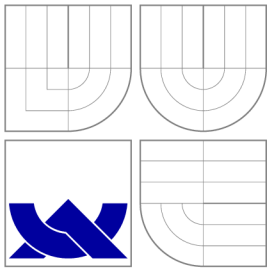
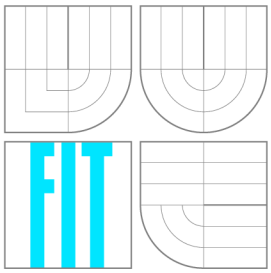


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

METODY A METRIKY PRO MĚŘENÍ POUŽITELNOSTI SOFTWARE

METHODS AND METRICS FOR SOFTWARE USABILITY MEASUREMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETRA STEHLÍKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2015

Abstrakt

Práce se zabývá kvalitou softwarových produktů se zaměřením na použitelnost těchto produktů a způsoby jejího měření. Práce shrnuje definice kvality z různých zdrojů a modely kvality dle dostupných standardů. Dále podrobněji popisuje použitelnost a metody a metriky pro její měření a analyzuje současnou situaci v této oblasti. V rámci této práce byl navržen a implementován nástroj pro měření použitelnosti desktopového software.

Abstract

The master's thesis deals with the quality of software products focusing on usability of these products and methods of its measurement. The thesis resumes the definitions of quality from different sources and quality models according to available standards. It also describes usability in detail and methods and metrics for its measurement and analyses current situation in this field. In the thesis the tool for measuring usability of desktop software products was designed and implemented.

Klíčová slova

kvalita softwarového produktu, použitelnost, měření kvality, měření použitelnosti, metriky, nástroj pro měření použitelnosti software

Keywords

software product quality, usability, quality measurement, usability maesurement, metrics, tool for measuring software usability

Citace

Petra Stehlíková: Metody a metriky pro měření použitelnosti software, diplomová práce, Brno, FIT VUT v Brně, 2015

Metody a metriky pro měření použitelnosti software

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Další informace mi poskytli Ing. Martin Minařík, Ph.D. a Ing. Jan Verner ze společnosti Siemens, Corporate Technology Brno. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Petra Stehlíková
27. května 2015

Poděkování

Děkuji doc. RNDr. Jitce Kreslíkové, CSc. za vedení mé práce a cenné rady při hledání informací a Ing. Janu Vernerovi a Ing. Martinu Minaříkovi, Ph.D. za návrh zajímavého tématu práce a za rady a pomoc při návrhu a implementaci nástroje, který je předmětem práce.

Obsah

Kapitola 1

Úvod

Kvalita je v současné době důležitou charakteristikou jakéhokoliv produktu, softwarové produkty nevyjímaje. Stále častěji a intenzivněji je do procesu vývoje softwarových produktů zapojován zákazník, resp. uživatel výsledného produktu a jeho subjektivní názor na produkt a práci s ním je brán jako významný ukazatel kvality produktu. Pro vývojové týmy i jejich vedení je důležitá spokojenost zákazníka, která pramení mimo jiné z dobré použitelnosti produktu a zákazníkovy pozitivní zkušenosti.

Aby mohla být zlepšována použitelnost softwarových produktů a tím zvyšována spokojenost zákazníků, je třeba použitelnost produktů měřit a testovat, výsledky těchto měření analyzovat a podle nich produkty upravovat. Je třeba zjistit, jak na zákazníka působí uživatelské rozhraní, jak rychle se zákazník naučí produkt používat a co mu používání produktu znesnadňuje. Velmi často se jedná o subjektivní charakteristiky, které lze exaktně měřit jen obtížně. Je však možné měřit či sledovat dílčí informace o používání softwaru uživateli a na jejich základě vyvodit s pomocí názorů a komentářů uživatelů závěry o použitelnosti produktu a návrhy na její zlepšení.

Cílem této diplomové práce je navrhnout a implementovat nástroj pro měření použitelnosti softwaru pro oddělení Corporate Technology společnosti Siemens v Brně, který umožní sběr těchto informací a jejich následné využití pro zlepšování použitelnosti produktů tohoto oddělení.

Kapitola 2 shrnuje definice kvality z dostupných standardů a literatury zabývající se kvalitou softwaru. Dále srovnává různé modely kvality a představuje základní principy měření a zlepšování kvality softwaru. V kapitole 3 se práce zabývá jednou z charakteristik kvality softwaru – použitelností. Definuje použitelnost a termíny s ní související a představuje způsoby, jakými lze použitelnost měřit a testovat.

V kapitole 4 jsou analyzovány aktuálně dostupné nástroje pro měření a sledování softwarových produktů se zaměřením na sledování desktopových aplikací, pro které je v kapitole 5 navržen nástroj pro měření použitelnosti. Kapitola 6 popisuje implementaci nástroje a kapitola 7 jeho testování.

Kapitoly 2 (Kvalita), 3 (Použitelnost) a 4 (Analýza existujících řešení) byly převzaty ze semestrálního projektu, přičemž kapitoly o použitelnosti a existujících řešeních byly doplněny o další informace. Kapitola 5 (Analýza a návrh řešení) byla v části o klientské části nástroje upravena, část o serverové části nástroje byla rozšířena a doplněna o podrobnější návrh nástroje.

Kapitola 2

Kvalita

Tato kapitola se zabývá kvalitou produktu v obecném pojetí i se zaměřením na softwarové produkty. V úvodu shrnuje různé definice kvality obsažené v dostupných normách a odborné literatuře zabývající se kvalitou. V kapitole 2.2 jsou nastíněny modely kvality softwarových produktů, modely dle norem ISO/IEC 25010 a ČSN ISO/IEC 9126-1 a model FURPS, resp. rozšířený model FURPS+ vytvořený společností Hewlett-Packard.

2.1 Definice kvality

Definice kvality se napříč literaturou různí, všechny se však shodují v tom, že kvalita je souborem jistých požadavků, které by měl výsledný produkt splňovat.

Garvin [17] popisuje pět současně existujících pohledů na definici kvality:

1. transcendentální, kdy je na kvalitu nahlíženo jako na něco, co lze rozpoznat, ale jen obtížně definovat [29],
2. produktový, kdy je kvalita posuzována jako množství žádoucích vlastností,
3. založený na uživateli, kdy je kvalitou myšlena vhodnost produktu k zamýšlenému použití [29],
4. založený na výrobě, kdy jde o shodu vlastností výsledného produktu se specifikací
5. a založený na hodnotě, kdy kvalita závisí na tom, kolik je zákazník ochoten za ni zaplatit [29], jde o srovnání ceny a spokojenosti zákazníka.

Dahlgaard, Kristensen a Kanji [12] posuzují kvalitu jako vícedimenzionální filozofii, kterou lze shrnout jako „dělání věcí správně“ za účelem vysoké konkurenceschopnosti a zisku.

Norma ČSN EN ISO 9000 definuje kvalitu jako stupeň splnění požadavků souborem inherentních[1] charakteristik. Kvalita je tedy definována souborem požadavků, které mohou být definovány kteroukoliv zainteresovanou stranou. Stupeň splnění požadavku pak znamená, že požadavek musí být měřitelný, a tudíž musí existovat jednoznačná metodika jeho měření. [28]

Norma ISO 8402:1994 definuje kvalitu jako celkový souhrn charakteristik entity, které ovlivňují její schopnost uspokojovat stanovené a předpokládané potřeby [2].

[1] vnitřně obsažených

Norma ISO/IEC 25010 definuje kvalitu softwaru jako stupeň, do jakého softwarový produkt uspokojuje stanovené a předpokládané potřeby, pokud je používán za specifikovaných podmínek [4].

2.2 Model kvality softwarového produktu

V současnosti je používáno několik modelů kvality softwarových produktů, např. jednodušší FURPS+ nebo složitější a podrobněji definované modely popsané mezinárodními standardy. Všechny tyto modely vymezují celou řadu charakteristik kvality a tyto charakteristiky sdružují do několika skupin.

2.2.1 Model kvality FURPS a FURPS+

Model kvality FURPS byl vytvořen společností Hewlett-Packard v 70. letech 20. století [21]. Jeho název je akronymem anglických názvů pěti vlastností produktu, ze kterých je tento model složen [41]:

- funkčnost (Functionality),
- použitelnost (Usability),
- spolehlivost (Realiablity),
- výkonnost (Performance)
- a podporovatelnost (Supportability).

Každý z těchto kvalitativních faktorů je dále rozdělen na dílčí charakteristiky. Toto rozdělení je znázorněno na obrázku 2.1.

Model FURPS je také používán softwarovými inženýry jako seznam kvalitativních faktorů, které je třeba prodiskutovat se zainteresovanými stranami za účelem definice kvalitativních požadavků na výsledný produkt [41].

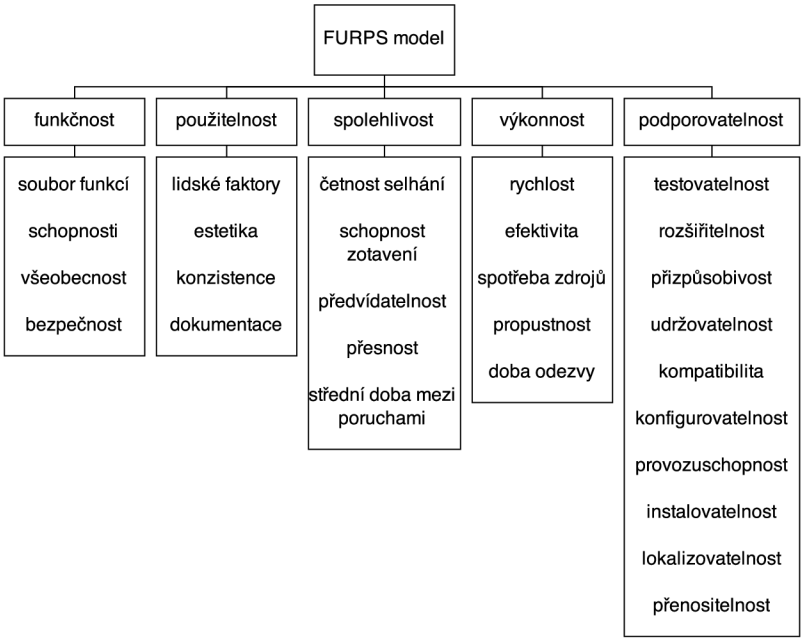
Rozšířený model kvality FURPS+ definuje navíc tyto čtyři charakteristiky [15]:

- požadavky na návrh (Design requirements),
- požadavky na implementaci (Implementation requirements),
- požadavky na rozhraní (Interface requirements)
- a požadavky na fyzické vlastnosti (Physical requirements).

2.2.2 Model kvality dle norem ISO/IEC 25010 a ČSN ISO/IEC 9126-1

Na základě modelu kvality FURPS popsaného v předchozí kapitole byl vytvořen standard ISO/IEC 9126 z roku 1991 [21], který byl v roce 2001 přepracován do standardu ISO/IEC 9126-1 a tento v roce 2011 nahrazen standardem ISO/IEC 25010, který je součástí řady standardů s názvem Software product Quality Requirements and Evaluation (SQuaRE).

Model kvality softwarového produktu se dle normy ISO/IEC 25010 skládá z modelu kvality produktu a z modelu kvality při používání [4]. Starší norma ČSN ISO/IEC 9126-1



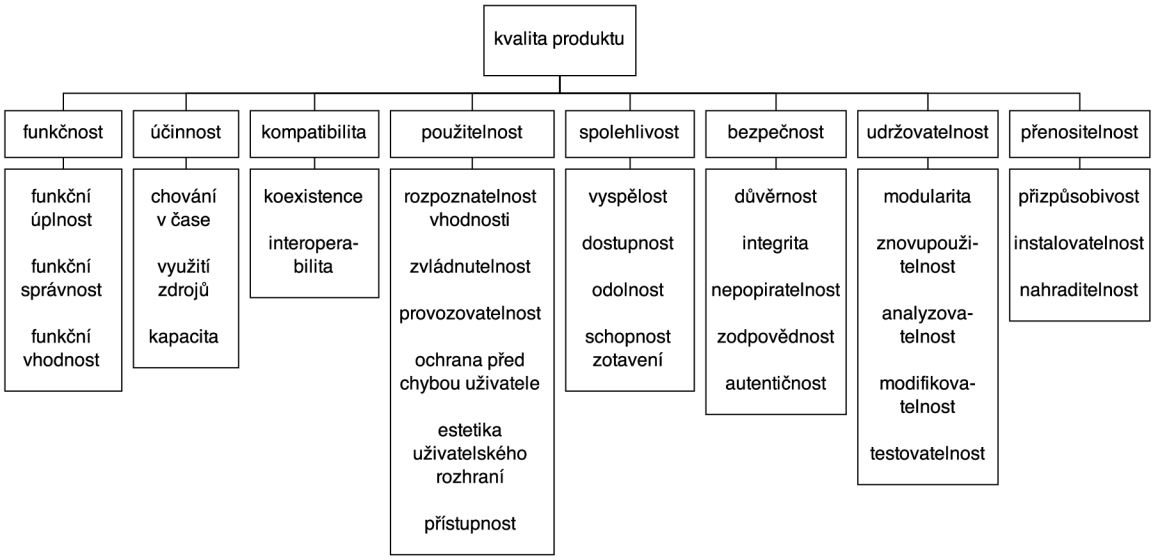
Obrázek 2.1: Model kvality FURPS, vytvořeno dle [41]

rozlišovala navíc vnitřní a vnější kvalitu v rámci modelu kvality produktu, přičemž vnitřní kvalitu definovala jako souhrn charakteristik nespustitelného programu a vnější kvalitu jako kvalitu softwaru vykonávajícího svoji činnost [2].

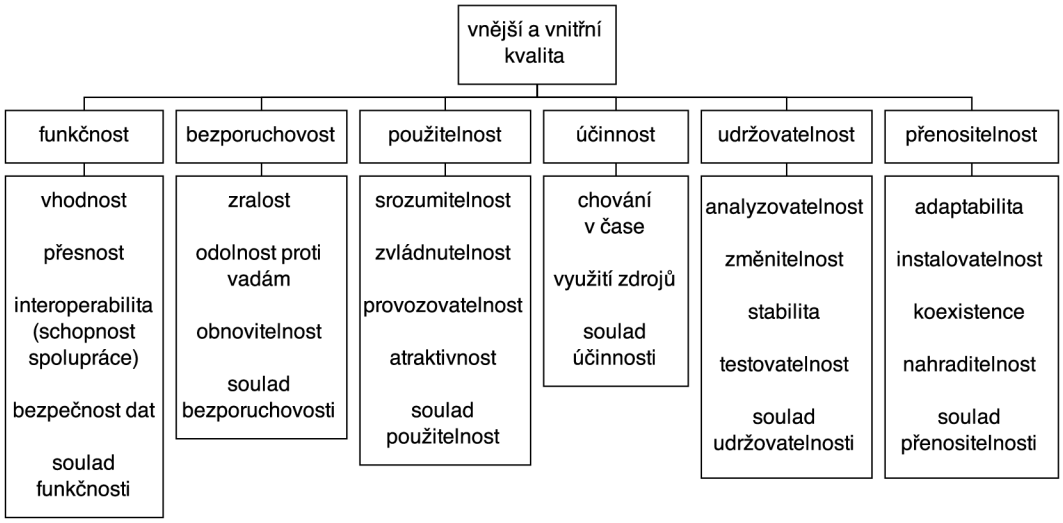
Obě normy specifikují model kvality produktu, resp. model vnitřní a vnější kvality jako souhrn charakteristik a subcharakteristik. Tyto vlastnosti jsou shrnuty na obrázcích 2.2 (norma ISO/IEC 25010) a 2.3 (norma ČSN ISO/IEC 9126-1). Charakteristiky a subcharakteristiky kvality při používání dle těchto norem jsou pak zobrazeny na obrázcích 2.4 a 2.5.

2.3 Model zlepšování kvality

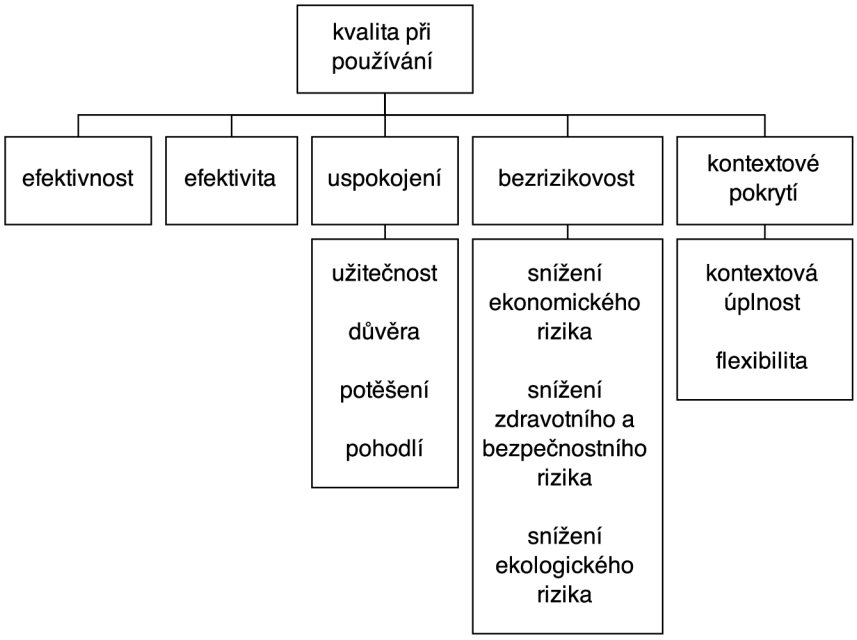
Základem řízení kvality v kterémkoliv odvětví je iterativní proces neustálého zlepšování. Nejznámějším modelem tohoto procesu je model PDCA, jehož název je zkratkou pojmenování jeho jednotlivých fází: *Plan* (plánuj), *Do* (udělej), *Check* (zkontroluj) a *Act* (jednej). V první fázi (*Plan*) je identifikováno potenciální zlepšení, analyzován současný stav a naplánován způsob zlepšení. Ve druhé fázi (*Do*) je toto zlepšení realizováno, v rámci třetí fáze (*Check*) vyhodnoceno a v poslední fázi (*Act*) je pak představeno konečné řešení zlepšení. [41] Jednotlivé fáze a jejich návaznosti jsou znázorněny na obrázku 2.6.



