Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program : Aplikovaná informatika

Obor: Informační systémy a technologie

Testování webových služeb nástrojem SoapUI

DIPLOMOVÁ PRÁCE

Student : Bc. Petr Sobotka

Vedoucí : doc. Ing. Alena Buchalcevová, Ph.D.

Oponent : Ing. Jan Holoubek

2015

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 6. 12. 2015 . . . . . .

Bc. Petr Sobotka

Poděkování

Děkuji doc. Ing. Aleně Buchalcevové, Ph.D., za cenné rady a věcné připomínky po dobu vedení této diplomové práce. Dále bych rád poděkoval Ing. Davidu Šimkovi za poskytnuté rady při vypracovávání metodiky v nástroji EPFC.

Abstrakt

Předmětem této diplomové práce je testování webových služeb nástrojem SoapUI. Hlavním cílem této práce je vytvoření metodiky pro testování webových služeb. Dalším cílem, který je součástí metodiky je vytvoření příručky pro testování webových služeb nástrojem SoapUI. Příručka slouží jako manuál pro testování webových služeb v nástroji SoapUI.

Teoretická část práce je věnována objasnění termínů testování, webová služba a metodika. V rámci této části jsou také představeny nástroje SoapUI a Eclipse Process Framework Composer. Praktická část je věnována samotné metodice testování webových služeb. Metodika je vytvořena v nástroji Eclipse Process Framework Composer a každý element této metodiky je zde popsán. Některé elementy metodiky (příručka, koncepty, nástroje a vybrané šablony) jsou uvedeny zvlášť v přílohách práce.

Klíčová slova

Webová služba, SoapUI, testování, metodologie, EPFC, MMSP.

Abstract

The subject of this diploma thesis is web services testing with SoapUI. The main objective is to create a methodology for web services testing. The next objective (which is a part of the methodology) is to create a guidance for web services testing using SoapUI. The guidance serves as a manual to web services testing with SoapUI.

The theoretical part of the thesis explains basic terms: testing, web service and methodology. This part is also devoted to the introduction of the SoapUI and Eclipse Process Framework Composer. The practical part of the thesis is focused on the methodology for web services testing itself. The methodology is created in Eclipse Process Framework Composer and each element of the methodology is described here. Some of the methodology elements (the guidance, concepts, tools and some templates) are listed as an appendix.

Keywords

Web service, SoapUI, testing, methodology, EPFC, MMSP.

Obsah

1 Úvod

Webové služby (angl. Web Services) jsou v dnešní době pojmem, s kterým se lze v informačních technologiích setkávat stále častěji. Webové služby dnes již nejsou využívány pouze jako integrační prvek v distribuovaných systémech, ale stále častěji se o webových službách mluví v kontextu služebně orientovaných výpočtů (SOC, Service Oriented Computing), respektive služebně orientované architektury (SOA, Service Oriented Architecture) (Barry a Dick, 2013; Erl, 2008).

Testování se již stalo neodmyslitelnou součástí procesu vývoje softwaru. S tím, jak jsou webové služby stále ve větší míře využívány, souvisí potřeba jejich adekvátního testování. Webové služby s sebou přinášejí nové výzvy a jejich testování je pokládáno (v porovnání s testováním klasických řešení) za náročnější (Bozkurt, Harman a Hassoun, 2010). Tomu, jak je možné webové služby testovat, se věnuje tato diplomová práce, v rámci které je vypracována metodika Testování webových služeb nástrojem SoapUI, společně s příručkou, která slouží jako manuál pro tvorbu testů webových služeb v nástroji SoapUI.

1.1 Vymezení tématu práce a důvod výběru tématu

Diplomová práce se věnuje tématu testování webových služeb se zaměřením na testování pomocí nástroje SoapUI. V rámci této práce je navržena metodika Testování webových služeb nástrojem SoapUI, společně s příručkou pro tvorbu testů v nástroji SoapUI. Samotná metodika je vytvořena v nástroji Eclipse Process Framework Composer, který slouží pro správu metodik.

Toto téma jsem si zvolil proto, že již třetím rokem pracuji na pozici testera a zavádění testování webových služeb byla jedna z oblastí, které jsem se na této pozici věnoval. Testování webových služeb je zároveň oblast, která je v dnešní testovací praxi vedle automatizace front-end rozhraní (například pomocí nástroje Selenium) ceněná a vyhledávaná.

1.2 Cíle práce

Hlavním cílem diplomové práce je vytvořit metodiku testování webových služeb, která poskytuje ucelený návod, jak k testování webových služeb přistupovat.

Mezi dílčí cíle patří:

- provedení rešerše oblasti testování webových služeb,
- objasnění pojmů testování, webová služba a metodika,
- představení nástroje SoapUI,

- návrh metodiky Testování webových služeb nástrojem SoapUI,
- vytvoření návodů pro práci s nástrojem SoapUI,
- implementace navržené metodiky v nástroji Eclipse Process Framework Composer (EPFC) a její publikace.

1.3 Předpoklady a omezení práce

Diplomová práce se zaměřuje na testování webových služeb nástrojem SoapUI. Práce se nevěnuje testování řešení služebně orientovaných výpočtů, respektive služebně orientované architektury, ale zaměřuje se především na webové služby jako takové.

Metodika Testování webových služeb nástrojem SoapUI je vytvořena a publikována jako samostatná metodika v nástroji EPFC, který slouží pro snadnou správu metodik. Vypracovávaná metodika vychází z metodiky MMSP, kterou vypracovala Petra Rejnková (2010). Metodika Testování webových služeb nástrojem SoapUI je zaměřena pouze na testování webových služeb. Metodiku MMSP tak upravuje a rozšiřuje o prvky podstatné pro testování webových služeb.

Verze nástroje SoapUI, se kterou je v rámci příručky pro tvorbu testů webových služeb v SoapUI pracováno, je SoapUI OS. SoapUI OS bylo vybráno především proto, že se jedná o verzi poskytovanou zdarma. Dále také proto, že nabízí většinu funkcionalit komerční verze nástroje SoapUI NG Pro (jednotlivým verzím a jejich srovnání se blíže věnují kapitoly 5.2.1 a 5.2.2 v této práci).

Pro účely tvorby příkladů v příručce k nástroji SoapUI je využita webová služba „Geo Services“, která je volně k dispozici na https://www.geosvc.com/. Služba Geo Services umožňuje vrátit informace o místě na základě PSČ, vzdálenost mezi dvěma lokacemi, vrátit lokace na základě PSČ nebo vrátit města a místa blízko určité lokace. Tato webová služba byla vybrána proto, že:

- je poskytována zdarma (s určitými omezenými jako je počet dotazů za den nebo maximální vzdálenost mezi městy),
- využívá SOAP i REST,
- pro komunikaci vyžaduje ověření,
- odpovědi pro SOAP jsou vraceny jako XML a odpovědi REST jsou vraceny v XML či JSON.

Jedná se tedy o webovou službu, na které je možné názorně předvést například rozdíly mezi SOAP a REST, či ukázat, jak je možné v dotazu na službu ověřovat identitu žadatele.

