způsobem zabránily opakovaným chybám a potřebě generovat opravy. Vývojáři tak mají více času na tvorbu nových funkcionalit, namísto opravování opakujících se chyb. Po tolika letech je množství ušetřeného času na opravách těchto chyb v podstatě nevyčíslitelné. Skriptingové testy a jejich časté používání dodávají jistotu dobře odvedené práce celému oddělení vývoj.

Velkým plusem skriptingových testů je také možnost sledování rychlosti systému Abra Gen. Vývojové oddělení v posledních letech vynaložilo obrovské úsilí na odstranění výkonnostních nedostatků. Systém Abra Gen se za poslední roky zrychlil místy až o čtyřicet procent.



Obrázek 10: Ukázka evidence výsledků rychlostních testů v grafické podobě
Zdroj: vlastní tvorba

6.2 Přínos vizuálních GUI testů

Bylo zjištěno, že díky automatickým GUI testům je časová úspora oproti manuálnímu provádění předzveřejňovacích testů kolem deseti hodin:

- příprava dat – třicet osm minut oproti třiceti minutám obnovy dat pro každý test,
- nákup a prodej – patnáct minut oproti třem hodinám,
- skladové hospodářství – jedenáct minut oproti třem hodinám,
- maloobchodní prodej (CZ) – dvanáct minut oproti dvou a půl hodinám,
- maloobchodní prodej (SK) – dvacet pět minut oproti třem a půl hodinám.

6.3 Problémy na cestě automatizace

Největším problémem při analýze nových možností je vždy časová náročnost potřebná pro seznámení s danou problematikou.

Při vytváření skriptingových testů může být největším problémem pouze nedostatečná znalost syntaxe použitého jazyka a nedostatečné programátorské zkušenosti.

V souvislosti s automatickými vizuálními GUI testy byl zřejmě největší problém s použitím zastaralé technologie. AutoIt je postaven na skriptovacím jazyce Basic a provádění skriptů tak není ničím omezeno a je rychlé. Systém Abra Gen má, ale občasné problémy s odezvou načítání vizuálních prvků, a tak se často může stát, že AutoIt skript předběhne ve vykonávání kroků Abru a test následně selže.

6.4 Vyhlídky do budoucnosti

Pro skriptingové testy:

- automatické spouštění testů a zápis výsledků pomocí Jenkins
    - po každém vygenerování verze,
    - po commitu do SVN (jen vybrané klíčové testy),
- funkčnost testů v režimu aplikační server – klient i pro ostatní databázové platformy (MSSQL a Oracle).

Pro vizuální GUI testy:

- příprava dávky pro spuštění více testů najednou,
- automatické spouštění po vygenerování verze,
- automatické spouštění po sestavení verze na Jenkins,
- přidávání dílčích testů,
- refaktorizace kódu testů,
- paralelizace spouštění předzveřejňovací sady testů,
- zdokonalená kontrola hodnot z GUI,
- testy CH/DE verze,
- řešení špatné odezvy.

Automatické vizuální testy nových webových aplikací komunikujících prostřednictvím WebAPI. Vytvoření automatických Python WebAPI testů.

Závěr

Být agilní znamená být hbitý, flexibilní a umět téměř okamžitě reagovat na změnu. Na těchto principech stojí agilní metody. Využívání agilních metod přináší velkou míru zaměření na průběžný sběr zpětné zákaznické vazby s možností pružné reakce na neplánované ale potřebné změny. Což vede ke spokojenosti všech zúčastněných stran.

Na základě získaných poznatků a zkušeností lze konstatovat, že agilní metody řízení vývoje softwaru jsou tomuto prostředí jako šité na míru. Čas věnovaný rozvoji a zkoumání možností automatizace se v případě dovedení myšlenky do konce vždy vyplatí a ve výsledku přinese velmi výraznou časovou úsporu.

V této práci se autor pokusil poskytnout náhled na zavádění agilního způsobu vedení a automatizovaného testování, a to z pohledu, s touto problematikou nezkušeného vývojáře. Popis postupného nástupu jednotlivých událostí, tak, jak je vnímal během procesu zavádění. Zároveň chtěl předat informace ze zákulisí tvorby a implementace těchto skutečností v zavedené softwarové společnosti, aniž by prozradil nějaké obchodní tajemství. Ale tak, aby téma bylo pro čtenáře zajímavé a poutavé.

Za přínos této diplomové práce může být považována prezentace postupných a dlouhodobých změn na vývojovém oddělení spolu s možností náhledu na možnosti využití automatizace ve vývoji softwaru. Autor prostřednictvím práce a přiložených materiálů poskytuje detailní pohled na průběh zavedení nových způsobů vedení a možností využití automatizace.

V této závěrečné práci jsou postupně vysvětleny základní pojmy týkající se agilního způsobu vedení vývoje, zejména metodiky Scrum a jejích nástrojů. Dále se autor věnoval popisu možností automatizovaného testování s využitím různých prostředků. Součástí práce jsou ukázkové zdrojové kódy. Je zajímavé porovnávat historicky první test, napsaný nezkušeným vývojářem, s testy, které byly podrobeny dlouholetému vývoji zkušených programátorů. V podstatě se dá tvrdit, že programování je jako hraní s lego stavebnicí. Díky využití metodiky Scrum bylo potvrzeno, že každý člen týmu je obrovským přínosem pro celý vývoj.