Obrázek 2.2: Model kvality produktu dle normy ISO/IEC 25010, vytvořeno dle [4]

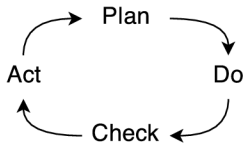


Obrázek 2.3: Model vnitřní a vnější kvality produktu dle normy ČSN ISO/IEC 9126-1, vytvořeno dle [2]



Obrázek 2.4: Model kvality při používání dle normy ISO/IEC 25010, vytvořeno dle [4]

Obrázek 2.5: Model kvality při používání dle normy ČSN ISO/IEC 9126-1, vytvořeno dle [2]



Obrázek 2.6: Model zlepšování kvality PDCA, vytvořeno dle [41]

2.4 Měření kvality softwaru

Kvalitu softwaru lze měřit na základě stanovených metrik. Metriku definuje norma ČSN ISO/IEC 9126-1 jako definovanou metodu měření a stupnici měření [2]. Metriky se dělí do dvou kategorií – tvrdé metriky a měkké metriky. Tvrdé metriky jsou založeny na přesných, objektivních hodnotách, zatímco měkké metriky jsou založeny na subjektivních hodnoceních. [29]

Z modelů kvality softwarového produktu uvedených v kapitole 2.2 je zřejmé, že kvalitu softwaru je třeba hodnotit z mnoha různých pohledů, podle řady charakteristik. Atributů, které je nutné sledovat a měřit pro zajištění vysoké kvality softwarového produktu, je velké množství.

Při každém měření kvality je nutné postupovat stejným způsobem. Tento postup popisuje Abran [6] v následujících třech krocích:

1. Před samotným měřením je třeba buď vybrat existující metodu měření, nebo vytvořit novou, pokud žádná z dostupných metod neodpovídá potřebám měření.

2. Jakmile je vybrána nebo vytvořena metoda měření, je třeba aplikovat její pravidla na softwarový produkt nebo jeho část pro získání specifických výsledků měření.

3. Nakonec jsou výsledky měření převedeny na kvantitativní nebo kvalitativní model obvykle v kombinaci s jinými výsledky měření jiného druhu.

Kapitola 3

Použitelnost

Tato kapitola se zabývá tématem použitelnosti, uživatelskou zkušeností a návrhem soustředěným na člověka, které s použitelností úzce souvisejí. Použitelnost je jedním z aspektů uživatelské zkušenosti a jedním z cílů návrhu zaměřeného na člověka. Kapitola dále představuje způsoby, jimiž je možné měřit použitelnost, uvádí příklady jednotlivých typů metrik a shrnuje aktuální trendy v oblasti měření a testování použitelnosti.

3.1 Uživatelská zkušenost

Uživatelská zkušenost (anglicky *User Experince*, zkráceně *UX*) se soustředí na získání hlubokého porozumění uživatelům produktu, jejich potřeb, schopností a omezení. Zároveň bere v potaz obchodní cíle vedoucích projektu. Osvědčené postupy UX podporují zvyšování kvality interakce uživatele s produktem. [39]

Podle Morvilla [24] sestává zkušenost uživatele ze sedmi aspektů zobrazovaných Morvillem v podobě plástve medu (anglicky *honeycomb*) jako na obrázku 3.1.

Obrázek 3.1: User Experience Honeycomb, vytvořeno dle [24]

Morvill vysvětluje jednotlivé aspekty takto:

- užitečnost – aplikace znalosti a dovednosti k vytvoření inovativních, užitečnějších řešení,
- použitelnost – jednoduchost použití,
- žádostivost – pochopení síly a hodnoty obrazu, identity, značky a dalších prvků emocionálního designu,
- zjistitelnost – návrh navigovatelného obsahu a lokalizovatelných objektů,
- přístupnost – přístupnost produktu a jeho obsahu lidem s hendikepem,
- důvěryhodnost – důvěra uživatelů v produkt a jeho obsah,
- hodnotnost – přínos hodnoty.

Definice uživatelské zkušenosti je také součástí normy ISO/IEC TR 25060. Ta ji popisuje jako pohled a reakci osoby na používání anebo očekávané používání produktu [3].

3.2 Návrh soustředěný na člověka

Návrh soustředěný na člověka (anglicky *human-centered design* nebo *user-centered design*[1]) může být pokládán za postup, rámec, filozofii či metodu návrhu nástrojů určených pro člověka zapojující jej do procesu návrhu. Uživatel (nebo jiná zainteresovaná osoba) nevytváří sám návrh, ale odborník zabývající se návrhem soustředěným na člověka bere při rozhodování o návrhu v potaz jeho profil, chování a preference. [42]

Norma ISO 9241-210 definuje návrh soustředěný na člověka jako přístup k návrhu a vývoji produktů, který usiluje o zvýšení jejich použitelnosti aplikací lidských faktorů a znalostí o použitelnosti [1].

Návrh soustředěný na člověka může být vnímán jako součást celé filozofie uživatelské zkušenosti, a to jako proces sloužící k dosažení dobré zkušenosti uživatele. [31]

3.3 Použitelnost

Použitelnost vyjadřuje obecně kvalitu zkušenosti uživatele při práci s produktem [38], přičemž produkt dělá použitelným absence frustrace uživatele při jeho používání [30].

Nielsen [25] definuje použitelnost softwarových produktů jako kvalitativní vlastnost produktu, která určuje, jak jednoduché je použití jeho uživatelského rozhraní. Jedná se o vlastnost produktu, která je složena z pěti kvalitativních komponent:

- zvládnutelnosti, která vyjadřuje, jak snadné je pro uživatele splnit základní úkoly při první interakci s produktem [25],
- efektivity, která představuje rychlost, s jakou může uživatel přesně a kompletně dosáhnout svých cílů [30],

[1] Standard ISO 9241-210 doporučuje používat termín *human-centered design*, protože se zabývá dopady na celou řadu zainteresovaných stran, nikoliv pouze na uživatele. V praxi se však oba termíny používají jako synonyma. [1]

- schopnosti zapamatovat si, která vyjadřuje, jak snadno si uživatelé vybaví získané dovednosti po období, v němž produkt nepoužívali [25],
- četnosti a vážnosti chyb uživatele, která sleduje, jak často uživatelé dělají při používání produktu chyby, jak vážné tyto chyby jsou a jak se z nich uživatelé zotavují [25]
- a subjektivní spokojenosti, která odkazuje na dojmy, pocity a názory uživatele na produkt [30].

Rubin a Chisnell [30] uvádějí jako součást použitelnosti také efektivnost a přístupnost. Efektivnost vyjadřuje rozsah, v jakém se produkt chová tak, jak uživatel předpokládal, a jednoduchost, s jakou jej použije zamýšleným způsobem. Přístupnost vyjadřuje použitelnost produktu pro osoby s hendikepem, přičemž dobrá přístupnost může zlepšit použitelnost i pro uživatele bez hendikepu při používání produktu v nevyhovujícím prostředí.

Jak již bylo zmíněno v kapitole 2.2.2, použitelnost je jednou z charakteristik modelu kvality produktu dle normy ISO/IEC 25010. Tato norma definuje použitelnost jako stupeň, do jakého může být produkt používán specifikovanými uživateli k dosažení specifikovaných cílů [4]. Její subcharakteristiky jsou:

- rozpoznatelnost vhodnosti, která určuje, do jaké míry je uživatel schopen rozpoznat, zda je produkt vhodný k uspokojení jeho potřeb,
- zvládnutelnost, která značí schopnost uživatele naučit se produkt rychle používat,
- provozovatelnost, která určuje snadnost provozu a řízení produktu,
- ochrana před chybou uživatele,
- estetika uživatelského rozhraní
- a přístupnost, která určuje míru, do jaké je produkt možné používat uživateli s širokou škálou vlastností a schopností.

S použitelností softwarového produktu také úzce souvisí kvalita při používání, kterou norma ISO/IEC 25010 uvádí jako součást kvality. Charakteristiky a subcharakteristiky uvedené na obrázku 2.4 mají tento význam:

- efektivnost – správnost a kompletnost, se kterými uživatel dosáhne specifikovaných cílů,
- efektivita – zdroje vynaložené na správnost a kompletnost uživatelem dosažených cílů,
- uspokojení – spokojenost uživatele s produktem, pokud je tento používán za specifikovaných podmínek,
  - užitečnost – spokojenost uživatele s dosaženými cíli, jejich výsledky a důsledky,
  - důvěra – důvěra uživatele, že se produkt bude chovat, jak bylo zamýšleno,
  - potěšení – potěšení uživatele ze splnění jeho osobních potřeb,
  - pohodlí – fyzické pohodlí uživatele,
- bezrizikovost – snížení potenciálního rizika pro ekonomiku, lidský život, zdraví a životní prostředí,
- kontextové pokrytí – produkt může být používán jak ve specifikovaném kontextu (kontextová úplnost), tak v kontextu širším (flexibilita).

3.4 Měření použitelnosti

Pro měření použitelnosti existuje velké množství metrik a metod. Ve skutečnosti však tyto metriky zkoumají, jak nepoužitelný produkt je[2] – lze kvantitativně měřit, kolik problémů uživatelé mají při používání produktu, jaké jsou tyto problémy a proč se vyskytují [30].

Jednotlivé komponenty použitelnosti, jak byly popsány v kapitole 3.3, je možné měřit nebo zkoumat pomocí různých metod. Některé (např. subjektivní spokojenost uživatele) je třeba zkoumat prostřednictvím kvalitativních metod, protože je velmi obtížné je kvantitativně ohodnotit. Další (např. efektivitu) lze měřit pomocí kvantitativních metrik. V praxi jsou oba tyto přístupy kombinovány pro zajištění co nejlepších výstupů testování použitelnosti aplikace a jejího následného zlepšení.

Pro všechny metody platí, že účastníci testování by měli být nejlépe koncoví uživatelé produktu, kteří mají určité vzdělání, zkušenosti a znalosti z oboru, ve kterém bude produkt používán.

3.4.1 Kvalitativní metody

Kvalitativní testy jsou nejrozšířenější formou testování použitelnosti. Jsou založené na analýze chování uživatele a jejich výsledkem není žádná statistika, ale nálezy, které mohou pomoci zlepšit návrh produktu a jeho použitelnost. [26] Tyto metody jsou většinou neformální a účastní se jich malý počet uživatelů. [22]

Pozorování

Pozorování je nejčastěji uváděným způsobem kvalitativního testování použitelnosti.

V průběhu testování pozorovatel komunikuje osobně s uživateli, zadává jim úkoly, a sleduje jejich počínání. Uživatel je sledován vývojovým týmem a případně dalšími zainteresovanými stranami. Po ukončení testování se koná porada, kde pozorovatelé porovnají své poznámky a postřehy a rozhodnou, které problémy a jak řešit. [22]

Průběh pozorování se může lišit v tom, zda pozorovatel pouze sleduje práci uživatele nebo zda jej žádá, aby mluvil o svých pocitech a názorech (tzv. přemýšlel nahlas). Je možné také nechat pracovat s produktem dva uživatele současně a sledovat jejich dialog při plnění úkolů. [26]

Další odlišnosti se mohou objevit ve způsobu zaznamenávání testování. Nejjednodušším způsobem je psaní poznámek na papír, dále je možné pořizovat zvukový záznam testování nebo jeho videozáznam. Ty je možné živě přenášet do vedlejší místnosti, kde může testování sledovat vývojový tým a další zainteresované osoby. [26]

Rozhovory

Jinou možností, jak zjistit názory uživatelů na produkt, je vést s nimi rozhovor. Tento způsob je vhodný pro sledování specifické problematiky a často vede k získání konstruktivních návrhů na zlepšení použitelnosti produktu. Nevýhodou této metody je subjektivnost výpovědí uživatelů, velká časová náročnost a možná zaujatost člověka, který rozhovor s uživatelem vede. [26]

[2] I přes tento fakt budou v dalším textu metriky označovány jako metriky pro měření či testování použitelnosti, nikoliv nepoužitelnosti. Jedná se o zavedený termín používaný v literatuře zabývající se touto problematikou.

Dotazníky

Pomocí dotazníků je možné oslovit mnohem větší skupinu uživatelů než v předchozích dvou metodách. Problémem dotazníků je velká závislost kvality získaných výsledků na kvalitě položených otázek. Tomuto problému je možné se vyhnout, a to použitím některého z již vytvořených a průzkumy a praxí ověřených dotazníků. Dva z nich jsou uvedeny níže v této části jako příklady. Výhodou této metody zjišťování názorů uživatelů je, že vyplňování dotazníků nemusí být přítomný jejich hodnotitel a také to, že výsledky dotazníků mohou být kvantifikovány [26].

Příkladem takového dotazníku, jehož výsledky je možné kvantifikovat, je *System Usability Scale* (zkráceně *SUS*) [8], který se řadí mezi dotazníky používané po ukončení testování. Jedná se o jednoduchý dotazník s deseti výroky, u kterých respondent hodnotí svůj souhlas či nesouhlas na pětibodové stupnici od „silně nesouhlasím“ po „silně souhlasím“. Tyto výroky jsou následující a jsou uváděny vždy v tomto pořadí:

1. Myslím, že bych chtěl tento systém používat často.
2. Shledal jsem systém zbytečně komplexní.
3. Myslím, že systém byl jednoduchý na použití.
4. Myslím, že bych potřeboval podporu odborné osoby, abych byl schopný systém používat.
5. Shledal jsem různé funkce v systému dobře integrované.
6. Myslím, že v systému bylo příliš nekonzistencí.
7. Dovedu si představit, že většina lidí se systém naučí používat velmi rychle.
8. Shledal jsem systém velmi nepohodlný pro použití.
9. Cítil jsem se velmi sebejistě při používání systému.
10. Musel jsem se naučit spoustu věcí, než jsem se seznámil se systémem.

Stupnice, na níž respondent uvádí svůj názor, je ohodnocena čísly od 0 („silně nesouhlasím“) do 4 („silně souhlasím“). Vyhodnocení pak probíhá odlišně pro výroky s lichým pořadím a pro výroky se sudým pořadím:

$$\text{skóre lichých výroků} = \text{odpověď respondenta} - 1$$

$$\text{skóre sudých výroků} = 5 - \text{odpověď respondenta}$$

Skóre jednotlivých výroků se pak sečtou a součet se vynásobí koeficientem $2,5$ pro získání celkové hodnoty použitelnosti systému, která je v rozmezí hodnot 0 až 100.

Výzkumem byla stanovena průměrná hranice použitelnosti systému na hodnotu 68 [40], tedy systémy s vyšším skóre lze hodnotit jako nadprůměrné a naopak s nižším skóre jako podprůměrné.

Dalším příkladem je jednoduchý dotazník *Single Ease Question* (zkráceně *SEQ*), který uživatel vyplňuje v průběhu testování vždy po dokončení jednoho úkolu. Sestává z jediné položky, a to hodnocení složitosti úkolu na pětibodové nebo sedmibodové stupnici od „velmi jednoduchý“ po „velmi složitý“. [35]

3.4.2 Kvantitativní metody

Cílem kvantitativních metod je dokázat prostřednictvím měření určité hypotézy o použitelnosti produktu (např. úspěšnost uživatelů při plnění určitého úkolu nebo čas potřebný pro jeho splnění). Pro tyto metody je nutné přesně definovat testovací protokol a důsledně se ho držet při testování s každým jednotlivým uživatelem a zároveň mít těchto uživatelů dostatek, aby byly závěry statisticky významné. Kvantitativní metody také vyžadují pečlivý sběr a analýzu dat a minimální interakci s účastníky testování, aby nedocházelo k ovlivnění výsledků. [22]

Mezi kvantitativní metody měření použitelnosti patří například měření úspěšnosti při plnění úkolů, času, který uživatelé pro plnění úkolů potřebují, počítání a analýza chyb, jichž se uživatelé dopouštějí, kliknutí na prvky grafického uživatelského rozhraní, zobrazení stránek webové aplikace či zjišťování úrovně obtížnosti zadaných úkolů. [32] Některé z těchto metrik jsou dále popsány v následujících podkapitolách.