1.4 Struktura práce

V první části práce je provedena nezbytná rešerše poznání oblasti testování webových služeb. Následuje objasnění hlavních a dílčích pojmů z oblastí testování a webových služeb. Další část práce je věnována specifikům testování webových služeb. V rámci této kapitoly je také představen nástroj SoapUI. Následující kapitola se věnuje obecně metodikám vývoje softwaru. V rámci této kapitoly jsou vysvětleny hlavní pojmy a představen nástroj pro správu metodik EPFC. Předposlední část práce je věnována metodice Testování webových služeb pomocí nástroje SoapUI. Na počátku je představena metodika MMSP, ze které vypracovávaná metodika vychází. Následuje představení metodiky Testování webových služeb nástrojem SoapUI společně s popisem všech jejích prvků. Poslední kapitolou je závěr, ve kterém jsou shrnuty poznatky této diplomové práce.

Součástí práce jsou i přílohy, které obsahují vybrané části metodiky, jež nebyly vzhledem k jejich rozsahu detailně popsány v hlavní části práce, ale přesto jsou pro metodiku důležité. Jedná se především o příručku k tvorbě testů v nástroji SoapUI, koncepty, nástroje a vybrané šablony.

1.5 Přínosy práce

Hlavním přínosem této diplomové práce je vytvoření metodiky pro testování webových služeb pomocí nástroje SoapUI. Dalším přínosem práce je vytvoření příručky k nástroji SoapUI, v níž je uvedena řada návodů a postupů pro práci s tímto nástrojem, které vycházejí z praktických zkušeností autora s prací v tomto nástroji.

Tato metodika může být aplikovatelná v praxi a sloužit pro zavedení testování webových služeb tam, kde s jejich testováním nejsou zatím žádné nebo malé zkušenosti. Zároveň může tato metodika nebo její části sloužit jako výukový materiál.

2 Rešerše

Obsahem této kapitoly je zmapování a charakterizování současného stavu zkoumané oblasti.

Webovým službám se věnuje velké množství autorů. Jak již bylo zmíněno v úvodu, je to dáno především tím, že v posledních několika letech jsou webové služby stále častěji implementovány jako prvek architektury. Oblasti testování webových služeb se však taková pozornost nevěnuje. Ještě horší situace je v České republice, kde publikací a článků k této problematice dostupných v českém jazyce je poskrovnu.

2.1 Odborné knihy

Mezi přední zahraniční publikace věnující se testování webových služeb nástrojem SoapUI patří kniha „Web Services Testing with SoapUI“ od autora Charitha Kankanamge (2012), která je zaměřena především na možnosti nástroje SoapUI. V několika kapitolách vysvětluje na praktických ukázkách základní, ale i pokročilé funkcionality nástroje. Autor se z počátku knihy také věnuje popisu toho, co webové služby jsou. Nicméně se již nevěnuje metodice testování webových služeb, ale pouze nastiňuje přístup k jejich testování a možné problémy, s kterými se lze při testování webových služeb setkat.

Nástroji SoapUI se dále věnuje kniha „SoapUI Cookbook“ od autora Rupeta Andersona (2015). Na rozdíl od publikace „Web Services Testing with SoapUI“ se však v této knize autor věnuje pouze nástroji SoapUI. Kniha slouží jako rozsáhlá sada návodů, jak něco vytvořit v SoapUI.

Webovým službám se komplexně věnuje práce „Web Services Principles and Technology“ Michaela P. Papazoglou (2008). Autor se věnuje především konceptu webových služeb a tomu, jak je možné webové služby aplikovat v organizaci. V kapitole 8 se menší část věnuje testování webových služeb, nezachází však do detailů a pouze nastiňuje možné fáze testování a jejich obsah. V roce 2012 vyšla aktualizovaná verze publikace s názvem „Web services and SOA: principles and technology“ (Papazoglou, 2012), kde se autor věnuje více architektuře SOA.

Publikace „Web Services, Service-Oriented Architectures, and Cloud Computing“ autorů Douglas K. Barry a David Dick (2013) se nevěnuje pouze webovým službám, ale i architektuře SOA a cloud computingu. Autoři se v publikaci věnují i vývoji webových služeb z historického hlediska a snaží se je zasadit do kontextu technologií dnešní doby (SOC, SOA, Cloud computing).

Mezi české knihy věnující se oblasti testování patří „Řízení kvality softwaru“ autorů Roudenského a Havlíčkové (2013). Jedná se o novější publikaci, která se zaměřuje na oblast

testování obecně a pokrývá témata od základních konceptů testování až po management testování nebo automatizaci.

Tvorbě informačních systémů se věnuje publikace „Tvorba informačních systémů: principy, metodiky, architektury“ autorů Bruckner, Voříšek, Buchalcevová, Stanovská, Chlapek, Řepa (2012). Publikace se mimo jiné zabývá vývojem a provozem IS, metodickým přístupem k tvorbě IS/ICT, architektuře a metodice MMDIS, která je vyvíjena katedrou informačních technologií na VŠE.

Na publikaci „Tvorba informačních systémů: principy, metodiky, architektury“ navazuje kniha věnující se tvorbě softwarových aplikací „Příklady modelů analýzy a návrhu aplikace v UML“ od autorek Buchalcevové a Stanovské (2013). Publikace se věnuje vývoji softwaru, popisu metodiky MMSP a obsahuje praktické příklady analýzy a návrhu aplikace dvou ukázkových projektů.

Metodikám budování IS/ICT se věnuje publikace Buchalcevové (2005) s názvem „Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky“. V knize jsou metodiky pro vývoj a údržbu IS kategorizovány a popsány s tím, že hlavní část knihy je zaměřena na rigorózní a agilní metodiky. V závěru knihy je část věnována metodickému rámci pro budování informačního systému MeFIS.

2.2 Akademické práce

Testování webových služeb se ve své bakalářské práci věnovala Zdeňka Repáňová (2014) z Univerzity Pardubice. Autorka se v práci nejprve věnovala teorii webových služeb a testování. V další části porovnala 4 dostupné nástroje pro testování webových služeb (SoapUI, SOAPSonar, Examine a Storm), ze kterých jako nejvhodnější vzešel SoapUI. V závěrečné části autorka navrhla testovací plán pro vybranou webovou službu, vytvořila testy v SoapUI, které následně provedla a vyhodnotila.

Metodikou životního cyklu softwaru se zabývá řada akademických prací. Zde bych vyzdvihl především diplomovou práci Petry Rejnkové „Lokalizace a přizpůsobení metodiky OpenUP“ (2010) z Vysoké školy ekonomické v Praze. Autorka v práci vytvořila metodiku MMSP, která vychází z metodiky OpenUP, je určena pro malé a střední projekty a pokrývá životní cyklus vývoje softwaru od zahájení až po nasazení. Metodika byla vytvořena pomocí nástroje EPFC a je publikována na stránkách http://mmsp.czweb.org/.

Diplomová práce „Testování a kvalita softwaru v metodikách vývoje softwaru“ Vladana Vachalce (2014) z Vysoké školy ekonomické v Praze metodiku MMSP v mnoha ohledech rozšiřuje. Autor se zaměřil především na oblast testování tak, aby byla metodika použitelná i ve větších týmech.