Seznam použité literatury

ABRA Help: Nápověda ABRA Gen [online]. Praha, 2022 [cit. 2022-01-11]. Dostupné z: https://help.abra.eu/

ABRA Software: Informační systém a ERP pro každou firmu | ABRA Software [online]. Praha: Abra, 2022 [cit. 2022-01-11]. Dostupné z: https://www.abra.eu/

AutoIt Scripting Language: Overview [online]. AutoIt Consulting, 2019 [cit. 2022-01-08]. Dostupné z: https://www.autoitscript.com/site/autoit/

BASL, Josef a Roman BLAŽÍČEK. 2012. *Podnikové informační systémy: Podnik v*

informační společnosti. 3., aktualizované a doplněné vydání. Praha: Grada

Publishing, a.s., 328 s. ISBN 978-80-247-4307-3.

BEDNÁROVÁ, Dagmar, Gabriela BOGDANOVSKÁ, Andrea MOJŽIŠOVÁ a Jana PÓCSOVÁ. Implementation of Agile Methodologies in an Engineering Course. *Education Sciences* [online]. MDPI and ACS Style, 2020, 2020(10), 19 [cit. 2022-01-08]. Dostupné z: doi:10.3390/educsi10110333

BRUCKNER, Tomáš, Alena BUCHALCEVOVÁ a Jiří VOŘÍŠEK. 2012. *Tvorba*

informačních systémů: Principy, metodiky, architektury. 1. vydání. Praha: Grada

Publishing, a.s., 360 s. ISBN 978-80-247-4153-6.

Bugzilla: What is Bugzilla? [online]. bugzilla.org, 2022 [cit. 2022-01-09]. Dostupné z: https://www.bugzilla.org/about/

CANTU, Marco, 2016. *Object Pascal Handbook*. Piacenza, Italy: Marco Cantu. ISBN 978-1514349946.

DUnit Overview. In: *Embarcadero/IDERA Documentation Wiki* [online]. RAD Studio, 2022 [cit. 2022-01-08]. Dostupné z: https://docwiki.embarcadero.com/

FastScript: FastScript je víceplatformní vícejazykový skriptovací stroj. Je užitečný pro programátory, kteří chtějí svým projektům dodat možnost skriptování. [online]. Fast Reports, 2022 [cit. 2022-01-09]. Dostupné z: https://www.fast-report.com/cz/product/fast-script/

FLESNER, Andy. *AutoItv3:Your QuickGuide* [online]. O'Reilly Media, 2007 [cit. 2022-01-08]. ISBN 978-0-596-51512-6.

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

PROQUEST. 2019 *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. Dostupné z: http://knihovna.tul.cz

SCHWABER, Ken a Jeff SUTHERLAND. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game* [online]. V2020. 2020 [cit. 2022-01-08]. Dostupné z: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

SciTE4AutoIt3: Introduction [online]. CHM Decoder autoitscript.com, 2022 [cit. 2022-01-08]. Dostupné z: https://www.autoitscript.com/autoit3/scite/docs/SciTE4AutoIt3.html

Scrum.org: Welcome to the Home of Scrum!™ [online]. Scrum.org, 2022 [cit. 2022-01-08]. Dostupné z: https://www.scrum.org/

ŠOCHOVÁ, Zuzana a Eduard KUNCE, 2019. *Agilní metody řízení projektů*. 2. vydání. Brno: Computer Press. ISBN 978-80-251-4961-4.

Seznam příloh

Příloha A Obsah přiloženého paměťového média

- diplomová práce ve formátu .pdf:
  - svarc_diplomova_prace.pdf,
- zdrojové kódy skriptingových testů:
  - sale.price,
  - sale.price.setcompanyparameters.pas,
  - receivedorder.rowdiscount.pas,
- zdrojové kódy základních pomocných knihoven:
  - base.lib.pas,
  - base.lib.companyparameters.pas,
  - base.lib.events.pas,
  - base.lib_businessobject.pas,
  - base.lib_const.pas,
  - base.lib_checkvalue.pas,
  - base.lib_necessary.pas,
  - base.lib_parameters.pas,
  - base.lib_roll.pas,
  - base.lib_sql.pas,
  - base.lib_store_basic_const.pas,
  - base.lib_stores,
  - events.lib,
  - log.tcpip.calls,
  - profiler.lib,
- zdrojové kódy ukázkového GUI testu:
  - base.lib.AbraGUITool.au3,
  - base.lib.Activate&Waits.au3,
  - base.lib.Agenda.au3,
  - base.lib.DateTime.au3,
  - base.lib.FileManagement.au3,
  - base.lib.Paths.au3,
  - base.lib.Start.au3,
  - profilertest.au3,

- kolekce Postman API testů:
    - API_tests.postman_collection.json,
- videonahrávka průběhu vizuálního GUI předzveřejňovcího testu:
    - GUI_test.mp4.