Úspěšnost

Úspěšnost při plnění úkolů (*success rate*) je základní metrikou pro měření použitelnosti. Obvykle nabývá pouze dvou hodnot: 1 při úspěšném splnění úkolu a 0 při neúspěchu, ale je možné definovat více úrovní splnění úkolu. Výsledkem měření pomocí této metriky pro každý jednotlivý úkol je aritmetický průměr úspěšnosti jednotlivých uživatelů při jeho plnění. [35] Pokud tedy při použití binární varianty metriky úkol splnilo 8 uživatelů z 10, pak úspěšnost je $0,8$. Obvykle se úspěšnost uvádí v procentech, tedy v tomto případě by byla úspěšnost 80%.

Doba plnění úkolu

Doba plnění úkolu (*task time*) je jednoduše čas strávený při nějaké aktivitě a je obvykle reportován jako průměrná hodnota mezi testovanými uživateli. Existuje několik způsobů, jak měřit a analyzovat čas [35]:

- čas potřebný na dokončení úkolu, kdy je započítáván pouze čas uživatelů, kteří úkol splnili úspěšně,
- čas do selhání, kdy je měřen čas od započetí plnění úkolu do okamžiku, kdy uživatel jeho plnění vzdá nebo úkol splní nesprávně,
- celkový čas potřebný na dokončení úkolu, kdy je měřen celkový čas strávený uživateli při plnění úkolu bez ohledu na to, zda jej dokončili úspěšně či neúspěšně.

(výběr vhodného průměru pro reportování → geometrický průměr)

Chyby

Chyby (*errors*) jsou neúmyslné kroky, omyly, chyby či opomenutí, kterých se uživatel dopustí v průběhu plnění úkolu. Je buď jednoduše zaznamenáván počet těchto chyb, nebo jsou chyby analyzovány jako binární metrika, kdy 0 znamená, že uživatel neudělal žádnou chybu, a 1 znamená, že se uživatel nějaké chyby dopustil. [35]

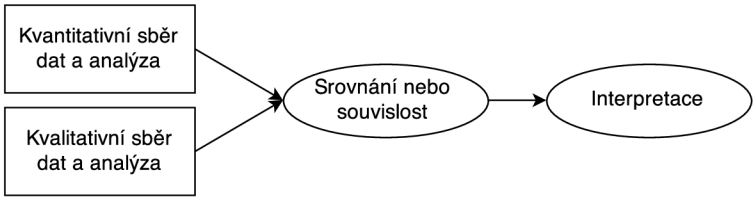
3.4.3 Kombinované metody

Současným trendem v testování použitelnosti softwarových produktů je použití kombinovaných metod, které umožňují sledovat použitelnost produktů z různých úhlů pohledu.

Creswell a Clark [11] popisují následujících šest hlavních způsobů, jak kombinovat kvalitativní a kvantitativní metody měření použitelnosti, přičemž mezi nejvíce používané patří první tři z nich [33]:

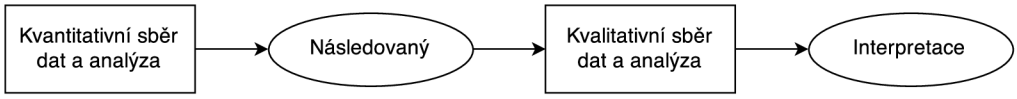
1. konvergentní paralelní návrh kombinace metod,
2. vysvětlující sekvenční návrh kombinace metod,
3. výzkumný sekvenční návrh kombinace metod,
4. vestavný návrh kombinace metod,
5. transformativní návrh kombinace metod
6. a vícefázový návrh kombinace metod.

Při konvergentním paralelním návrhu kombinace metod (viz obrázek 3.2) je provedeno měření použitelnosti zároveň některou kvantitativní a kvalitativní metodou ve stejné fázi tohoto měření, přičemž oběma metodám je přiřazena stejná důležitost a jsou prováděny nezávisle na sobě. Až během souhrnné interpretace jsou slučovány výsledky obou metod.



Obrázek 3.2: Konvergentní paralelní návrh kombinace metod měření použitelnost dle [11]

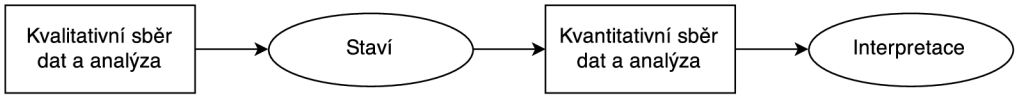
Vysvětlující sekvenční návrh kombinace metod (viz obrázek 3.3) obsahuje dvě oddělené fáze. Začíná se sběrem a analýzou kvantitativních dat, která jsou preferována při zkoumání otázek daného testování. Ve druhé fázi jsou pak sbírána a analyzována kvalitativní data. Tato fáze vychází z výsledků první a pomáhá její výsledky vysvětlit.



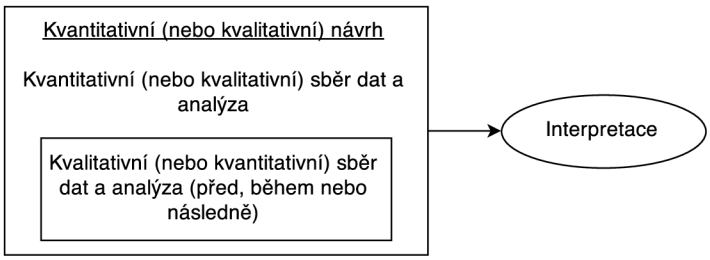
Obrázek 3.3: Vysvětlující sekvenční návrh kombinace metod měření použitelnost dle [11]

Ve výzkumném sekvenčním návrhu kombinovaných metod (viz obrázek 3.4) je nejdříve proveden kvalitativní sběr a analýza dat, ve druhé fázi na něj navazuje kvantitativní, který slouží k ověření nebo zobecnění výsledků první fáze.

Vestavný návrh kombinovaných metod (viz obrázek 3.5) umožňuje použít metodu jednoho typu v rámci metody typu druhého za účelem vylepšení výsledků základní metody. Je



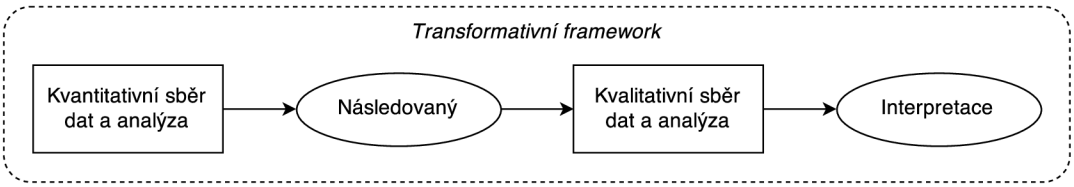
Obrázek 3.4: Výzkumný sekvenční návrh kombinace metod měření použitelnost dle [11]



Obrázek 3.5: Vestavný návrh kombinace metod měření použitelnost dle [11]

tak možné přidat kvalitativní prvek do kvantitativní metody (např. v podobě experimentu) nebo naopak přidat kvantitativní prvek do kvalitativní metody (např. případovou studii).

Při transformativním návrhu kombinovaných metod (viz obrázek 3.6) je výsledná metoda utvářena v rámci transformativního teoretického rámce. Všechna rozhodnutí jsou prováděna v závislosti na kontextu, může být například použita kvantitativní metoda k odhalení určitého jevu a následně kvalitativní metoda k jeho vysvětlení.



Obrázek 3.6: Transformativní návrh kombinace metod měření použitelnost dle [11]

Vícefázový návrh kombinovaných metod (viz obrázek 3.7) kombinuje principy sekvenčního a paralelního návrhu a umožňuje používat různé metody v různých fázích testování.

Obrázek 3.7: Vícefázový návrh kombinace metod měření použitelnost dle [11]

Kapitola 4

Analýza existujících řešení

Na trhu existuje celá řada nástrojů, které je možné využít pro měření statistik softwarových aplikací. Většina z nich je určena pro webové aplikace, najdou se však i takové, které slouží pro sběr informací o desktopových aplikacích, případně takové, jež lze využít pro oba typy aplikací. Tato kapitola představuje možná řešení v obou případech, více se však zaměřuje na oblast desktopových aplikací, kde je dostupných nástrojů méně.

4.1 Webové aplikace

Zřejmě nejznámějším a nejpoužívanějším nástrojem pro sledování webových aplikací je nástroj Google Analytics[1]. Tento nástroj umožňuje sledování velkého množství informací, například:

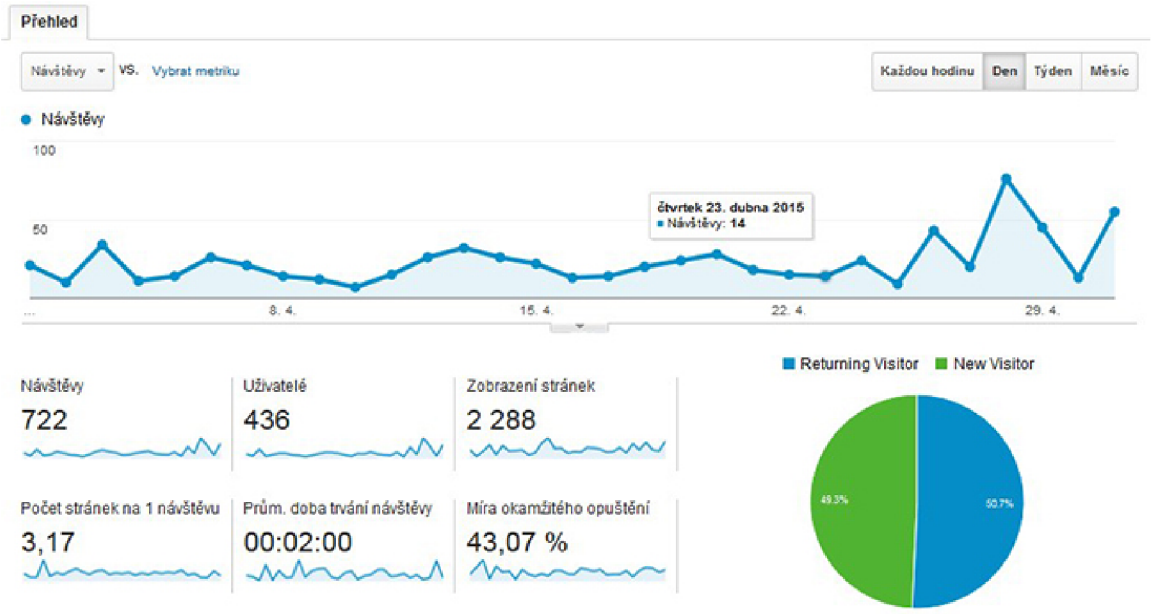
- množství návštěv, unikátních uživatelů a zobrazení stránek,
- množství nových a vracejících se uživatelů,
- geografické informace o uživatelích (jazyk a lokaci),
- dobu trvání návštěvy,
- operační systém a prohlížeč, které uživatelé používají,
- typ zařízení, na kterém je web prohlížen,
- postup uživatelů přes jednotlivé stránky webu,
- odkud uživatelé přišli (vyhledávač, sociální sítě, přímý přístup, e-mail ad.).

Webové rozhraní nástroje Google Analytics pro sledování statistik je na obrázku 4.1.

Společnost Google poskytuje také SDK (*Software Development Kit*) pro použití nástroje Google Analytics pro sledování mobilních aplikací [20] a také API pro sledování desktopových aplikací, které je popsáno v následující kapitole.

K tomuto nástroji existuje celá řada alternativ, například Clicky[2], Piwik[3], Gauges[4], které se liší především jednodušším uživatelským rozhraním, nebo Mixpanel[5], jehož metriky

[1] www.google.com/analytics
[2] www.clicky.com
[3] piwik.org
[4] get.gaug.es
[5] mixpanel.com



Obrázek 4.1: Nástroj Google Analytics

jsou založeny spíše na událostech než na zobrazení stránek, jako je tomu u Google Analytics i ostatních zmíněných nástrojů [9].

Trochu odlišné informace o sledované webové aplikaci umožňuje sbírat nástroj WebRemUsine[6]. Tento nástroj sbírá informace o událostech vznikajících při interakci uživatele s aplikací, jakými jsou například kliknutí na tlačítko, otevření menu, zápis textu do polí formulářů apod. Zároveň nástroj umožňuje sestavit modely chování uživatele (*task models*) a tyto pak porovnávat s nasbíranými daty. Od ostatních se tento nástroj liší také způsobem prezentace dat – využívá k tomu desktopovou aplikaci, všechny výše zmíněné aplikace prezentují data ve webovém rozhraní. [27]

4.2 Desktopové aplikace

Pro sledování desktopových aplikací je možné použít již zmiňovaný nástroj Google Analytics, primárně určený pro webové aplikace. V této kapitole bude nastíněn způsob, kterým jej lze využít pro desktop, a také nevýhody použití tohoto nástroje pro sledování desktopových aplikací. Vedle Google Analytics ale existují i nástroje specializované na sledování desktopových aplikací, mezi které patří například Trackerbird, DeskMetrics nebo StatHat.

4.2.1 Google Analytics pro desktop

Společnost Google umožňuje použít nástroj Google Analytics i pro desktopové aplikace, a to prostřednictvím protokolu Google Analytics Measurement Protocol [18]. Data požadovaná pro sledování aplikace, resp. pro sledování chování uživatelů je třeba zasílat prostřednictvím HTTP požadavků POST nebo GET, přičemž je doporučována metoda POST vzhledem k většímu množství dat, které lze touto metodou zaslat [19].

[6] http://giove.isti.cnr.it/tools/WebRemUSINE/home

Vzhledem k tomu, že nástroj Google Analytics je primárně určen pro webové aplikace, existuje několik nevýhod jeho použití pro aplikace desktopové. Jednou z nich je nutnost mapování každé události na zobrazení stránky a zaslání HTTP požadavku. Toto mapování vyžaduje důkladnou analýzu událostí v aplikaci, které změnu stránky způsobí [16]. Další nevýhodou použití tohoto nástroje je nemožnost sledování jednotlivých sezení. Není tedy možné zkoumat chování konkrétního uživatele při jednom konkrétním sezení, ale pouze obecné tendence při používání aplikace.

4.2.2 Trackerbird

Trackerbird[7] je ekvivalentem nástroje Google Analytics pro desktopové aplikace. Jedná se o komplexní nástroj umožňující sledovat velké množství informací o aplikaci. Metriky poskytované tímto nástrojem jsou [37]:

- množství nových, aktivních a ztracených uživatelů a jejich lokace,
- informace o instalované aplikaci (verze, sestavení, edice, licence ad.),
- informace o počítači, na kterém je aplikace instalována (operační systém, jeho jazyk, informace o monitoru ad.),
- četnost použití jednotlivých funkcí a četnost událostí,
- četnost a doba používání aplikace,
- sledování chyb a výjimek.

Webové rozhraní nástroje Trackerbird je na obrázku 4.2.

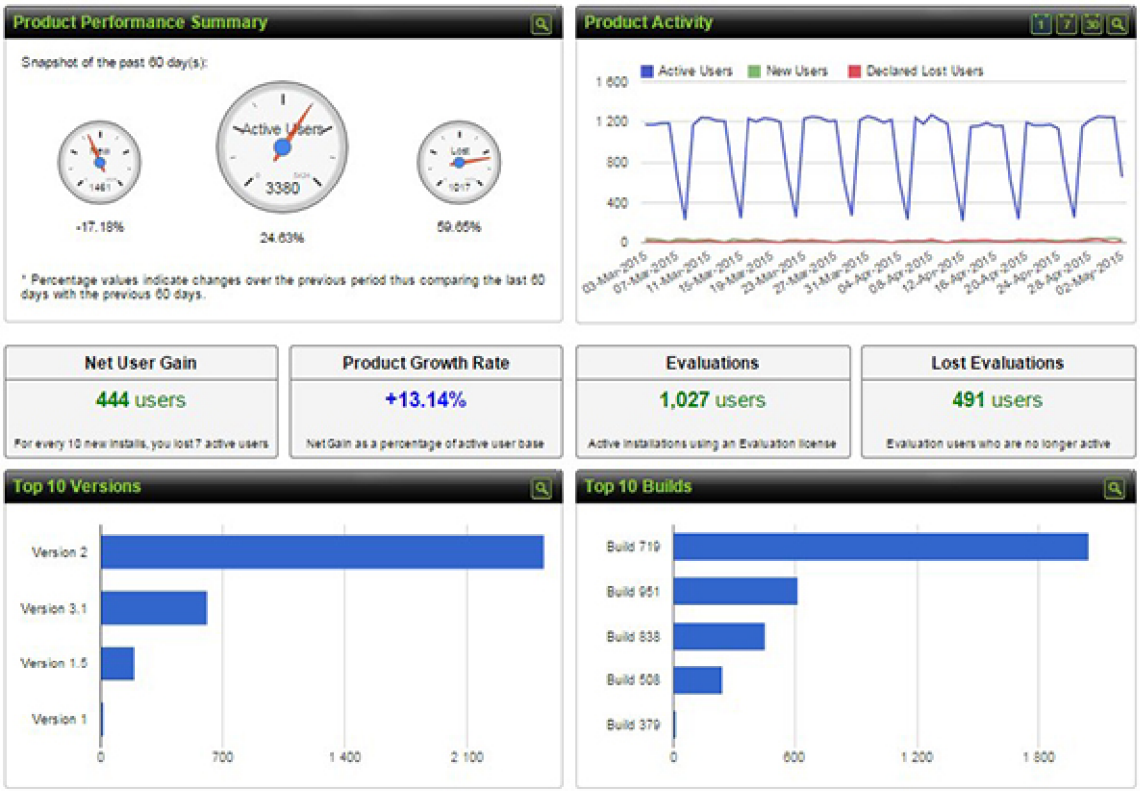
Nástroj dále umožňuje sběr zpětné vazby od uživatelů, validaci licenčních klíčů, službu automatického zjišťování aktualizací aplikace a další nástroje pro marketing i vývoj [37].