Oblastí testování webových služeb se zabývala bakalářská práce „Testování atomických softwarových služeb“ Lenky Sládkové (2013) z Vysoké školy ekonomické v Praze. Předmětem práce bylo testování atomických služeb s cílem zhodnotit, jak tomuto testování vyhovuje metodika RUP. Testován byl software, jehož základ integrační platformy byl tvořen atomickými službami. V práci bylo zjištěno, že metodika RUP byla na tomto projektu při testování atomických služeb využita pouze z 67,2%.

Architektuře orientované na služby (SOA) se věnovala disertační práce Romana Hauptvogela (2013). Cílem disertace bylo rozšíření metodického rámce MeFIS o metodický vzor „Budování IS v prostředí architektury orientované na služby“. V rámci této práce se autor věnoval webovým službám a konceptu servisně orientované architektury (SOA).

2.3 Ostatní zdroje

Studie „Testing web services: a survey“ (Bozkurt, Harman a Hassoun, 2010) se zaměřuje na techniky a metody testování webových služeb. Studie identifikuje již existující metody a přístupy v testování webových služeb a snaží se je představit a kategorizovat.

Článek Martina Kuby (2006) se komplexně věnuje technologii webových služeb. Autor se zde nevěnuje pouze popisu webových služeb, ale i technologiím s tím spojeným (XML, WSDL, SOAP, REST, ESB).

Základním informačním zdrojem o nástroji SoapUI jsou jeho internetové stránky http://www.soapui.org provozované aktuálním vlastníkem nástroje SoapUI – firmou SmartBear Software, Inc. Na těchto stránkách je zvláštní sekce o nástroji SoapUI a jeho jednotlivých funkcích (Smartbear, 2015). Zároveň firma SmartBear poskytuje zdarma i elektronickou knihu „SoapUI 101: Beginner’s Guide to Functional Testing“ (Smartbear, 2013), která obsahuje popis a návody základních funkcionalit nástroje SoapUI.

Na internetu jsem nenašel webový portál, který by se zabýval přímo testováním webových služeb. Nicméně i tak je zde možné najít články nebo sérii článků, které se testováním webových služeb zaobírají. Sérii článků o testování webových služeb například publikoval server Guru99 (2015). V těchto článcích je čtenář ve zkratce seznámen s tím, co testování webových služeb obnáší, zároveň je tu představen i nástroj SoapUI a uvedeno, jak s ním pracovat.

Z českých serverů vydal sérii článků o práci v nástroji SoapUI s názvem „SoapUI prakticky“ server „SW Testování“ (2009). Bohužel některé články jsou již ze staré verze SoapUI, ale i tak se jedná o zajímavou sérii návodů pro práci se SoapUI.

3 Testování softwaru

Předmětem této kapitoly je zasazení oblasti testování do procesu vývoje softwaru a vysvětlení základních pojmů s testováním souvisejících.

3.1 Testování a zajišťování kvality

Pojem testování definuje Roudenský a Havlíčková (2013) jako *„proces řízeného spouštění softwarového produktu s cílem zjistit, zda splňuje specifikované či implicitní potřeby uživatelů“*.

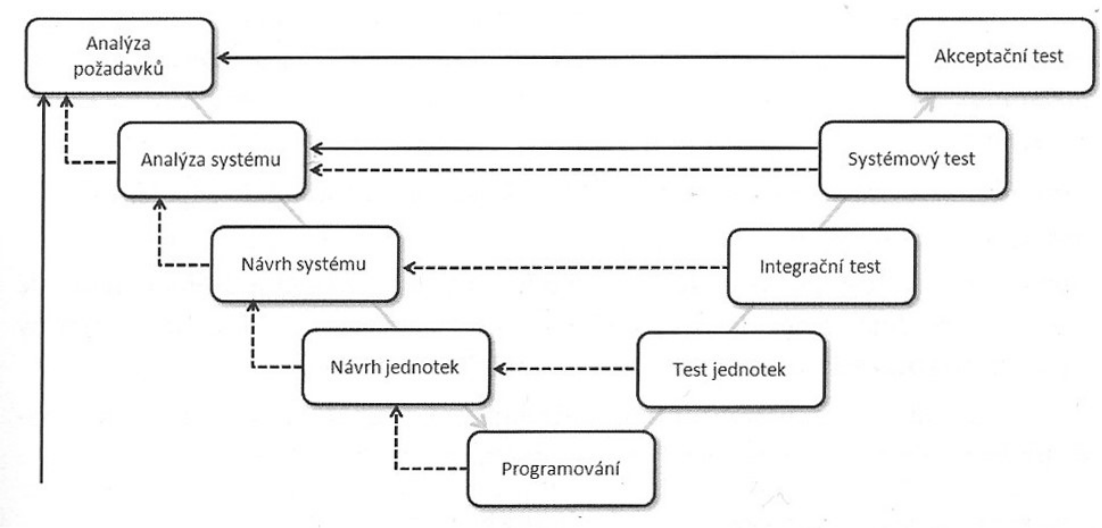
Z této definice vyplívá, že testování není pouze činnost, která má za cíl nalézat chyby ve zkoumaném produktu, ale že obsahem testování je především sběr informací, za účelem ověření toho, že zkoumaný produkt vyhovuje zamýšlenému použití (Roudenský a Havlíčková, 2013).

S pojmem testování úzce souvisí termín *zajišťování kvality*. Zajišťování kvality neboli QA (Quality Assurence), ale není synonymem k testování, jak by se na první pohled mohlo zdát. QA je rozsáhlejší oblast než testování a měla by prostupovat všemi procesy vývoje softwaru. QA definuje standardy, metodiky a metriky, které by měly vést k vyšší kvalitě produktu. Testování je tedy v rámci QA pouze jednou z oblastí (Roudenský a Havlíčková, 2013).

3.2 Testování v rámci procesu vývoje softwaru

Vývoj softwaru je velmi komplikovaný proces, ve kterém hraje testování důležitou roli. Pokud se testování zanedbá či dokonce vynechá, zákazník může dostat produkt, který obsahuje velké množství chyb a který nemusí plnit účel, pro nějž byl vytvořen. Utrpět tím může zákazník i dodavatel řešení, zákazník například ztrátou důvěry v produkt a dodavatel ve ztrátě reputace. I přes to je však testování oblastí, která je podceňována a na které se snaží někteří výrobci softwaru ušetřit (Roudenský a Havlíčková, 2013).

Roudenský a Havlíčková (2013) identifikují čtyři základní úrovně testování, s kterými se lze v procesu vývoje softwaru setkat. Tyto úrovně vycházejí z V-modelu, jenž je modelem životního cyklu vývoje softwaru. Z V-modelu je patrné (viz obrázek 1), že každé fázi vývoje odpovídá určitá fáze testování.