4.2.3 DeskMetrics

Nástroj DeskMetrics[8] je nástroj podobný Trackerbird. Umožňuje sledovat tyto informace o aplikaci [13][14]:

- počet aktivních uživatelů,
- jak často je aplikace používána a kdy (který den v týdnu a čas),
- četnost používání jednotlivých funkcí aplikace,
- verze operačního systému, informace o instalaci Javy,
- vývojářem definované události a další záznamy o činnosti aplikace,
- počet instalací a odinstalací aplikace.

[7] www.trackerbird.com
[8] www.deskmetrics.com



Obrázek 4.2: Nástroj Trackerbird

4.2.4 StatHat

Nástroj StatHat[9] se od ostatních liší, a to tím, že neposkytuje žádné konkrétní metriky, ale umožňuje vytvoření vlastních. Tyto metriky mohou být dvou typů [36]:

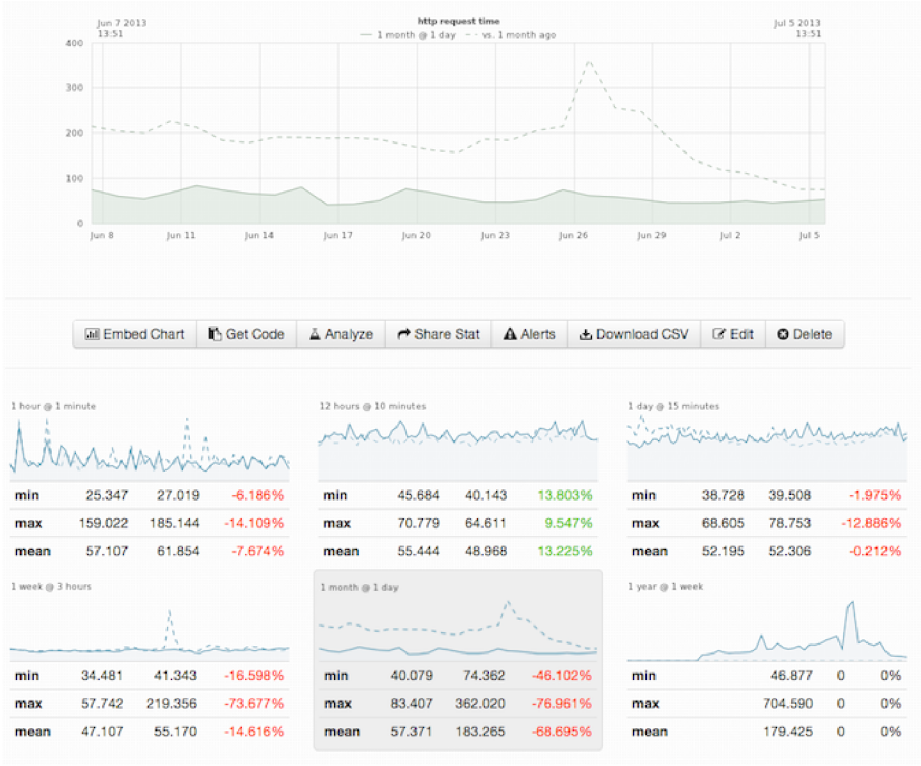
- čítač (*counter*)
- a hodnota (*value*).

V prvním případě nástroj počítá sumu zaslaných hodnot a je možné tak sledovat například četnost použití jednotlivých funkcí aplikace. Ve druhém případě nástroj počítá průměr zaslaných hodnot, čímž umožňuje sledovat například průměrnou dobu strávenou v průvodci nastavením aplikace. [36]

Tento nástroj poskytuje API pro 16 různých programovacích jazyků a je možné použít jej i pro webové aplikace.

Ukázka prezentace statistiky ve webovém rozhraní nástroje je na obrázku 4.3.

[9] www.stathat.com



Obrázek 4.3: Nástroj StatHat, obrázek převzat z `http://www.stathat.com`

4.3 Zhodnocení existujících řešení

Všechny dostupné nástroje umožňují sledovat či měřit použitelnost desktopové aplikace jen ve velmi omezené míře, například sledováním nejpoužívanějších funkcí. Širší možnosti nabízí nástroj Google Analytics, který sleduje průchod uživatele stránkami webu, na které lze namapovat události v desktopové aplikaci. Toto řešení však neposkytuje možnost rozlišit konkrétní sezení uživatele, sleduje pouze obecné tendence v chování uživatelů, což může skrýt informace důležité pro zlepšení použitelnosti aplikace.

Kapitola 5

Analýza a návrh řešení

Vzhledem k neexistenci vyhovujícího řešení pro měření použitelnosti desktopových aplikací, které by mohlo pomoci použitelnost těchto aplikací zlepšit, byl navržen nástroj pro toto měření.

Nástroj bude sestávat z klientské části, jejímž úkolem bude sběr a odesílání dat, a ze serverové části, která bude data přijímat, vyhodnocovat je a zobrazovat.

5.1 Výběr metrik

Pro měření použitelnost aplikace je třeba sledovat především chování uživatele. To bude provedeno prostřednictvím zaznamenávání následujících událostí a informací:

- prvků grafického uživatelského rozhraní testované aplikace, na něž uživatel klikl myší,
- doby mezi jednotlivými kliknutími,
- dráhy, kterou uživatel urazil kurzorem myši na obrazovce mezi jednotlivými kliknutími,
- a chybných vstupních hodnot, které uživatel zadal.

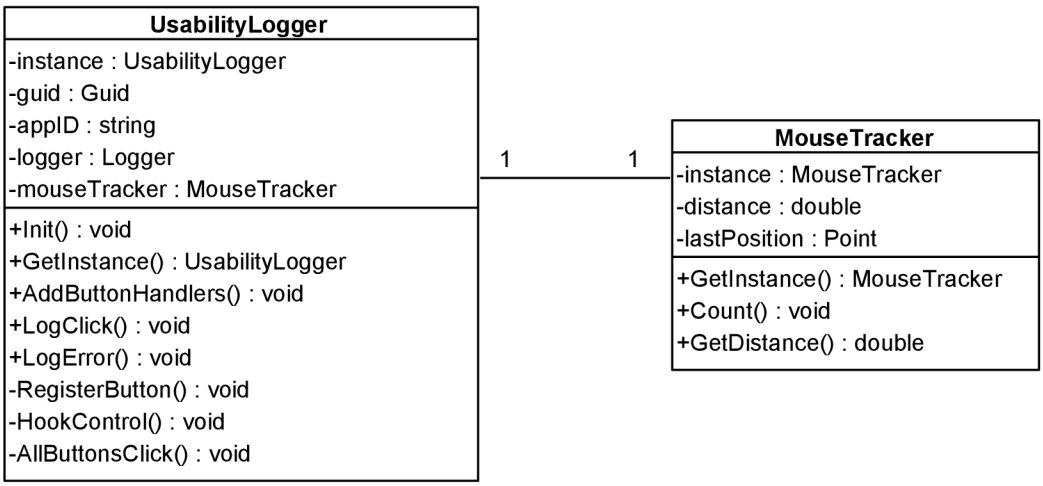
5.2 Vývojové prostředí a technologie

Pro implementaci nástroje pro měření použitelnosti byla vzhledem k aplikacím, v nichž bude nástroj využit, zvolena platforma Microsoft .NET Framework a programovací jazyk C#. V budoucnu bude možné rozšířit nástroj o klientskou část implementovanou v dalších jazycích, aby byl nástroj použitelný pro více aplikací.

5.3 Klient

Návrh klientské části sestává ze dvou tříd, obou vytvořených podle návrhového vzoru *jedináček* (*singleton*). Třídy jsou znázorněny v diagramu na obrázku 5.1.

Jádrem klientské části je třída `UsabilityLogger`, která vývojáři aplikace umožní zasílat zprávy o vzniklých událostech prostřednictvím metod `LogClick()` a `LogError()`. Pomocí první budou zaznamenávány kliknutí myši na prvky grafického uživatelského rozhraní, pomocí druhé pak chybné vstupní hodnoty. Dráhu kurzoru myši bude počítat metoda `Count()`



Obrázek 5.1: Návrh klientské části

třídy `MouseTracker` v samostatném vlákně. Doba mezi jednotlivými kliknutími bude počítána serverovou částí na základě času vytvoření zpráv o po sobě následujících událostech.

Klient umožní vývojáři také automatické zasílání zpráv o kliknutí na tlačítko. Za tímto účelem bude obsahovat metodu `AddButtonHandlers()`, která v daném okně aplikace (objekt třídy `Form` ve Windows Forms) přiřadí všem tlačítkům (objektům, na které lze kliknout) *handler*, jenž při kliknutí na tlačítko odešle zprávu metodou `LogClick()`. Metoda `AddButtonHandlers()` také zajistí odeslání informace o přítomnosti tlačítka v okně serverové části, a to pomocí metody `RegisterButton()`.

5.3.1 Formát zasílaných zpráv

Klientská část bude serveru zasílat čtyři typy zpráv. Tyto zprávy budou sestávat z data a času vzniku zprávy, typu zprávy, identifikátoru sledované aplikace, unikátního identifikátoru sezení (GUID, *Globally unique identifier*) a informací specifických pro daný typ zprávy. Obecný formát zpráv lze zapsat takto:

```
<datum_a_čas>|<typ_zprávy>|<id_aplikace>|<specifické_informace>
```

Datum a čas bude uváděn ve formátu `yyyy-MM-dd HH:mm:ss.mmm`. Typy zpráv ukazuje tabulka 5.1.

`R`	zpráva o přítomnosti tlačítka v okně aplikace
`C`	zpráva o kliknutí myší na prvek grafického uživatelského rozhraní
`E`	zpráva o chybném vstupu
`M`	zpráva s délkou dráhy kurzoru myši mezi dvěma kliknutími

Tabulka 5.1: Typy zpráv

Unikátní identifikátor bude generován pomocí metody `NewGuid()` třídy `Guid`, která je součástí .NET Frameworku.

Informace specifické pro daný typ zprávy ukazuje tabulka 5.2.

R	<jméno_prvku>
C	<guid>|<jméno_prvku>
E	<guid>|<jméno_prvku>|<chybná_hodnota>
M	<guid>|<jméno_prvku_od>|<jméno_prvku_do>|<dráha_myši>

Tabulka 5.2: Informace specifické pro jednotlivé druhy zpráv

5.3.2 Odesílání zpráv

Zprávy budou serveru zasílány pomocí protokolu HTTP (*Hypertext Transfer Protocol*) a jeho metody POST. Samotné zasílání zpráv bude implementováno pomocí frameworku NLog[1].

5.4 Server

Úkolem serverové části nástroje pro měření použitelnosti software bude přijímat, uchovávat a prezentovat informace získané klientskou částí.

5.4.1 Specifikace požadavků

Serverová část bude umožňovat vytvářet vzory chování uživatele. Tyto vzory budou sestávat ze sekvence prvků grafického uživatelského rozhraní sledované aplikace a budou představovat ideální způsob práce se sledovanou aplikací (celou nebo její částí), jak jej definoval vývojový tým. Vytvořené vzory pak bude možné automaticky porovnávat se sekvencemi zpráv o kliknutí na prvek grafického uživatelského rozhraní sledované aplikace přijatými od klientské části nástroje.

Serverová část bude dále poskytovat statistiky času a délky dráhy kurzoru myši mezi kliknutími a chyb při zadávání vstupních hodnot, kterých se dopustil uživatel sledované aplikace. Analýza chyb uživatele sledované aplikace může přinést cenné informace o tom, kde, jak často a jaké chyby uživatelé dělají, a může pomoci při návrhu opatření, která uživatelům usnadní vkládání dat a ochrání je tak před dalšími chybami.

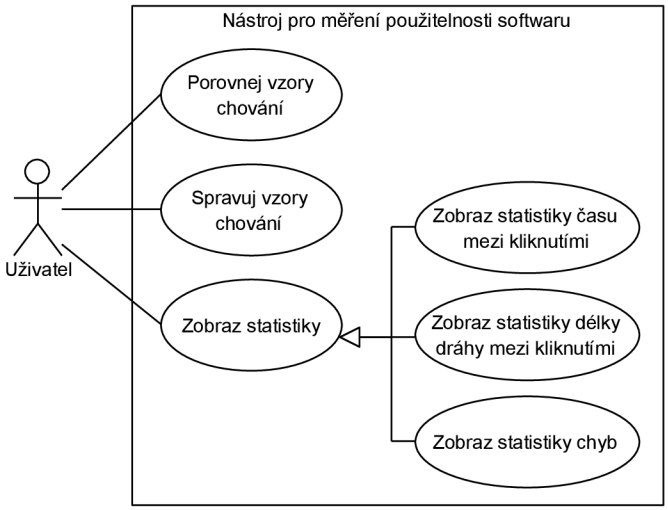
Případy užití nástroje pro měření použitelnosti software ukazuje obrázek 5.2.

5.4.2 Architektura MVC

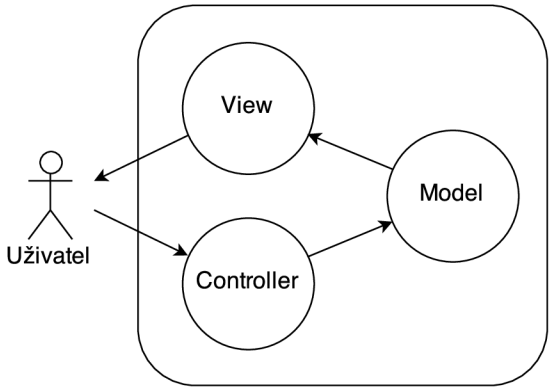
Serverová část nástroje je navržena jako webová aplikace implementovaná pomocí technologie ASP.NET podle architektonického vzoru Model-View-Controller (MVC). Tato architektura umožňuje oddělit aplikační logiku od prezentace dat. Vztahy mezi třemi hlavními komponentami MVC jsou znázorněny na obrázku 5.3.