Obrázek 1: *V-model (Zdroj: Roudenský a Havlíčková, 2013)*

Popis jednotlivých fází testování odpovídajících V-modelu:

1. **Unit testování** – nebo také „Testování jednotek“. Jedná se o testy na první úrovni, nejčastěji je provádí programátor, který danou část vyvinul. Obsahem testování jsou malé části programu (jednotky) s cílem ověřit správné chování každé jednotky nezávisle na ostatních jednotkách (Roudenský a Havlíčková, 2013).

   V rámci této úrovně se prováděj zejména unit testy.

2. **Integrační testování –** po otestování samostatných částí systému (jednotek) je možné testovat systém jako celek. S tímto testováním lze začít, jakmile je připraveno dostatečné množství modulů (minimálně však dva). Cílem je ověřit správné napojení jednotlivých modulů (Roudenský a Havlíčková, 2013).

   V rámci této úrovně se provádějí zejména assembly testy.

3. **Systémové testování –** jedná se o klíčovou fázi testování, v rámci které dochází ke komplexnímu ověření aplikace. Jedná se o poslední úroveň testů, než je aplikace předána k testování zákazníkem. Cílem je ověřit, zda produkt splňuje požadavky specifikované zákazníkem (Roudenský a Havlíčková, 2013).

   V rámci této úrovně se provádějí testy funkčních a nefunkčních požadavků. Jedná se zejména o funkční testy, regresní testy, výkonnostní testy či bezpečnostní testy.

4. **Akceptační testování –** jde o závěrečnou fázi testování, bývá obvykle označována jako UAT (User Acceptance Testing). V této fázi produkt testuje zákazník (respektive uživatelé nebo testeři za stranu zákazníka), s cílem, zda produkt splňuje akceptační kritéria specifikovaná zákazníkem (Roudenský a Havlíčková, 2013).

   V rámci této úrovně se provádějí zejména akceptační testy.

3.2.1 Typy testů

V této části jsou popsány základní typy testů, se kterými se lze v procesu testování obvykle setkat.

Mezi základní typy testů vycházejících z výše uvedeného V-modelu patří:

- **Unit testy –** testy izolovaných malých programových modulů (jednotek). Unit testy obvykle tvoří a provádí programátor. Vzhledem ke své povaze jsou jednotkové testy automatizované (Roudenský a Havlíčková, 2013).
- **Assembly testy –** testy zaměřené na interakci mezi moduly, někdy také nazývané integrační testy. Účelem těchto testů je ověření, zda moduly mezi sebou komunikují a chovají se dle požadavků specifikovaných zákazníkem (Dasari, 2010).
- **Funkční testy –** jejich účelem je ověřit, zda systém splňuje specifikované nebo implicitní požadavky (Roudenský a Havlíčková, 2013).
- **Regresní testy –** testy zaměřené na opětovné testování již dříve vyvinutých funkcionalit systému. Jejich účelem je ověření, zda při určité změně v systému (například implementaci nové funkčnosti) zůstanou zachovány vlastnosti a funkce stávajícího systému. Regresní testy mohou být obzvláště u větších projektů časově velmi náročné, proto se lze u tohoto typu testů často setkat s automatizací (Roudenský a Havlíčková, 2013).
- **Akceptační testy –** předmětem akceptačních testů je ověření, zda dodaný produkt splňuje akceptační kritéria, což jsou podmínky, které jsou společně se zákazníkem specifikovány na počátku projektu (Roudenský a Havlíčková, 2013).
- **Bezpečnostní testy** – jsou testy, jejichž cílem je zjištění zranitelných míst v systému (lze se také setkat s pojmy *Penetrační testování* nebo *Ethical Hacking*). Výsledkem provedení takovýchto testů by mělo být doporučení pro nápravu nalezených zranitelných míst (Hanzlíková, 2013).
- **Výkonnostní testy –** jsou testy, jejichž účelem je zjištění, do jaké míry testovaný systém naplňuje funkce v rámci daných omezení času a dalších zdrojů (Vodrážka, 2015).

  U výkonnostních testů lze identifikovat dva základní typy:

  1. **Zátěžový test (Load Test) –** testování systému ve zvýšené nebo dlouhodobé nominální zátěži, která je v praxi očekávána. Mezi sledované parametry patří počet požadavků za sekundu, doba odezvy, propustnost nebo využití zdrojů (Vodrážka, 2015).
  2. **Test hraniční zátěže (Stress Test)** – testování systému v hraničních podmínkách. Sleduje se chování systému s cílem určit hranici, za kterou již systém není

schopen pracovat. Mezi sledované parametry patří počet požadavků za sekundu, doba odezvy, propustnost nebo využití zdrojů (Vodrážka, 2015).

Mimo výše uvedené základní typy testů se lze setkat i s následujícími:

- **Smoke testy –** účelem smoke testů je ověřit stabilitu a hlavní části (funkce) produktu. Velmi často se jedná o automatizované testy, které jsou spouštěny například po nasazení nového sestavení aplikace (Roudenský a Havlíčková, 2013).
- **End-to-End testy –** testy, které mají za účel sledování životního cyklu určitého objektu napříč systémem. V rámci životního cyklu se kontroluje správnost jednotlivých interakcí, přechodů a v neposlední řadě konečného stavu (Roudenský a Havlíčková, 2013).

3.2.2 Chyba v softwaru

Chyba v softwaru (angl. bug) označuje chování, které Patton (2002) definoval ve vztahu k specifikaci produktu takto:

- Software nedělá něco, co by podle specifikace produktu dělat měl.
- Software dělá něco, co by podle údajů specifikace produktu dělat neměl.
- Software dělá něco, o čem se produktová specifikace nezmiňuje.
- Software nedělá něco, o čem se produktová specifikace nezmiňuje, ale měla by se zmiňovat.
- Software je obtížně srozumitelný, těžko se s ním pracuje, je pomalý, nebo – podle názoru testera softwaru – jej koncový uživatel nebude považovat za správný.

Specifikací produktu je rozuměna specifikace, ve které je produkt definován. Specifikace popisuje, jak bude produkt vypadat, jak se bude chovat, co bude a nebude dělat (Patton, 2002).

3.2.3 Manuální a automatizované testování

Základní rozdíl mezi těmito dvěma přístupy spočívá v tom, že manuální testování je prováděno člověkem (testerem), zatímco k provedení automatizovaného testování je používán nějaký software (Roudenský a Havlíčková, 2013).

V případě manuálního testování prochází tester předem připravené testovací případy a porovnává aktuální chování aplikace s očekávaným chováním popsaném v testovacím případu. V případě automatizované testování je připraven testovací skript v daném softwaru, který dle skriptu vykoná sadu příkazů, pomocí kterých například ověří chování aplikace. Automa-

tizované testování tak v určitých případech může přinést úsporu nejen v čase stráveném testováním, ale také v nákladech na testování (často se s automatizovaným testováním lze setkat u regresních nebo smoke testů).

Díky tomu je automatizované testování v dnešní době velmi populární (Roudenský a Havlíčková, 2013). Nicméně je třeba poznamenat, že ne vždy je automatizované testování výhodnější než manuální, oba přístupy mají své výhody a nevýhody.