Model popisuje datový model aplikace, je to jediná část aplikace, která interaguje s datovým úložištěm. Jako datové úložiště pro serverovou část nástroje pro měření použitelnosti bude použita relační databáze. *View* (pohled) je ta část aplikace, která je prezentována uživateli. Její obsah umožňuje uživateli provádět operace *CRUD* (*create, read, update, delete*), tedy vytvářet data, číst je, upravovat a odstraňovat. *Controller* tvoří most mezi modelem a pohledy. Přijímá požadavky od uživatele a vybírá pohledy, které tyto požadavky obslouží. [23]

[1] nlog-project.org



Obrázek 5.2: Model případů užití nástroje pro měření použitelnosti software



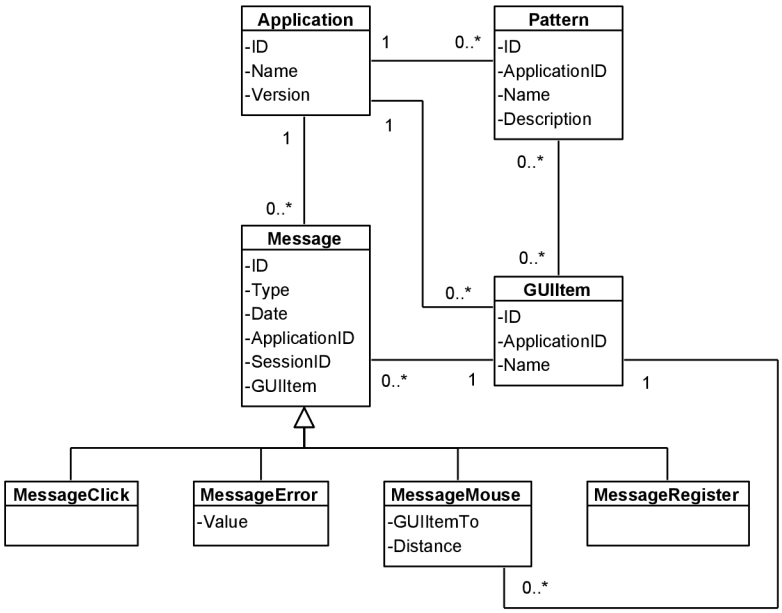
Obrázek 5.3: Architektura Model-View-Controller, vytvořeno dle [7]

5.4.3 Datový model aplikace

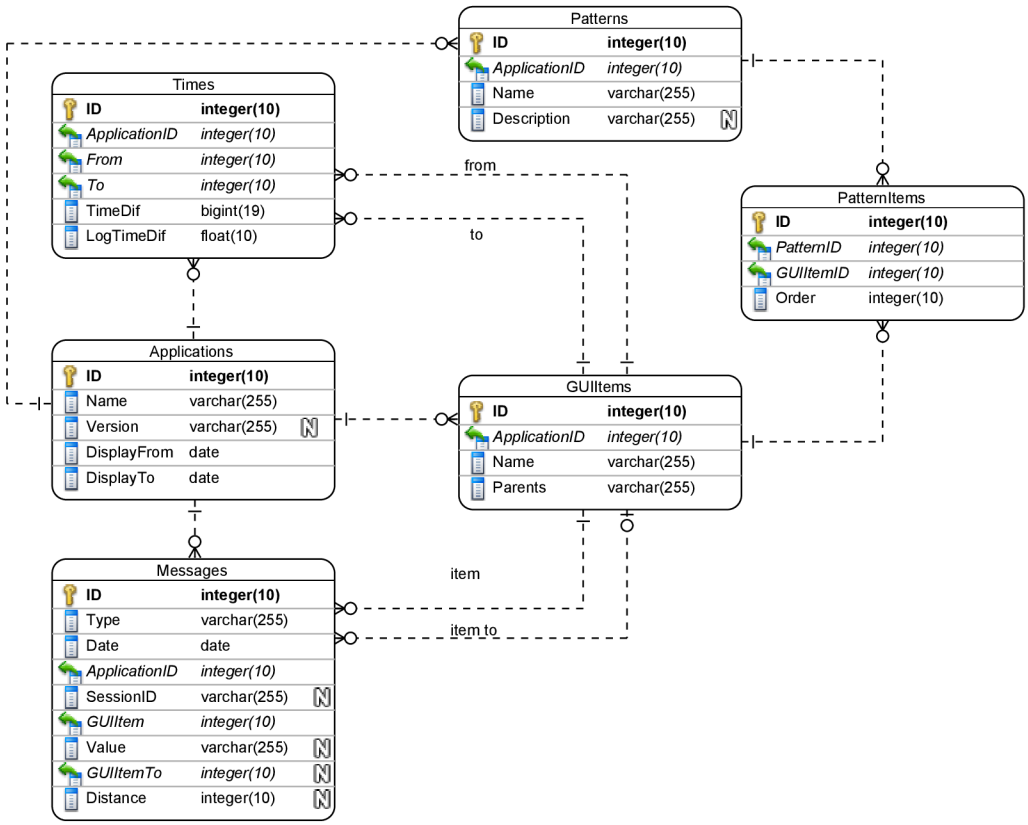
Datový model aplikace obsahuje entity pro sledovanou aplikaci (*Application*), prvky jejího grafického uživatelského rozhraní (*GUIItem*), vzory chování uživatelů (*Pattern*) a zprávy od klientské části (*Message*). Jednotlivé typy zpráv jsou modelovány samostatnými třídami, přičemž všechny jsou potomky třídy *Message*. Datový model je zobrazen na obrázku 5.4.

5.4.4 Úložiště dat

Jako úložiště dat byla vybrána relační databáze MS SQL Server. Návrh databáze je na obrázku 5.5 a vychází z datového modelu aplikace představeného v kapitole 5.4.3.



Obrázek 5.4: Návrh datového modelu aplikace



Obrázek 5.5: Návrh databáze

5.4.5 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní serverové části nástroje bylo navrženo tak, aby bylo co možná nejjednodušší. Implementováno bude pomocí HTML, CSS a JavaScript frameworku Bootstrap[2]. Návrh uživatelského rozhraní je na obrázcích 5.7 a 5.8 (návrhy byly vytvořeny pomocí nástroje NinjaMock[3]).

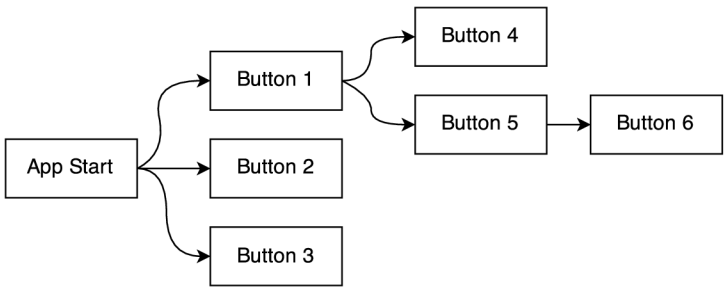
5.4.6 Správa sledovaných aplikací

Serverová část nástroje bude umožňovat sledovat více různých aplikací zároveň. Při vložení sledované aplikace bude vygenerován identifikátor aplikace potřebný při inicializaci klientské části. Aplikaci bude možné editovat a odstranit. Uživateli nástroje bude také umožněno smazat všechna data přijatá dosud od klientů, případně odstranit data z vybraného časového období, aby při zásadnější aktualizaci sledované aplikace nebyly statistiky zkreslovány starými daty.

5.4.7 Analýza chování uživatelů na základě vzorů jejich chování

Serverová část nástroje bude umožňovat vytvářet vzory chování uživatelů. Tyto vzory budou sestávat ze sekvence jednotlivých prvků grafického uživatelského rozhraní, na které může uživatel kliknout (tlačítka). Vytvořené vzory budou porovnávány s daty přijatými od klientské části.

Data od klientské části nástroje budou transformována do stromů, jehož kořen bude představovat start sledované aplikace a každá další úroveň stromu pak prvek grafického uživatelského rozhraní aplikace, na který uživatel klikl myší jako první, druhý, třetí atd. Tento strom je ilustrován obrázkem 5.6.

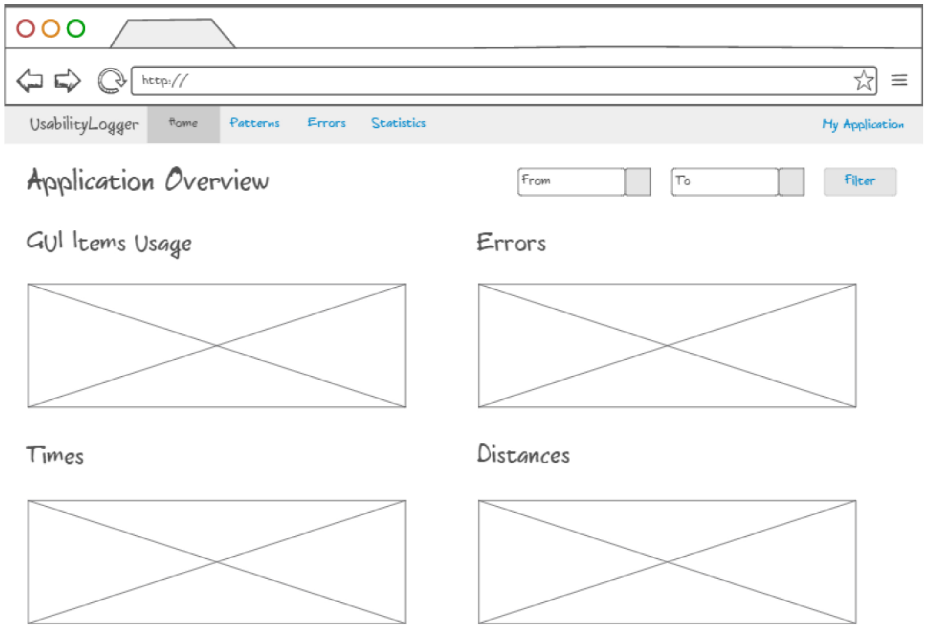


Obrázek 5.6: Strom kliknutí na prvky grafického uživatelského rozhraní

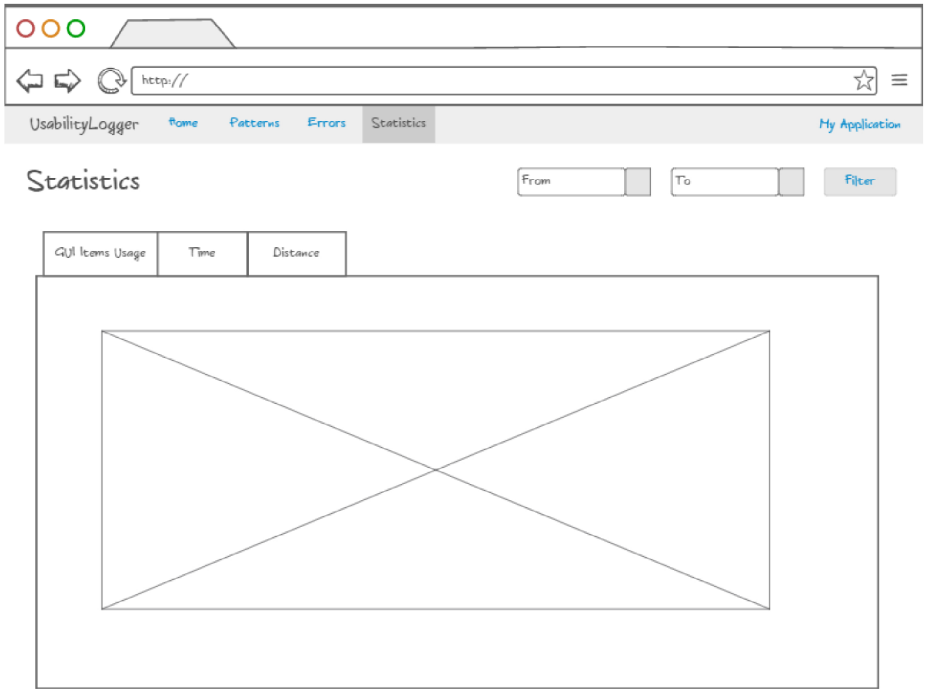
Zobrazení vzorů

Data od klientské části transformovaná do stromů budou zobrazována pomocí knihovny JavaScript InfoVis Toolkit[4], která obsahuje metody pro zobrazení stromu prvků. Části stromu odpovídající uloženým vzorům chování budou zvýrazněny zelenou barvou. Na jednotlivých úrovních jednotlivých podstromů bude zobrazena také procentuální četnost použití daného

[2] http://getbootstrap.com
[3] http://ninjamock.com
[4] http://philogb.github.io/jit/



Obrázek 5.7: Návrh grafického uživatelského rozhraní – úvodní stránka s přehledem



Obrázek 5.8: Návrh grafického uživatelského rozhraní – stránka se statistikami

prvku na dané úrovni, což umožní najít cestu aplikací nebo její částí, kterou uživatelé volí nejčastěji, a také průměrný čas mezi kliknutími na prvky uživatelského rozhraní.

Vybraná knihovna umožňuje přepínat mezi vertikálním a horizontálním zobrazením stromu, což dovolí uživateli nástroje vybrat takové zobrazení, které odpovídá množství a struktuře dat a zároveň uživateli více vyhovuje. Aby byla orientace ve velkém množství dat snazší, bude možné strom přibližovat a oddalovat, posouvat v rámci oblasti, ve které je zobrazen, a vybírat k zobrazení jen tu větev stromu, kterou chce uživatel nástroje zkoumat.

Správa modelových vzorů

Pro každou sledovanou aplikaci bude možné vytvořit více modelových vzorů. Každý vzor bude mít přiřazeno jméno a pro lepší identifikaci bude možné uchovávat i jeho popis. Jak již bylo napsáno výše, vzor bude sestávat ze sekvence prvků grafického uživatelského rozhraní sledované aplikace. Tyto prvky budou uloženy v databázi serverové části po přijetí registračních zpráv prvků od klienta. Uživateli nástroje budou uložené prvky nabídnuty při vytváření sekvence. Již vytvořený vzor bude moci uživatel upravovat odstraňováním a přidáváním nabídnutých prvků.

5.4.8 Statistické zpracování dat

Kromě analýzy chování uživatelů na základě vzorů jejich chování bude serverová část nástroje poskytovat také statistiky času mezi kliknutími, dráhy ujeté myší mezi kliknutími a chyb, jichž se uživatel sledované aplikace dopustil (jedná se o chybně zadané hodnoty do vstupních polí formulářů).

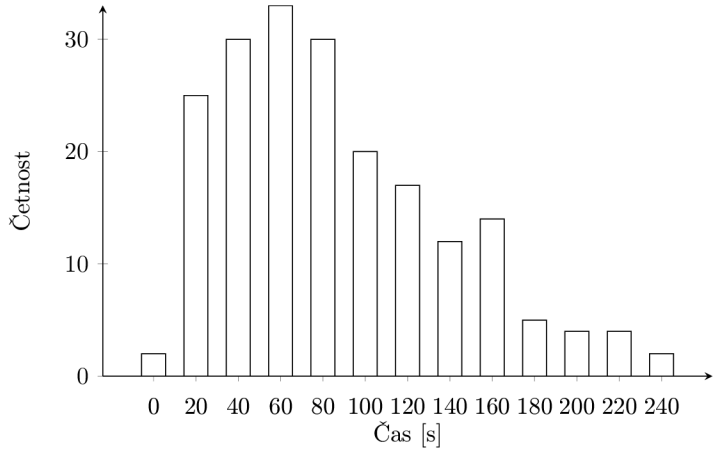
Čas měřený při uživatelském testování použitelnosti mívá obvykle rozložení hodnot pozitivně vychýlené jako na obrázku 5.9. Při takovém rozložení dat je nejvíce vypovídajícím průměrem průměr geometrický [34]. Existují dva způsoby, jak geometrický průměr spočítat. [10] První metoda počítá geometrický průměr n čísel jako n-tou odmocninu součinu těchto čísel:

$$\overline{x}_G = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

Tato metoda není pro počítačové zpracování velkého množství dat příliš vhodná, protože při součinu čísel může snadno dojít k přetečení. Tento problém řeší metoda druhá, která využívá aritmetického průměru a logaritmické funkce a bude použita pro počítání průměrného času mezi kliknutími:

$$\overline{x}_G = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln x_i\right)$$

Spolu s počítáním průměrného času mezi kliknutími bude také analyzován rozdíl mezi ním a krajními hodnotami (minimem a maximem). Taková analýza může upozornit na místa, kde je rozložení času mezi kliknutími vychýlené spíše negativně, což může ukazovat na problém v takových místech (např. uživatelé dlouho přemýšlejí nad dalším krokem).



Obrázek 5.9: Pozitivně vychýlené rozložení času

Rozložení hodnot dráhy ujeté myší po monitoru mezi kliknutími bude s největší pravděpodobností náhodné, hodnoty se budou pohybovat od nejkratší možné cesty až po velmi vysoké hodnoty (např. pokud bude uživatel zároveň pracovat i s jinou aplikací). Pro účely statistik bude počítán z přijatých hodnot aritmetický průměr:

$$\overline{x} = \sum_{i=1}^{n} x_i$$

Při zpracování počtu chyb a použití jednotlivých prvků grafického uživatelského rozhraní aplikace bude počítána pouze četnost výskytů chyb v daném prvku a četnost kliknutí na prvky rozhraní. V obou případech bude zajímavé sledovat maximální hodnoty, tedy kde uživatelé nejčastěji dělají chyby a které prvky nejčastěji používají.

Pro grafickou prezentaci všech statistik bude použit pruhový graf s hodnotami seřazenými od nejvyšších po nejmenší. Tento graf umožní přehledné zobrazení i velkého množství dat na stránce. Zobrazení grafů bude realizováno pomocí knihovny Google Charts[5].

[5] https://developers.google.com/chart/

Kapitola 6

Implementace

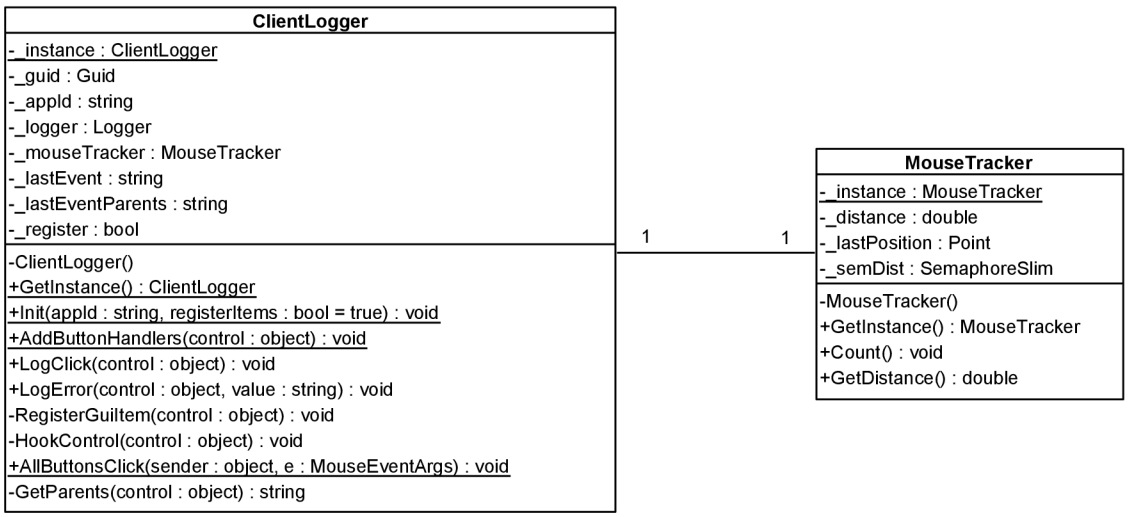
Tato kapitola se věnuje implementaci nástroje pro měření použitelnosti software. Popisuje podrobněji některé zajímavější části implementace jak klientské, tak serverové části tohoto nástroje.