Mezi výhody automatizovaného testování patří již zmiňovaná úspora času a nákladů. Nevýhodou ovšem je, že ne vše lze automatizovat (jedná se například o vnímání použitelnosti aplikace, estetiky a dalších mimofunkčních požadavků) a někdy může být režie spojená s automatizovaným testováním vyšší než přínos. Člověk jako tester se navíc dokáže lépe vyrovnat s nestandardními situacemi, dokáže chybu okamžitě analyzovat a nalézat chyby, které automatizovaným testem není lehké podchytit (Roudenský a Havlíčková, 2013).

4 Webové služby

Před uvedením, jak je možné webové služby testovat, je nejprve nutné objasnit samotný pojem *webová služba* (v angličtině *web service*).

Standardizační organizace W3C (2004) definuje webovou službu jako „*softwarový systém pro podporu interakce strojů prostřednictvím sítě. Webová služba má popsáno své rozhraní ve strojově zpracovatelném formátu (konkrétně WSDL). Ostatní systémy s webovou službou komunikují na základě jejího popisu za pomoci SOAP zpráv, obvykle přes HTTP s využitím XML a dalších webových standardů.*“.

Z definice je patrné její zaměření na technickou stránku webových služeb. Papazoglou (2008) definuje pojem webová služba obecněji, a to jako stavební blok pro vytváření distribuovaných aplikací, které mohou být publikovány a konzumovány prostřednictvím Internetu či intranetových sítí. Webové služby jsou postaveny na otevřených webových standardech a je pro ně typické, že na jedné straně je poskytovatel služby, který vystavuje rozhraní, a na druhé straně příjemce služby, který webovou službu konzumuje.

Autoři Breshne, Fuhrer a Pasquier (2009) dále identifikovali pět základních charakteristik, kterými se webové služby vyznačují. Jedná se o:

- **XML** – webové služby využívají formátu XML jak pro prezentaci dat, tak pro komunikaci. XML zajišťuje webových službám jejich nezávislost na programovacím jazyku nebo například na operačním systému.
- **volná vazba** (anglicky loosely coupled) – webové služby mají mezi sebou volnou vazbu. Pokud se například změní vnitřní logika webové služby, neovlivní to subjekt, který takovou webovou službu využívá, protože se nezmění stávající rozhraní.
- **synchronní a asynchronní komunikace –** webové služby umožňují synchronní i asynchronní způsob komunikace. V synchronním typu klient zašle požadavek a čeká na odpověď, aniž by mohl provádět další činnosti. V případě asynchronní komunikace klient zašle požadavek, ale již nečeká na odpověď a pokračuje dál v další činnosti.
- **vzdálené volání (RPC) –** webové služby umožňují komunikovat se vzdálenými objekty (na základě XML postavených protokolech).
- **výměna dokumentů** – webové služby umožnují přenášet komplexní dokumenty (na základě XML) a ne pouze jednoduchá data.

Díky svým charakteristikám přinášejí webové služby řadu výhod. Mezi jejich hlavní výhody patří zejména

- nezávislost (na programovacím jazyku, platformě),
- interoperabilita (tedy schopnost systémů vzájemně spolupracovat) a

- znuvupoužitelnost (Ptáček, 2007; Kuba, 2006).

4.1 Architektura webových služeb

Specifikace webových služeb se skládá ze tří hlavních principů (Barry a Dick, 2013):

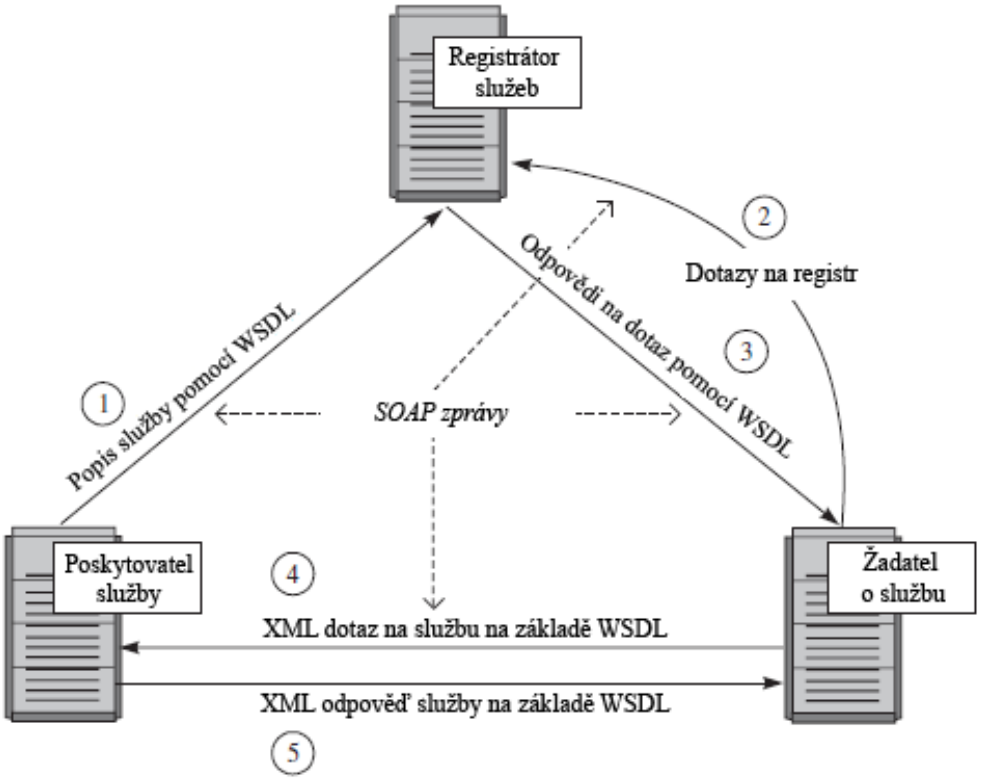
- **WSDL** pro popis rozhraní webové služby,
- protokolu **SOAP** pro komunikaci a
- adresáře **UDDI** pro registraci a vyhledávání webových služeb.

V případě UDDI (Universal Description, Discovery and Integration) je třeba poznamenat, že se dnes ve větší míře již nevyužívá. UDDI slouží pro registraci a vyhledávání služeb popsaných pomocí WSDL. Vzhledem k tomu, že registry UDDI se v řadě případů neimplementovaly, postupem času o UDDI opadl zájem. Obrázek 2 znázorňuje popularitu termínu UDDI dle Google Trends, který klesající zájem o UDDI dokládá.

Obrázek 2: *Popularita termínu Universal Description Discovery and Integration dle Google Trends (Zdroj: autor)*

Na místo UDDI se lze dnes setkat s repositáři webových služeb, které nejsou vázané tak přísnými pravidly implementace, jak tomu bylo v případě UDDI (Barry a Dick, 2013). Mezi populární repositáře webových služeb patří například web ProgrammableWeb dostupný na http://www.programmableweb.com/.

Následující obrázek 3 znázorňuje původně zamýšlený koncept webových služeb.