6.1 Klient

Klientská část nástroje byla implementována v jazyce C# pro použití v aplikacích napsaných v tomto jazyce. Fungovat bude v aplikacích vytvořených pomocí frameworku Windows Forms. Do budoucna je možné vytvořit klientskou část i v jiných jazycích (např. Java), případně pro jiné frameworky (např. Windows Presentation Foundation), pokud tyto implementace dodrží formát zpráv zasílaných serverové části nástroje.

6.1.1 Diagram tříd

Diagram tříd klientské části nástroje je zobrazen na obrázku 6.1. Popis tříd a funkcí jejich metod je uveden v následujících kapitolách.



Obrázek 6.1: Diagram tříd klientské části nástroje

6.1.2 Zasílání zpráv

Samotné zasílání zpráv bylo implementováno pomocí frameworku NLog. Tento framework nabízí širokou škálu možností logování (např. zápis do souboru, zasílání zpráv pomocí protokolů UDP (*User Datagram Protocol*) a TCP (*Transmission Control Protocol*) a mnoho dalších). Vzhledem k tomu, že serverová část byla navržena a implementována jako webová aplikace, byl zvolen způsob zasílání zpráv pomocí protokolu HTTP. Framework NLog zároveň umožňuje zasílat zprávy asynchronně, čehož bylo také využito, a byly tak sníženy nároky na běh klientské části aplikace v rámci sledované aplikace.

Před posláním první zprávy je třeba klientskou část inicializovat. K tomu slouží metoda `Init()`, která jako parametry přijímá identifikátor aplikace a informaci o tom, zda posílat serveru informace o existenci prvků grafického uživatelského rozhraní. Identifikátor aplikace generuje serverová část při přidání sledované aplikace do její databáze. Následně je možné získat instanci klientské části voláním metody `GetInstance()`.

Jak již bylo popsáno v návrhu v kapitole 5.3, zprávu lze poslat voláním metod `LogClick()` pro zaslání zprávy o kliknutí na prvek grafického uživatelského rozhraní a `LogError()` pro zaslání zprávy o chybě ve vstupní hodnotě. Obě tyto metody přijímají jako parametry jméno okna, ve kterém se prvek nachází, jméno prvku grafického uživatelského rozhraní, metoda `LogClick()` navíc ještě text prvku a metoda `LogError()` chybnou hodnotu.

Ostatní informace obsažené ve zprávě (ID aplikace, ID sezení, datum a čas) již klientská část nástroje doplní sama podle formátu definovaného v kapitole 5.3.1.

6.1.3 Přidání *handlerů* všem tlačítkům okna

V předchozí části bylo popsáno manuální zasílání zpráv o kliknutí na prvek grafického uživatelského rozhraní prostřednictvím volání metody `LogClick()`. Klientská část však implementuje také možnost přiřazení *handlerů* všem tlačítkům daného okna (objektu třídy `Form`), a to voláním metody `AddButtonHandlers()`, jejímž parametrem je právě objekt reprezentující toto okno.

Tato metoda pak volá metodu `HookControl()`, jejímž úkolem je rekurzivně projít všechny grafické prvky okna, všem objektům, které mohou reagovat na událost kliknutí myši, přidat *handler* `AllButtonsClick()` a zároveň voláním metody `RegisterGuiItem()` upozornit serverovou část nástroje na existenci tlačítka, aby ta jej mohla uložit do databáze a nabídnout uživateli nástroje při vytváření modelových vzorů chování. Přiřazení *handleru* danému objektu podle toho, zda může či nemůže reagovat na událost kliknutí myši, je implementováno pomocí reflexe. Tato technika umožňuje zjistit, zda typ (třída) daného objektu reaguje na určitou událost. V tomto případě je hledána událost `MouseClick`.

Metoda `AllButtonsClick()`, která je tlačítkům přiřazována jako *handler* události při kliknutí na něj, pak při kliknutí na tlačítko zasílá zprávu voláním metody `LogClick()`.

6.1.4 Počítání dráhy myši

Počítání dráhy ujeté myší po monitoru mezi jednotlivými kliknutími má na starosti třídy `MouseTracker`. Tato třída poskytuje metodu pro získání instance `GetInstance()`, metodu pro samotné počítání vzdálenosti `Count()` a metodu pro získání aktuálně spočítané dráhy `GetDistance()`.

Po vytvoření jediné instance třídy `MouseTracker` je v samostatném vlákně spuštěna metoda `Count()`, která každých 200 ms zjistí aktuální pozici kurzoru myši a spočítá Euklidovskou vzdálenost aktuální a předchozí pozice kurzoru:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Nová vzdálenost je pak přičtena k již spočítané vzdálenosti. Tato metoda běží nepřetržitě po celou dobu běhu klientské aplikace (přesněji od okamžiku inicializace klientské části nástroje metodou `Init()`).

Výše zmíněná metoda `LogClick()` kromě zprávy o kliknutí odesílá také zprávu o dráze myši od předchozího kliknutí na tlačítko. Metoda získá tuto informaci voláním metody `GetDistance()`, která vrací aktuálně spočítanou dráhu a zároveň tuto vynuluje pro počítání nové vzdálenosti. Z toho plyne, že k proměnné, v níž je aktuálně spočítaná vzdálenost uchovávána, mohou v jednu chvíli přistupovat dvě metody – `Count()` pro zápis a `GetDistance()` pro čtení. Vyloučení této situace je řešeno pomocí semaforu.

6.1.5 Identifikace prvků grafického uživatelského rozhraní

Jako identifikace prvků grafického uživatelského rozhraní sledované aplikace bylo navrženo jméno (atribut `Name`) daného prvku. Taková identifikace se v průběhu testování ukázala být nedostatečnou, protože některé prvky se stejným jménem se mohou vyskytovat ve více oknech aplikace a některé prvky nemusí mít tento atribut vůbec přiřazen. Ke jménu prvku bylo následně přidáno ještě jméno okna, ve kterém se prvek nachází. Ani tato identifikace ovšem nebyla dostatečná. Každý prvek je tedy identifikován kompletní hierarchií svých rodičů, přičemž na každé úrovni je zjišťováno jméno a typ prvku grafického uživatelského rozhraní. Pro tyto účely byly přidány do zpráv zasílaných serveru (viz kapitolu 5.3.1) položky pro seznam typů a jmen rodičů oddělených znakem „/“. I v tomto případě se však mohou vyskytnou prvky, které budou identifikovány jako stejné. Jednoznačnější identifikaci umožnilo ještě přidání textového obsahu prvku, ten ovšem také nemusí být přiřazen všem prvkům.

Pro přesnou identifikaci prvků se nabízel tzv. *hash* kód. Podle dokumentace společnosti Microsoft však není vhodné používat implicitní funkce generující tento kód, protože dva různé objekty mohou mít *hash* kód stejný a kód stejných objektů se může lišit v závislosti na procesu a platformě [5].

Jako jediným úplným řešením se jeví poctivé přiřazování unikátních jmen jednotlivým prvkům grafického uživatelského rozhraní sledovaných aplikací.

6.2 Server

Serverová část nástroje pro měření použitelnosti software byla implementována jako webová aplikace v jazyce C# za použití frameworku ASP.NET.

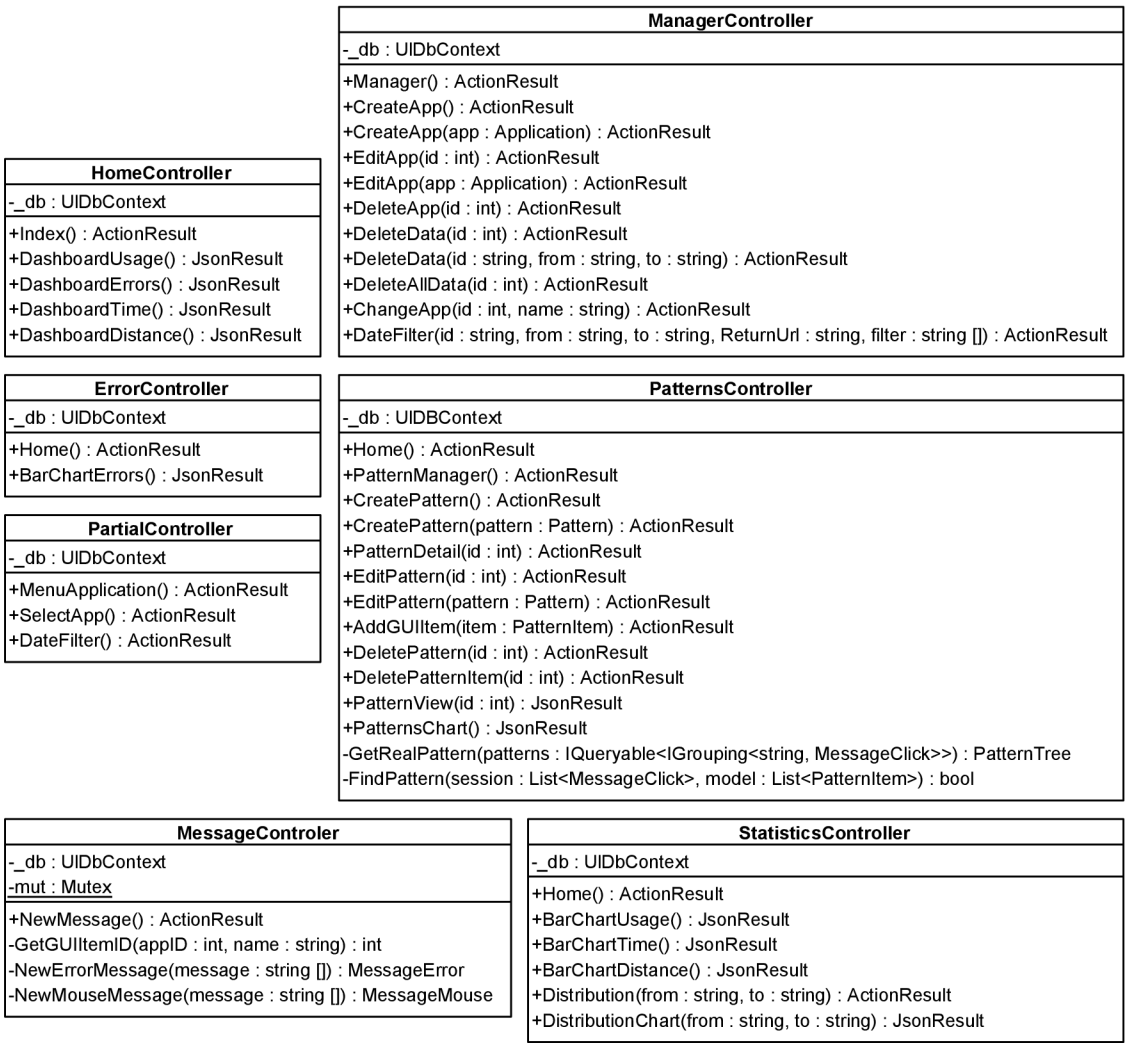
6.2.1 Diagram tříd

Diagram tříd serverové části nástroje pro měření použitelnosti je na obrázcích 6.2 a 6.3. Implementace serveru je rozdělena do dvou projektů.

První projekt (*UsabilityLoggerServer*) obsahuje třídy (kontroléry) tvořící most mezi pohledy a datovým modelem a také samotné pohledy, tedy HTML stránky zobrazované uživateli nástroje.

Druhý (*UsabilityLoggerServer.Models*) potom obsahuje všechny ostatní třídy:

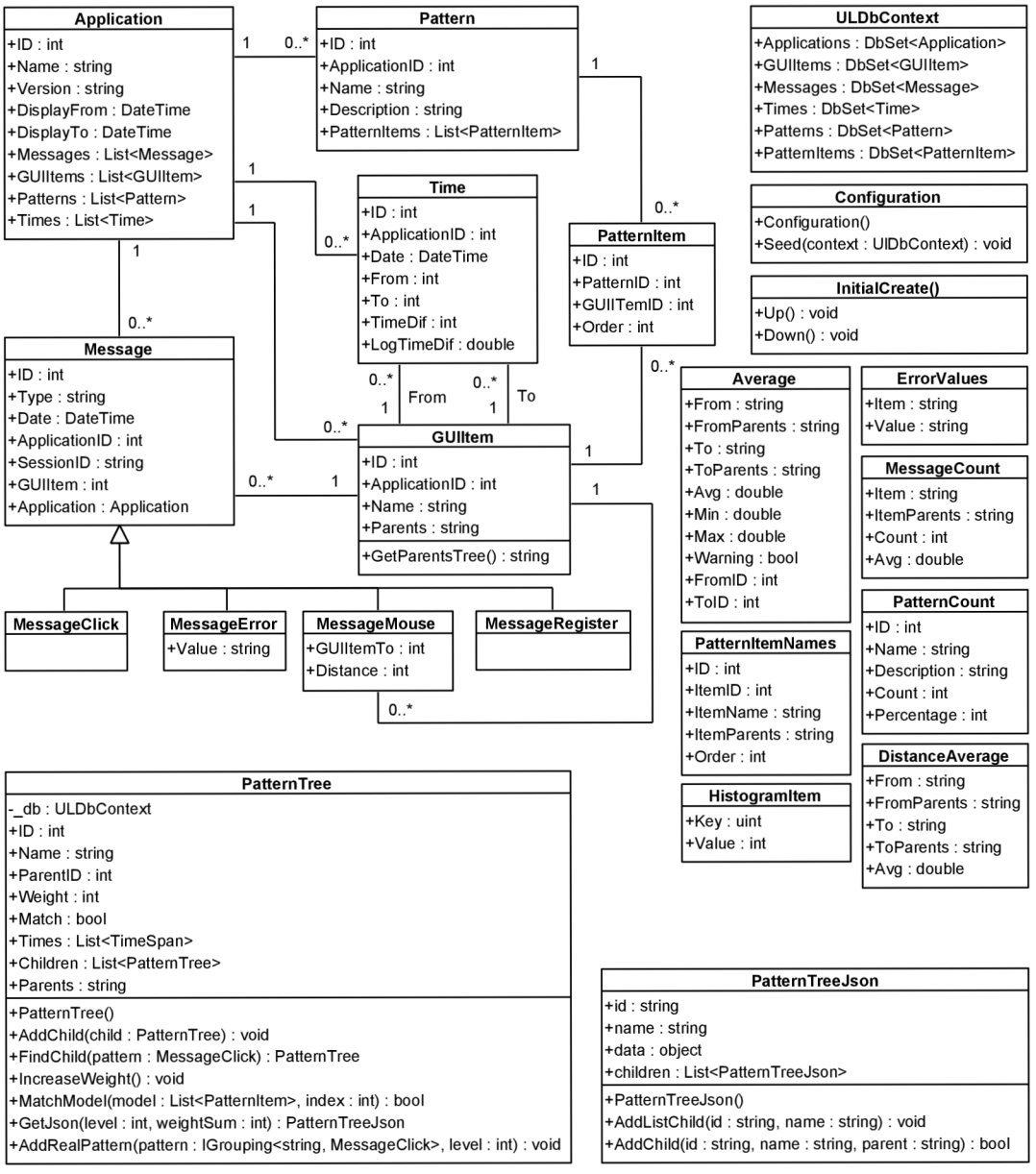
- třídy datového modelu odpovídající struktuře databáze (`Application`, `Message` a její potomky, `GUIItem`, `Time`, `Pattern` a `PatternItem`),
- třídu `ULDbContext` pro přístup k databázi,
- třídy `Configuration` a `InitialCreate`, které zajišťují migraci databáze při změně datového modelu, a to beze ztráty dat uložených v databázi,
- třídy sloužící pro vytvoření stromových struktur vzorů chování (`PatternTree` a `PatternTreeJson`)
- a třídy určené k přenosu dat z kontrolérů do pohledů (`Average`, `ErrorValues`, `MessageCount`, `PatternCount`, `DistanceAverage` a `PatternItemNames`).



Obrázek 6.2: Diagram tříd serverové části nástroje – projekt *UsabilityLoggerServer*

Aplikace využívá přístupu *code first* pro automatické vytvoření struktury databáze na základě datového modelu aplikace.