Obrázek 3: *Architektura webových služeb (Zdroj: Barry a Dick, 2013)*

Proces komunikace znázorněný na obrázku 3 pak probíhal následovně (Barry a Dick, 2013):

1. Poskytovatel služby zaregistroval službu popsanou pomocí WSDL u UDDI registrátora služeb.
2. Žadatel o službu zaslal jeden nebo více dotazů na registrátora služeb.
3. Žadateli o službu byl vrácen popis služby ve formě WSDL, který poskytovatel služby zveřejnil. Tento popis mimo jiné obsahoval, jak je možné se službou komunikovat.
4. Žadatel o službu na základě WSDL zaslal dotaz na službu poskytovatele služby.
5. Žadateli byla vrácena odpověď webové služby poskytovatele ve formátu, který byl popsán ve WSDL.

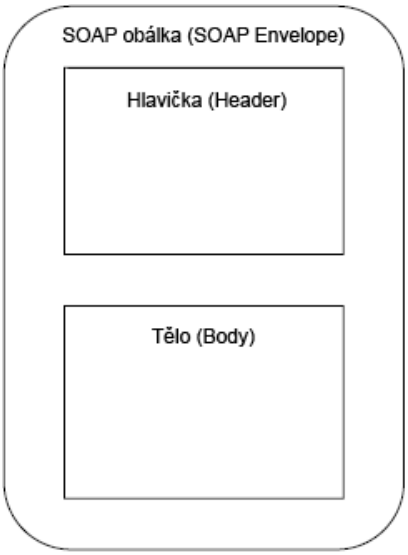
Jak lze vidět na obrázku 3, pro vlastní komunikaci s webovou službou není potřeba UDDI registr nebo repositář webových služeb. Zapotřebí je pouze WSDL popis webové služby a protokol SOAP.

Nicméně kombinace WSDL a SOAP není dnes jediný způsob komunikace s webovou službou. V dnešní době se lze setkat i s technologií REST, která na rozdíl od SOAP nevyužívá WSDL a nemusí komunikovat pomocí XML. Oba přístupy (SOAP i REST) však mají svá specifika, výhody a nevýhody. Popisu obou těchto přístupů jsou proto věnovány následující dvě kapitoly.

4.2 SOAP

Protokol pro zasílání zpráv mezi webovými službami. SOAP slouží jako obálka pro zprávy webových služeb v XML. O standardizaci se stará World Wide Web Consorcium (W3C). Původně akronym SOAP znamenal Simple Object Access Protocol, ale od verze 1.2 se od tohoto akronymu odstoupilo a zkratka SOAP již nemá žádný konkrétní význam (Breshne, Fuhrer a Pasquier, 2009; Kankanamge, 2012).

SOAP obálka se skládá ze dvou částí, hlavičky a těla (viz obrázek 4). Hlavička je volitelná část SOAP obálky a může nést informace jako je autentizace nebo kódování. Tělo SOAP obálky je povinné a obsahuje vlastní XML zprávu definovanou pomocí WSDL (Kankanamge, 2012).



Obrázek 4: *SOAP obálka (Zdroj: Kankanamge, 2012)*

Jako transportní protokol se u SOAP webových služeb obvykle využívá HTTP. SOAP ale není na tento protokol na rozdíl od REST přímo vázán, a je tak možné použít například i SMTP (Mueller, 2013). Zprávy SOAP jsou zasílány a přijímány výhradě ve formátu XML. Výhodou formátu zpráv v XML je jejich čitelnost pro člověka. Jejich nevýhodou je, že jsou například oproti formátu JSON zprávy v XML mnohem větší a tak náročnější na přenos.

Webové služby SOAP jsou rozšiřovány o standardy WS*, které například zvyšují bezpečnost. Jedná se například o WS-Security, WS-Reliability, WS-Transaction, WS-Addressing, WS-Policy, WS-MetadataExchange, WS-Notification, WS-Eventing Breshne (Fuhrer a Pasquier, 2009).

Dále je uveden ukázkový SOAP dotaz a odpověď.

Výpis 1: SOAP dotaz na službu Geo Services

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
```

```
   <soapenv:Header>
      <geos:ApiKey>1a0d749e646e46b1a459b10afb559112</geos:ApiKey>
   </soapenv:Header>
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>10065</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```

Výpis 1 obsahuje dotaz na službu Geo Services pro získání informací o místě na základě státu a PSČ. Ve výpisu lze vidět, že v elementu *Header* je uveden `<geos:ApiKey>`, který slouží k ověření identity.

V elementu *Body* je uveden již samotný dotaz na metodu `<geos:GetPlace>` s parametry `<geos:Country>` a `<geos:Place>` (dotaz je na místo ležící v *US* s PSČ *10065* a pro ověření se používá API klíč *1a0d749e646e46b1a459b10afb559112*, který byl automaticky přidělen při registraci na stránkách provozovatele služby).

Odpověď na tento dotaz je uvedena ve výpisu 2. Z odpovědi lze vyčíst, že tomuto PSČ odpovídá stát *New York*. Obsahem odpovědi jsou i další informace jako region nebo souřadnice místa.

Výpis 2: SOAP odpověď služby Geo Services

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Body>
      <Location xmlns="http://geoservice/"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <Type>PostalCode</Type>
         <Name>10065</Name>
         <City>New York</City>
         <Region>NY</Region>
         <PostalCode>10065</PostalCode>
         <Lat>41</Lat>
         <Lon>-74</Lon>
      </Location>
   </s:Body>
</s:Envelope>
```

Následující tabulka 1 shrnuje některé výhody a nevýhody technologie SOAP.

Tabulka 1: *Výhody a nevýhody SOAP (Zdroj: autor)*

Výhody	**Nevýhody**
Standardizovaný	Datově náročný
Nezávislý na transportním protokolu	
WS* standardy	

4.3 REST

REST (Representational State Transfer) je alternativa k protokolu SOAP. Jedná se o architekturu postavenou na principech popisující lokalizaci a přístup ke zdrojům. Autorem tohoto přístupu je Roy Fielding, který tento přístup prvně definoval v roce 2000 (Barry a Dick, 2013).

REST je na rozdíl od SOAP orientován datově. Všechny zdroje mají vlastní identifikátor URI a REST určuje, jak se k těmto datům přistupuje. Existují čtyři základní metody pro manipulaci se zdrojem, které odpovídají metodám HTTP protokolu (Malý, 2009):

- **GET** – získá zdroj na základě URI (odpovídá operaci READ).
- **POST** – vytvoří nový zdroj (odpovídá operaci CREATE).
- **PUT** – aktualizuje zdroj nebo v případě, že neexistuje, zdroj vytvoří (odpovídá operaci UPDATE).
- **DELETE** – vymaže identifikovaný zdroj (odpovídá operaci DELETE).

REST je jednodušší a datově méně náročný než jeho protějšek SOAP. Na rozdíl od SOAP není REST závislý na XML a odpovědi tak mohou být zaslány v mnohem lehčí formě jako je například JSON (Kankanamge, 2012).

Ukázka REST struktury dotazu a odpovědi ve formátu XML je uvedena v následujících výpisech 3 a 4.

Výpis 3: REST dotaz na službu Geo Services pro získání odpovědi v XML