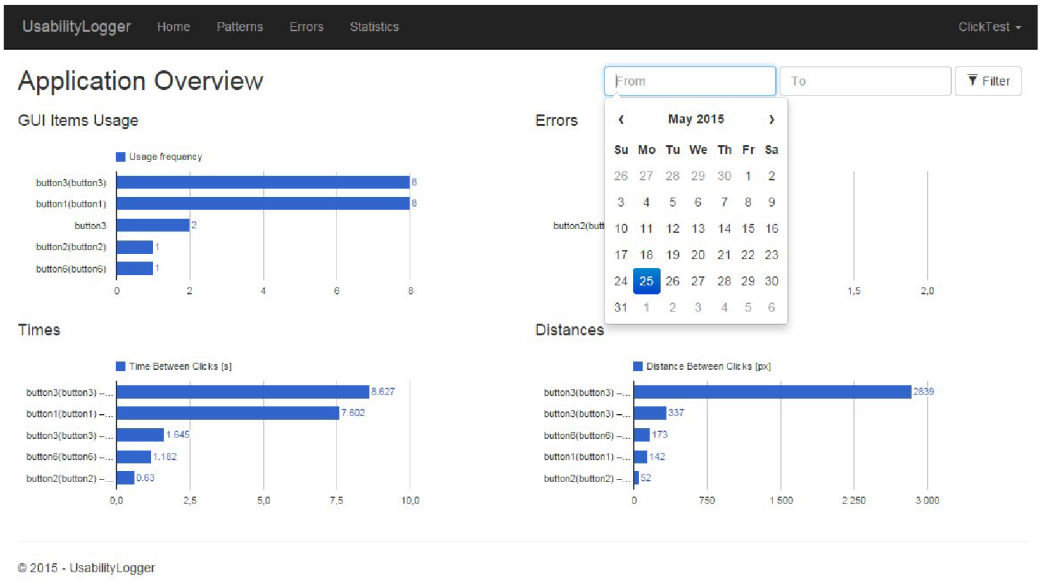
Pro přenos dat z kontrolérů do pohledů nelze využít tzv. anonymních typů, které jsou často typem výsledků dotazů do databáze. Výsledek dotazu je třeba buď serializovat, nebo vytvořit speciální typy pro jejich přenos. Pro tuto aplikaci byla zvolena druhá možnost, protože je jednodušší a čitelnější.



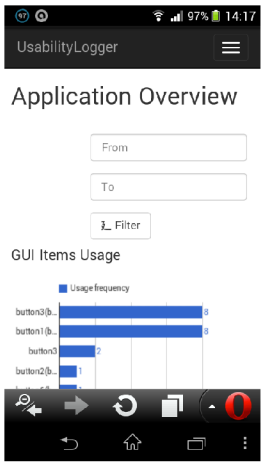
Obrázek 6.3: Diagram tříd serverové části nástroje – projekt *UsabilityLoggerServer.Models*

6.2.2 Uživatelské rozhraní

Uživatelské rozhraní aplikace bylo implementováno v jazycích HTML (*HyperText Markup Language*) verze 5, CSS (*Cascading Style Sheets*) verze 3 a JavaScript. Byl použit CSS framework Bootstrap verze 3.3.4, který umožňuje vytvoření tzv. responzivního uživatelského rozhraní, tedy rozhraní, které se upravuje podle velikosti displeje zařízení, na němž je web zobrazen. Vzhledem k očekávanému velkému množství dat, které bude serverovou částí zobrazováno, se však předpokládá, že bude web prohlížen spíše na větších monitorech než v mobilních zařízeních. Ukázky uživatelského rozhraní aplikace na monitoru notebooku a na displeji mobilního telefonu jsou na obrázcích 6.4 a 6.5.

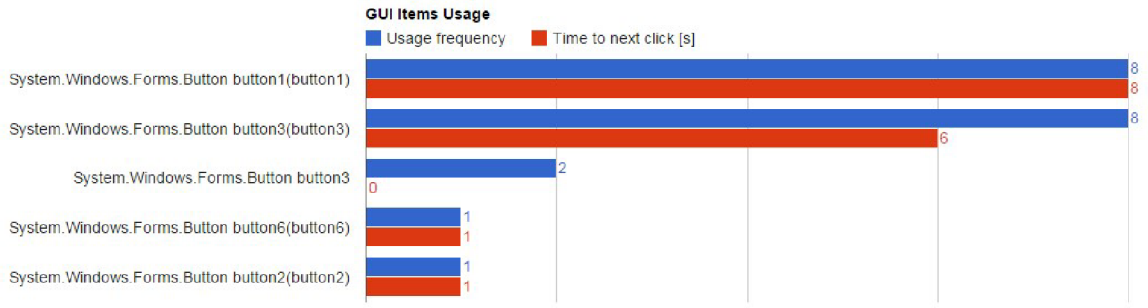


Obrázek 6.4: Uživatelské rozhraní aplikace na monitoru notebooku – úvodní stránka

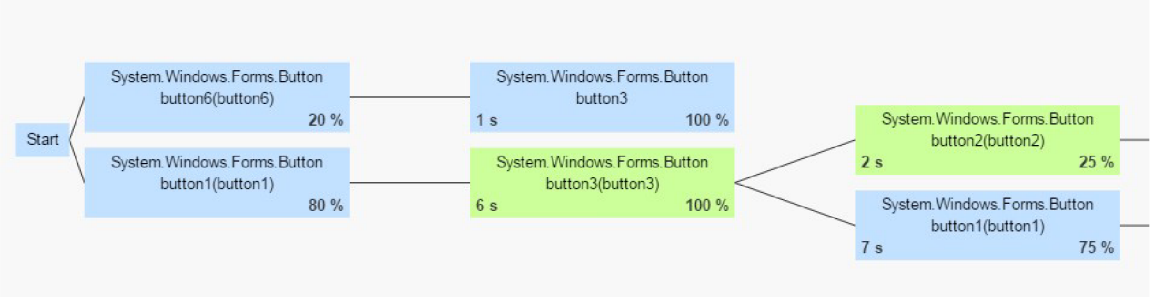


Obrázek 6.5: Uživatelské rozhraní aplikace na displeji mobilního telefonu – úvodní stránka

Pro zobrazení statistických dat byla použita knihovna Google Charts napsaná v jazyce JavaScript, pro zobrazení vzorů chování pak knihovna JavaScript Infovis Toolkit napsaná ve stejném jazyce. Statistická data jsou vzhledem k předpokládanému velkému množství těchto dat zobrazována v řádkových grafech (*Bar Charts*) a řazena dle velikosti sestupně, takže nejdůležitější informace (nejdelší časy mezi kliknutími či nejčastěji používané prvky) budou vždy na začátku stránky. Vzory chování budou zobrazovány ve stromovém grafu (*SpaceTree*). Příklady zobrazení dat jsou na obrázcích 6.6 a 6.7.



Obrázek 6.6: Graf se statistikou četnosti používání prvků grafického uživatelského rozhraní



Obrázek 6.7: Příklad zobrazení stromu prvků a nalezených vzorů

6.2.3 Vyhledávání modelových vzorů

Modelové vzory mají podobu jednoduchých sekvencí (seznamů) prvků grafického uživatelského rozhraní – objektů třídy `PatternItem`. Vzory odpovídající reálnému chování uživatelů sledované aplikace jsou pro zobrazení převáděny do stromu, jehož jednotlivé uzly jsou objekty třídy `PatternTree`. Vyhledávání modelových vzorů ve stromu reálného chování uživatelů provádí rekurzivně metoda `MatchModel()` třídy `PatternTree` podle algoritmu 1.

Tento algoritmus vyhledává všechny výskyty daného modelového vzoru ve stromu reálného chování uživatelů a jednotlivé uzly označuje jako odpovídající modelu, pokud byl model nalezen. Tyto uzly jsou v zobrazení označeny zelenou barvou namísto světle modré, použité pro ostatní uzly (viz ukázka na obrázku 6.7).

Strom reálného chování uživatelů s vyznačenými částmi odpovídajícími modelovým vzorům je pak převeden na strom, jehož uzly jsou objekty třídy `PatternTreeJson`, a to pomocí metody `GetJson()` třídy `PatternTree`. Třída `PatternTreeJson` odpovídá svými

atributy struktuře vyžadované knihovnou JavaScript Infovis Toolkit a je snadno serializovatelná do formátu JSON pro přenos do pohledu prostřednictvím technologie AJAX. Tato technologie umožňuje rychlé načtení uživatelského rozhraní aplikace bez nutnosti čekat na všechna data potřebná pro grafy. Grafy jsou pak zobrazeny až v okamžiku, kdy jsou data k dispozici. Stejný způsob získávání dat je použit také pro grafy statistik.

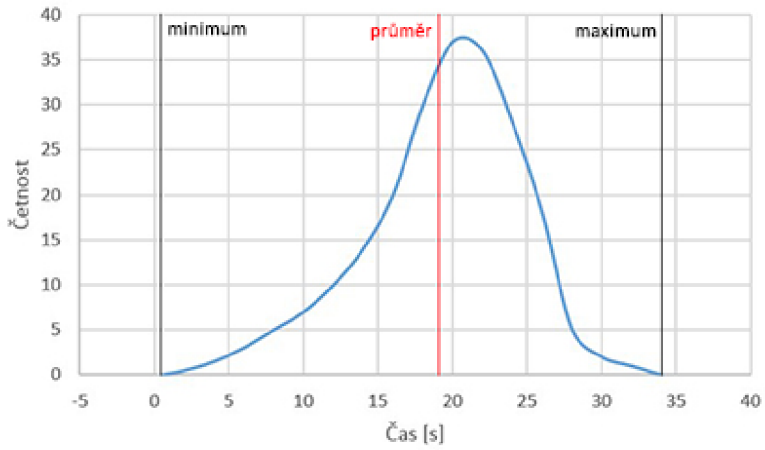
Algoritmus 1 Vyhledávání modelových vzorů

```
procedure MatchModel(model, index)
    if model je prázdný then
        return false
    end if
    match ← false
    found ← false
    if model[index] == aktuální uzel then
        match ← true
        if model[index] není posledním prvkem modelu then
            for all potomek aktuálního uzlu do
                if potomek.MatchModel(model, index + 1) then
                    found ← true
                end if
            end for
        else
            found ← true
        end if
    end if
    for all potomek aktuálního uzlu do
        potomek.MatchModel(model, 0)
    end for
    if match and found then
        aktuální uzel označ jako odpovídající modelu
        return true
    else
        return false
    end if
end procedure
```

6.2.4 Statistiky

Statistické zpracování dat bylo implementováno podle návrhu v kapitole 5.4.8. Pod grafy zobrazujícími statistiky času a vzdálenosti mezi kliknutími na prvky grafického uživatelského rozhraní se zobrazuje tabulka s minimálními, maximálními a průměrnými časy, resp. vzdálenostmi mezi kliknutími na prvky. V tabulce je zobrazen také náhled histogramu, z něhož lze vyčíst rozložení dat a určit tak problematická místa v aplikaci. Pro statistiky času jsou také generována varování, a to na základě analýzy průměrného, minimálního a maximálního času mezi danými prvky. Pokud je rozdíl průměrného a minimálního času větší než rozdíl maximálního a průměrného času, je rozložení hodnot časů pravděpodobně vychýlené negativně a průměrný čas je vyhodnocen jako nepřiměřeně dlouhý. Tuto situaci

ilustruje obrázek 6.8. Řádek tabulky s prvky, u nichž je varování vygenerováno, je červeně podbarven (viz obrázek 6.9).



Obrázek 6.8: Analýza času

From	To	Minimum time [s]	Average time [s]	Maximum time [s]	Time distribution
System.Windows.Forms.Button button3(button3)	System.Windows.Forms.Button button1(button1)	4	9	19	
System.Windows.Forms.Button button1(button1)	System.Windows.Forms.Button button3(button3)	2	8	12	

ClickTest.Form1 Form1
System.Windows.Forms.Button button3(button3)

Obrázek 6.9: Tabulka se statistikami času a varováním na nepřiměřeně dlouhý čas

6.2.5 Filtrování dat

Pro výběr počátečního a koncového data pro filtrování zobrazených statistik i dat ve vzorech chování byla použita knihovna Datepicker for Bootstrap[1] pro jazyk JavaScript. Filtrování je aktivní, pokud je vybráno alespoň jedno datum (druhé se doplní automaticky) a je zmáčknuto tlačítko *Filter*. Změna intervalu se projeví ihned po vybrání nového počátečního nebo koncového data. Filtrování je deaktivováno opětovným klepnutím na tlačítko *Filter*. Poslední nastavené filtrování je uchováváno v databázi pro každou sledovanou aplikaci zvlášť a po vypršení platnosti proměnných *Session* je automaticky nastaveno.

[1] http://www.eyecon.ro/bootstrap-datepicker/

Kapitola 7

Testování

Tato kapitola se věnuje způsobu testování nástroje pro měření použitelnosti software, problémům, které byly v jeho průběhu objeveny, a způsobům, jakým byly tyto problémy vyřešeny, pokud se je vyřešit podařilo.

7.1 Klient

Pro prvotní testování funkčnosti klientské části nástroje byla vytvořena malá desktopová aplikace ve frameworku Windows Forms. Testovací aplikace sestávala ze dvou oken a několika tlačítek v daném okně. Byla na ní ověřována základní funkčnost klientské části – správné přiřazování *handlerů* tlačítkům, počítání dráhy myši a odesílání zpráv serveru.

Jakmile byla implementována serverová část aplikace, začal se klient testovat také na aplikaci smartCore společnosti Siemens, CT Brno. Během tohoto testování se objevily problémy s přiřazováním *handlerů* tlačítkům. Původní implementace klienta totiž předpokládala, že všechna tlačítka jsou v hierarchii tříd potomky třídy `Button` v daném frameworku. Společnost však ve svých aplikacích používá také frameworky pro tvorbu grafického uživatelského rozhraní, jejichž prvky od této třídy nutně dědit nemusí, takže jim není *handler* přiřazen a kliknutí na ně nejsou sledována. Tento problém byl vyřešen použitím techniky reflexe a zjišťováním, zda prvek reaguje na kliknutí myši, jak bylo popsáno v kapitole 6.1.3.

Dalším problémem, který se vyskytl během testování klientské části, byla nedostatečná identifikace prvku grafického uživatelského rozhraní pomocí jména tohoto prvku. Ukázalo se, že některé prvky aplikace nemají jméno vůbec přiřazeno a že prvky se stejným jménem se mohou vyskytovat ve více oknech aplikace. Tento problém byl částečně vyřešen přidáním kompletní hierarchie předků daného prvku (viz kapitolu 6.1.5).

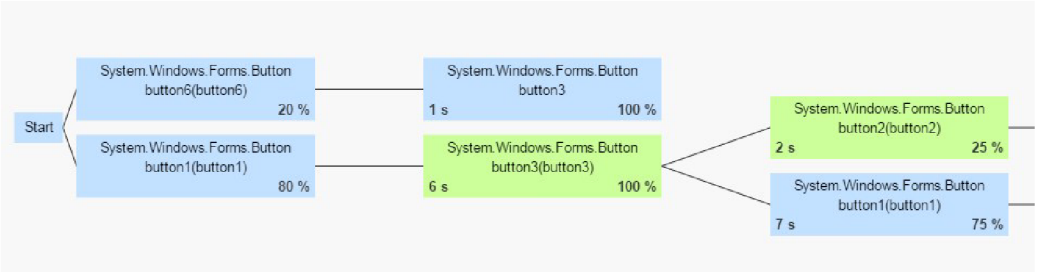
7.2 Server

Serverová část nástroje pro měření použitelnosti software byla testována na lokálním serveru IIS (*Interner Information Services*) Express verze 8.0 a také prostřednictvím služby AppHarbor[1] na adrese http://usabilitylogger.apphb.com, a to v prohlížečích Opera verze 29, Google Chrome ve verzi 42, Firefox verze 38 a Internet Explorer 11.

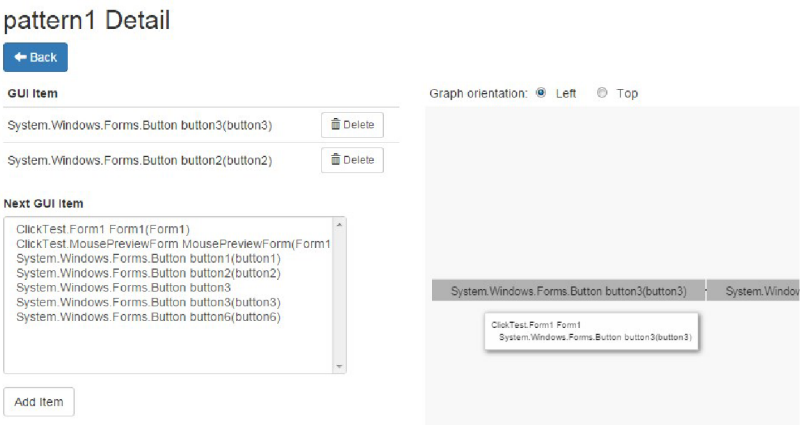
Jako testovací data byla použita data z testovací aplikace i z aplikace společnosti Siemens, CT Brno (viz kapitolu 7.1).

Prostřednictvím testovací aplikace (pojmenované jako *ClickTest*) byla vygenerována data tak, aby mohla být otestována veškerá funkčnost serverové části nástroje. Na datech z testovací aplikace bude na následujících řádcích představena kompletně funkčnost nástroje.

Stránka se vzory (*Patterns*) ukazuje vzory chování uživatelů. Z obrázku 7.1 je patrné, že po startu aplikace klikne 20 % uživatelů na tlačítko *button6* a zbylých 80 % uživatelů na tlačítko *button1*. V tomto případě pak vždy uživatelé pokračují kliknutím na tlačítko *button3* a průměrná doba do tohoto kliknutí je 6 sekund. V nástroji byl vytvořen modelový vzor chování obsahující dva prvky – tlačítka *button3* a *button2* (). Tento vzor byl v reálném chování uživatelů nalezen a je zobrazen zelenou barvou.