```
http://api.geosvc.com/rest/US/10065?apikey=1a0d749e646e46b1a459b10afb559112
```

Výpis 4: REST odpověď služby Geo Services v XML

```
<Location xmlns="http://geoservice/" xmlns:i="http://www.w3.org/2001/XMLSchema-
instance">
   <Type>PostalCode</Type>
```

```
    <Name>10065</Name>
    <City>New York</City>
    <Region>NY</Region>
    <PostalCode>10065</PostalCode>
    <Lat>41</Lat>
    <Lon>-74</Lon>
</Location>
```

Výpis 3 obsahuje ten samý dotaz na službu Geo Services jako dotaz SOAP ve výpisu 1. Z výpisu lze vidět, že dotaz je zapsán mnohem úsporněji, ale zároveň ztrácí na první pohled svou čitelnost pro člověka, protože jsou zadány již pouze hodnoty parametrů a chybí názvy parametrů.

Formát dotazu pro výpis 3 je pro názornost uveden v následujícím výpisu 5. V dotazu se opět zadávají 3 parametry:

- `{COUNTRYCODE}` pro kód lokace (*US*),
- `{POSTALCODE}` pro PSČ (*10065*) a
- `{APIKEY}` pro ověření (*1a0d749e646e46b1a459b10afb559112*).

Výpis 5: REST dotaz s popsanými parametry

```
http://api.geosvc.com/rest/{COUNTRYCODE}/{POSTALCODE}?apikey={APIKEY}
```

Pro názorné porovnání struktury odpovědi v XML a JSON je dále uvedena ukázka REST dotazu a odpovědi ve formátu JSON (výpisy 6 a 7).

Výpis 6: REST dotaz na službu Geo Services pro získání odpovědi v JSON

```
http://api.geosvc.com/rest/US/10065?apikey=1a0d749e646e46b1a459b10afb559112&form
at=json
```

Výpis 7: REST odpověď služby Geo Services v JSON (formátovaná)