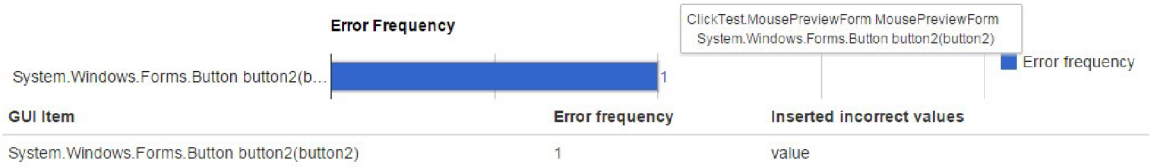
Obrázek 7.1: Vzory chování uživatelů – ilustrace zobrazení na testovacích datech



Obrázek 7.2: Vytvoření modelového vzoru chování

[1] http://appharbor.com

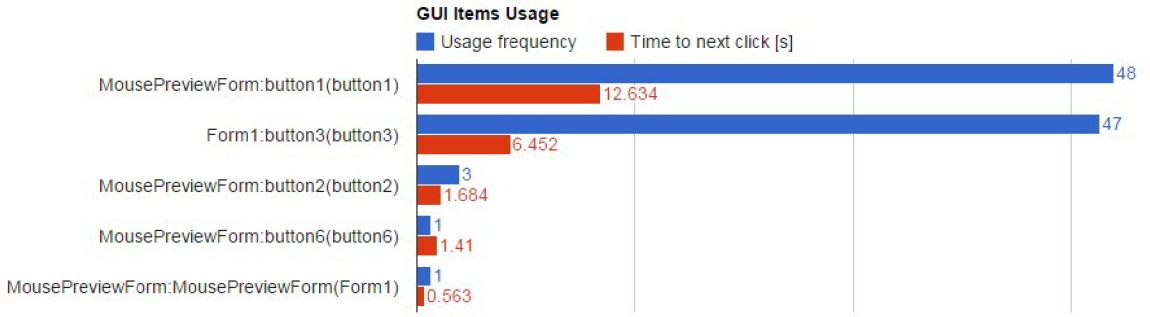
Stránka s chybami (*Errors*) ukazuje (viz obrázek 7.3), že byla zaslána zpráva o chybě v prvku *button2* s chybnou hodnotou „value“. Tato data jsou pouze testovací, při reálném použití bude chybná hodnota ta, kterou uživatel chybně zadal do určitého prvku grafického uživatelského rozhraní.



Obrázek 7.3: Chyba ve vstupní hodnotě – ilustrace zobrazení na testovacích datech

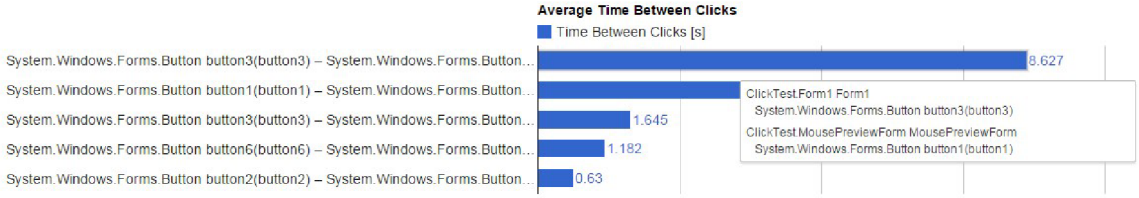
Stránka se statistikami obsahuje tři záložky – použití prvků grafického uživatelského rozhraní (*GUI Items Usage*), časy mezi kliknutími na prvky (*Times*) a dráhy ujeté myší po monitoru mezi kliknutími (*Distances*).

Graf na záložce s použitím prvků ukazuje, že nejčastěji používaných prvkem je tlačítko *button1* a že nejdéle se uživatel rozhoduje, na který další prvek kliknout, po kliknutí na stejné tlačítko (viz obrázek 7.4).



Obrázek 7.4: Nejčastěji používané prvky grafického uživatelského rozhraní – ilustrace zobrazení na testovacích datech

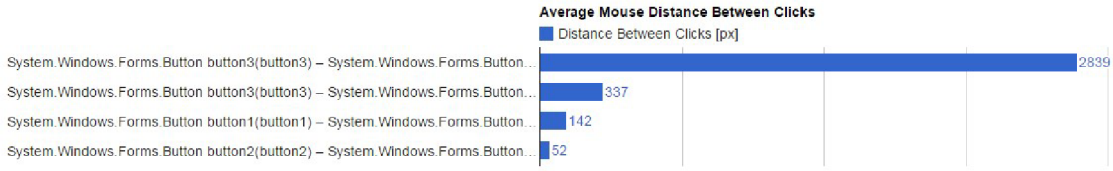
Záložka *Times* obsahuje graf, ze kterého lze vyčíst, že nejdelší čas stráví uživatelé mezi kliknutími na tlačítka *button3* a *button1*, a to v průměru 8, 6 sekundy (viz obrázek 7.5). Pod grafem je zobrazena tabulka s podrobnějšími informacemi, která ukazuje minimální, maximální a průměrné hodnoty časů mezi kliknutími na jednotlivé prvky. Mezi tlačítky *button1* a *button3* bylo vygenerováno varování o nepřiměřeně dlouhém čase, což je indikováno červeným zbarvením řádku tabulky. V pravé části tohoto řádku je náhled histogramu, který ukazuje negativně vychýlené rozložení dat. Takové rozložení znamená pravděpodobný problém, a pokud by se jednalo o reálnou aplikaci, bylo by vhodné se na toto místo v aplikaci zaměřit a například během uživatelského testování zjistit, co tento problém způsobuje, aby jej bylo možné vyřešit.



From	To	Minimum time [s]	Average time [s]	Maximum time [s]	Time distribution
System.Windows.Forms.Button button3(button3)	System.Windows.Forms.Button button1(button1)	4	9	19	
System.Windows.Forms.Button button1(button1)	System.Windows.Forms.Button button3(button3)	2	8	12	

Obrázek 7.5: Graf zobrazující dobu mezi kliknutími na prvky grafického uživatelského rozhraní – ilustrace zobrazení na testovacích datech

Záložka *Distances* ukazuje podobná data jako záložka s časy, nejsou zde však generována varování (viz obrázek 7.6).



From	To	Minimum distance [px]	Average distance [px]	Maximum distance [px]	Distance distribution
System.Windows.Forms.Button button3(button3)	System.Windows.Forms.Button button1(button1)	437	2839	5498	
System.Windows.Forms.Button button3(button3)	System.Windows.Forms.Button button2(button2)	337	337	337	
System.Windows.Forms.Button button1(button1)	System.Windows.Forms.Button button3(button3)	91	142	207	

Obrázek 7.6: Graf a tabulka zobrazující dráhu myši mezi kliknutími na prvky grafického uživatelského rozhraní – ilustrace zobrazení na testovacích datech

Kapitola 8

Závěr

Cílem diplomové práce bylo nastudovat problematiku kvality a použitelnosti softwaru, popsat možné způsoby jejího měření a navrhnout a implementovat nástroj pro měření použitelnosti softwarových produktů.

Nástroj měří použitelnost softwarových produktů prostřednictvím sledování činnosti uživatele a její analýzou a srovnáním se vzory definovanými vývojáři sledovaných produktů. Tato analýza umožní identifikovat místa v produktu, kde uživatelé postupují jinak, než bylo zamýšleno, případně místa, kde uživatelům trvá dlouho rozhodnout se, jaký další krok učinit, aby dospěli ke kýženému výsledku. Získané informace budou moci posloužit pro vytvoření scénářů kvalitativního testování založeném na přímém pozorování uživatelů při práci s produktem. Analýza chybně zadaných hodnot pak přinese možnosti pro vylepšení uživatelského rozhraní produktu tak, aby se snížil počet chyb uživatele a spolu s tím se zrychlila a zjednodušila jeho práce s produktem.

Jako další krok ve vývoji nástroje pro měření použitelnosti software se nabízí rozšíření nástroje o další sledované metriky, podrobnější analýzu získaných dat za účelem vyhledávání problematických míst ve sledovaných aplikacích či filtrování získaných dat podle jazykové verze sledované aplikace, což může poukázat například na problematický překlad částí těchto aplikací. Pro snazší analýzu dat by byla přínosná možnost nastavení očekávaných hodnot časů mezi kliknutími na prvky grafického uživatelského rozhraní. Dalším možným rozšířením je zobrazení tzv. *klikacích map*, tedy obrázků uživatelského rozhraní sledované aplikace s grafickým znázorněním míst, na které uživatelé aplikace klikají.

Literatura

[1] ISO 9241-210:2010 Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.

[2] ČSN ISO/IEC 9126-1 Softwarové inženýrství – Jakost produktu – Část 1: Model jakosti. listopad 2002.

[3] ISO/IEC TR 25060:2010 Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for usability: General framework for usability-related information. 2010.

[4] ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. 2011.

[5] Object.GetHashCode Method. [online], [cit. 24. 5. 2015].
URL https://msdn.microsoft.com/cs-cz/library/system.object.gethashcode%28v=vs.110%29.aspx

[6] ABRAN, A.: *Software Metrics and Software Metrology*. New Jersey: Wiley Publishing, Inc., 2010, ISBN 978-0-470-59720-0.

[7] BARAY, C.: the model-view-controller (MVC) design pattern. [online], [cit. 5. 1. 2015].
URL http://cristobal.baray.com/indiana/projects/mvc.html

[8] BROOKE, J.: SUS - A quick and dirty usability scale. [online], [cit. 6. 1. 2015].
URL http://www.usabilitynet.org/trump/documents/Suschapt.doc

[9] CONNELL, A.: 9 Google Analytics Alternatives. [online], březen 2014 [cit. 4. 1. 2015].
URL http://www.searchenginejournal.com/9-google-analytics-alternatives/92071/

[10] COSTA, J.: Calculating Geometric Means. [online], [cit. 2. 5. 2015].
URL http://www.buzzardsbay.org/geomean.htm

[11] CRESWELL, J. W.; CLARK, V. L. P.: *Designing and Conducting Mixed Methods Research*. SAGE Publications, druhé vydání, 2011 [cit. 20. 3. 2015], ISBN 9781412975179.

[12] DAHLGAARD, J. J.; KRISTENSEN, K.; KANJI, G. K.: *Fundamentals of Total Quality Management*. Londýn: Chapman-Hall, 2002, ISBN 0-7487-7293-6.

[13] DESKMETRICS: Getting Started. [online], [cit. 4. 1. 2015].
URL http://deskmetrics.com/docs/getting-started/

[14] DESKMETRICS: SDK for Windows. [online], [cit. 4. 1. 2015].
URL http://deskmetrics.com/docs/sdk/windows/

[15] EELES, P.: Capturing Architectural Requirements. [online], listopad 2005 [cit. 28. 12. 2014].
URL http://www.ibm.com/developerworks/rational/library/4706.html

[16] FENECH, K.: Tracking Desktop Applications with Google Analytics – what you should know... [online], listopad 2012 [cit. 22. 10. 2014].
URL http://blog.trackerbird.com/content/tracking-desktop-applications-with-google-analytics-what-you-should-know/

[17] GARVIN, D. A.: *Managing Quality: The Strategic and Competitive Edge*. New York: Free Press, únor 1988, ISBN 978-0029113806.

[18] GOOGLE: Measurement Protocol Developer Guide. [online], červenec 2014 [cit. 4. 1. 2015].
URL https://developers.google.com/analytics/devguides/collection/protocol/v1/devguide

[19] GOOGLE: Measurement Protocol Reference. [online], červenec 2014 [cit. 4. 1. 2015].
URL https://developers.google.com/analytics/devguides/collection/protocol/v1/reference

[20] GOOGLE: Mobilní analýza. [online], [cit. 4. 1. 2015].
URL http://www.google.com/intl/cs_ALL/analytics/features/mobile.html

[21] KENETT, R. S.; BAKER, E.: *Software Process Quality: Management and Control*. CRC Press, 1999, ISBN 978-0-8247-1733-9.

[22] KRUG, S.: *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. Berkeley: New Riders, 2010, ISBN 978-0-321-65729-9.

[23] MACDONALD, M.; FREEMAN, A.: *Pro ASP.NET 4 in C# 2010*. New York: Apress, 2010, ISBN 978-1-4302-2529-4.

[24] MORVILLE, P.: User Experience Design. [online], červen 2004 [cit. 1. 11. 2014].
URL http://semanticstudios.com/user_experience_design/

[25] NIELSEN, J.: Usability 101: Introduction to Usability. [online], leden 2012 [cit. 7. 11. 2014].
URL http://www.nngroup.com/articles/usability-101-introduction-to-usability/

[26] NIELSEN NORMAN GROUP: Usability Testing & UX Research. [online], [cit. 6. 1. 2015].
URL http://www.nngroup.com/consulting/ux-research-usability-testing/

[27] Paganelli, L.; Paterno, F.: Tool for remote usability evaluation of Web applications through browser logs and task models. *Behavior Research Methods*, ročník 35, č. 3, srpen 2003: s. 369–378, ISSN 1554-3528.
URL http://download.springer.com/static/pdf/390/art%253A10.3758%252FBF03195513.pdf?auth66=1423905093_1235de200a0f03e83f31c0f407ae126d&ext=.pdf

[28] PRCHAL, M.: Integrovaný management a úloha norem ISO řady 9000. *Perspektivy jakosti*, ročník III, č. 3, září 2006 [cit. 3. 12. 2014]: s. 20–23, ISSN 1214-8865.

[29] ROUDENSKÝ, P.; HAVLÍČKOVÁ, A.: *Řízení kvality softwaru: Průvodce testováním*. Brno: Computer Press, 2013, ISBN 978-80-251-3816-8.

[30] RUBIN, J.; CHISNELL, D.: *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Indianapolis: Wiley Publishing, Inc., druhé vydání, 2008, ISBN 978-0-470-18548-3.

[31] SANFORD, M.: Is user experience design (UXD) equal to user centered design (UCD)? [online], říjen 2012 [cit. 30. 12. 2014].
URL http://ux.stackexchange.com/a/28256

[32] SAURO, J.: 10 Essential Usability Metrics. [online], listopad 2011 [cit. 18. 5. 2015].
URL http://www.measuringu.com/blog/essential-metrics.php

[33] SAURO, J.: 3 Ways To Combine Quantitative And Qualitative Research. [online], duben 2015 [cit. 2. 5. 2015].
URL http://www.measuringu.com/blog/mixing-methods.php

[34] SAURO, J.; LEWIS, J. R.: Average Task Times in Usability Tests: What to Report? *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010 [cit. 2. 5. 2015]: s. 2347–2350, doi:10.1145/1753326.1753679.
URL http://doi.acm.org/10.1145/1753326.1753679

[35] SAURO, J.; LEWIS, J. R.: *Quantifying the User Experience: Practical Statistics for User Research*. Waltham: Elsevier, 2012, ISBN 978-0-12-384968-7.

[36] STATHAT: Welcome to StatHat! [online], [cit. 4. 1. 2015].
URL https://www.stathat.com/v

[37] TRACKERBIRD: Features. [online], [cit. 22. 10. 2014].
URL http://www.trackerbird.com/features/

[38] U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES. Office of the Assistant Secretary for Public Affairs. Digital Communications Division: Usability Evaluation Basics. [online], říjen 2013 [cit. 7. 11. 2014].
URL http://www.usability.gov/what-and-why/usability-evaluation.html

[39] U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES. Office of the Assistant Secretary for Public Affairs. Digital Communications Division: User Experience Basics. [online], únor 2014 [cit. 30. 12. 2014].
URL http://www.usability.gov/what-and-why/user-experience.html

[40] U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES. Office of the Assistant Secretary for Public Affairs. Digital Communications Division: System Usability Scale (SUS). [online], [cit. 6. 1. 2015].
URL http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[41] WAGNER, S.: *Software Product Quality Control*. Heidelberg: Springer, 2013, ISBN 978-3-642-38571-1, doi:10.1007/978-3-642-38571-1.

[42] WILLIAMS, A.: User-centered Design, Activity-centered Design, and Goal-directed Design: A Review of Three Methods for Designing Web Applications. In *Proceedings of the 27th ACM International Conference on Design of Communication*, SIGDOC '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-559-8, s. 1–8, doi:10.1145/1621995.1621997.
URL http://doi.acm.org/10.1145/1621995.1621997

Příloha A

Obsah CD

Přiložené CD obsahuje:

- text diplomové práce ve formátu PDF,
- obrázky a zdrojové soubory LaTeX diplomové práce,
- zdrojové soubory klientské i serverové části nástroje pro měření použitelnosti software.

Struktura souborů na CD je následující:

Adresář	**Obsah adresáře**
/text_prace/	Text diplomové práce ve formátu PDF
/text_prace/src/	Obrázky a zdrojové soubory LaTeX diplomové práce
/usability_logger/UsabilityLoggerClient	Zdrojové soubory klientské části nástroje
/usability_logger/UsabilityLoggerServer	Zdrojové soubory serverové části nástroje