```
{
   "Type": "PostalCode",
   "Name": "10065",
   "City": "New York",
   "Region": "NY",
   "PostalCode": "10065",
   "Lat": 41,
   "Lon": -74
}
```

Jak je patrné z výpisu 7, odpověď REST ve formátu JSON je oproti odpovědi ve formátu XML opět o něco menší.

Výsledkem je, že REST odpověď je ve formátu XML i JSON v zápisu stručnější a méně náročnější na přenos než SOAP. Zároveň pro odeslání dotazu není zapotřebí specializovaný nástroj jako například SoapUI, jako je tomu v případě protokolu SOAP. Vzhledem k tomu, že se jedná o URI, je možné dotaz zadat přímo do internetového prohlížeče jako webovou adresu. Po odeslání dotazu je v okně internetového prohlížeče zobrazena odpověď.

Následující tabulka 2 shrnuje některé výhody a nevýhody technologie REST.

Tabulka 2: *Výhody a nevýhody REST (Zdroj: autor)*

Výhody	**Nevýhody**
Datově méně náročný než SOAP	Závislost na HTTP
Pro komunikaci nevyžaduje speciální nástroj	
Jednodušší než SOAP	

4.4 Popis rozhraní webové služby

Pro popis rozhraní webových služeb slouží dokumenty WSDL nebo například WADL. Dokument pro popis rozhraní definuje rozhraní služby tak, aby ten, kdo chce na službu odeslat požadavek, byl schopen sestavit dotaz (Breshne, Fuhrer a Pasquier, 2009).

S dokumentem popisujícím rozhraní se můžeme setkat obvykle u webových služeb SOAP, kde je na rozdíl od webových služeb REST dokument WSDL povinný.

4.4.1 WSDL

WSDL (Web Services Description Language) je XML dokument, který slouží k popisu rozhraní webové služby. V rámci této specifikace by mělo být popsáno rozhraní a další informace o dané webové službě. Mezi tyto informace patří například protokol nebo adresa, kde je webová služba dosažitelná (Kankanamge, 2012; Kuba, 2006).

V dnešní době existují 2 verze WSDL, které se stále používají (Elkstein, 2010):

- **WSDL 1.1** – verze používaná především pro popis SOAP webových služeb. Pro popis REST webových služeb chybí některé HTTP metody jako PUT a DELETE.
- **WSDL 2.0 –** především zlepšená podpora pro popis REST webových služeb, díky přidání podpory všech HTTP metod.

Alternativou k WSDL (především pro webové služby REST) je WADL.

4.4.2 WADL

WADL (Web Application Description Language) stejně jako WSDL slouží k popisu webových služeb. Jedná se o XML popis webových služeb obvykle komunikujících prostřednictvím protokolu HTTP. Z toho důvodu se s WADL lze setkat především u webových služeb REST. Nicméně WADL na rozdíl od WSDL není příliš rozšířen (Kankanamge, 2012; Breshne, Fuhrer a Pasquier, 2009).

5 Testování webových služeb

Jak již bylo zmíněno v úvodu, testování webových služeb má svá specifika. Obsahem této kapitoly je jejich popsání a uvedení možností a nástrojů pro testování webových služeb.

5.1 Specifika testování webových služeb

Významným specifikem webových služeb je, že webová služba obvykle nemá uživatelské rozhraní (UI). Je to dáno především tím, že webová služba je obvykle volána jako komponenta v systému (ať již například aplikací nebo jinou webovou službou). Odpověď webové služby je systémem dále zpracovávána a až následně prezentována uživateli (Huw, 2013).

Vzhledem k tomu, že webové služby obvykle nemají vystavené uživatelské rozhraní, je potřeba mít pro testování webových služeb vhodný nástroj, který by s nimi usnadnil komunikaci a ideálně umožnil jednoduchou tvorbu testů (pro funkční, integrační a bezpečností testy webových služeb je kritické mít správný nástroj) (Huw, 2013).

Dalším specifikem je, že testování webových služeb může začít ještě před tím, než je vyvinut front-end systému (GUI aplikace). Testovat webové služby je tak možné již v raných fázích vývojového cyklu (Huw, 2013). Včasným testováním webových služeb lze významně snížit počet chyb nalezených v systémových testech.

U webových služeb se lze také setkat s problémem týkající se jejich dokumentace. K webové službě může být dostupný pouze popis rozhraní webové služby například pomocí již zmiňovaného WSDL. To je však nedostatečné pro specifikaci funkčních a nefunkčních požadavků pro účely testování.

5.1.1 Možnosti a nástroje pro testování webových služeb

Jak již bylo uvedeno, pro testování webových služeb je důležité mít správný nástroj. Nástrojů pro testování webových služeb však na trhu není mnoho. Ve své práci identifikovala Repáňová (2014) celkem čtyři nástroje, z nichž jako nejlepší byl zvolen SoapUI:

- **SoapUI -** jeden z nejznámějších nástrojů pro komplexní testování webových služeb. Je nabízen jako open source, tak i v placené verzi, která nabízí oproti bezplatné více funkcionalit. Popisu nástroje SoapUI je věnována kapitola 5.2.
- **SOAPSonar –** nástroj firmy Crosscheck Networks určený především pro testování a diagnostiku SOAP, XML a REST služeb. Nabízen je ve verzi zdarma i v komerční verzi, která obsahuje více funkčností.

- **Examine –** komerční nástroj pro testování webových služeb poskytovaný firmou Stratumsoft Technologies. Poslední verze tohoto nástroje je datována k roku 2013.
- **Strom –** open source nástroj pro testování webových služeb. Poslední verze nástroje je Storm r1.1-Adarna z roku 2009.

Autoři Azzam, Al-Kabi a Alsmadi (2012) ve své studii porovnávali tři nástroje pro testování webových služeb, ze kterých opět vzešel vítězně nástroj SoapUI:

- **SoapUI**,
- **PushToTest** – open source nástroj pro testování nejen webových služeb.
- **WebInject** – nástroj pro testování webových aplikací a služeb. Poslední verze nástroje je 1.41 z roku 2006.

K otestování webových služeb lze využít i specializovaných nástrojů jako je Apache JMeter, který slouží k provádění zátěžových testů. Apache JMeter podporuje jak SOAP, tak REST webové služby.

Otestovat webové služby lze také pomocí různých frameworků. Jedním takovým je framework Codeception. Jedná se o testovací framework, který mimo jiné umí testovat SOAP i REST webové služby (Codeception, 2015).

5.2 SoapUI

SoapUI je nástrojem společnosti SmartBear Software pro tvorbu komplexních funkční a nefunkčních testů. SoapUI podporuje řadu technologií a standardů: SOAP a REST webové služby, JMS nebo RIA. Není to tedy nástroj pouze pro testování webových služeb, jak by se na první pohled mohlo zdát, nicméně oblast testování SOAP a REST webových služeb je pro něj stěžejní.

První verze nástroje SoapUI (1.0) vznikla v roce 2005. Autorem byl Ole Lensmer, který SoapUI tvořil ve svém volném čase. Do roku 2011 byl nástroj publikován pod firmou Eviware, která nabízela jak verzi open source, tak komerční „Pro“. V roce 2011 byla Eviware odkoupena firmou SmartBear Software zabývající se nástroji na podporu testování (Kankanamge, 2012). V roce 2014 SmartBear integroval nástroj SoapUI Pro do svého nového nástroje „Ready! API“, zatímco open source verze SoapUI označovaná jako „OS“ nadále pokračuje jako standalone verze.

Mezi hlavní přednosti nástroje patří:

- podpora SOAP a REST webových služeb,
- možnost tvorby funkčních a nefunkčních testů webových služeb (regresní, bezpečností a zátěžové testy) s širokou paletou aplikovatelných kontrol,
- tvorba skriptů (Groovy, JavaScript),

- možnost vytvářet mocky webových služeb,
- automatizace provádění testů.

V následujících kapitolách jsou představeny jednotlivé verze nástroje SoapUI a porovnány jejich nabízené funkcionality.

5.2.1 Verze SoapUI

Nástroj SoapUI, jak již bylo výše zmíněno je poskytován ve verzi zdarma jako SoapUI OS a v placené verzi jako SoapUI NG Pro.

SoapUI OS

Verze OS (Open Source) je poskytována zdarma pod licencí EUPL, European Union Public Licence. Oproti placené verzi však obsahuje méně funkcionalit (srovnání jednotlivých verzí se věnuje následující kapitola).

Poslední dostupná verze SoapUI OS v době psaní této DP je 5.2.1. Je možné ji stáhnout z webových stránek SoapUI:

- http://www.soapui.org/downloads/soapui/open-source.html.

SoapUI NG Pro

Jde o placenou verzi nástroje SoapUI, dříve poskytovanou pod názvem SoapUI Pro. Od konce roku 2014 došlo k přejmenování na SoapUI NG Pro (NG = New Generation) a k integrování do nového nástroje „Ready! API“.

Nástroj je možné zdarma vyzkoušet ve 14denní lhůtě (tzv. Trial verze). SoapUI NG Pro je možné stáhnout z webových stránek společnosti SmartBear:

- http://smartbear.com/lp/soapui/download-soapui-ng-pro/.

Ready! API

Nový nástroj firmy Smart Bear. První verze Ready! API byla publikována 15.10.2014. Ready! API v sobě integruje několik nástrojů pro komplexní testování API. Jedná se o:

- SoapUI NG Pro,
- LoadUI NG Pro,
- Secure Pro,
- ServiceV Pro.

Ready! API je možné vyzkoušet ve 14denní lhůtě. Poslední publikovaná verze v době psaní této DP je 1.4.1. Nástroj je možné stáhnout ze stránek SmartBear:

- http://smartbear.com/product/ready-api/free-trial/.

5.2.2 Rozdíly mezi SoapUI OS a SoapUI NG Pro

Placená verze SoapUI obsahuje oproti verzi zdarma několik funkcionalit navíc. Jedná se o funkce, které usnadňují a zpříjemňují práci s nástrojem.

Tabulka 3 uvádí rozdíly těchto verzí. Detailní srovnání poskytuje výrobce a je k dispozici na stránkách:

- http://www.soapui.org/about-soapui-pro/product-comparison/soapui-and-soapui-pro.html.

Je však vhodné poznamenat, že řadu funkcí, které nabízí SoapUI NG Pro, je možné provést i ve verzi zdarma, je jen potřeba více využít skriptovací jazyk Groovy, případně JavaScript, který je k dispozici i ve verzi SoapUI OS.

Tabulka 3: *Srovnání verze SoapUI OS a verze Pro (Zdroj: autor)*

Funkcionalita	**SoapUI OS**	**SoapUI NG Pro**
Podpora	Ano (ale pouze komunita)	Ano (oficiální podpora od vývojářů)
Environments (funkcionalita přepínání definovaných prostředí)	Ne	Ano
Tvorba a správa požadavků	Ne	Ano
SOAP	Ano	Ano
REST	Ano	Ano
Tvorba testů (test suits, test cases, assertions, proměnné)	Ano (omezená o některé funkce)	Ano
Tvorba mock-up (SOAP, REST)	Ano	Ano
Groovy, JavaScript	Ano	Ano
WS-Security, WS-I Integration	Ano	Ano
SSL podpora	Ano	Ano
Zátěžové testy (Load Testing)	Ano (omezené)	Ano

Bezpečnostní testy (Security Testing)	Ano	Ano (navíc automatické generování)
HTTP, JDBC, WSDL, JMS	Ano	Ano
Automatizace testů	Ano	Ano
Reporty	Ano (omezené)	Ano

6 Metodiky vývoje softwaru

Obsahem této kapitoly je vysvětlení pojmu metodika a představení nástroje pro správu metodik Eclipse Process Framework Composer (EPFC), ve kterém je metodika Testování webových služeb nástrojem SoapUI vytvořena.

6.1 Metodika

Pojem metodika dle Buchalcevové (2005) představuje v obecném smyslu „*souhrn metod a postupů pro realizaci určitého úkolu*“. Vzhledem k obecné povaze této definice se lze setkat i s pojmem „Metodika budování IS/ICT“, který je více vztažený k vývoji a údržbě informačních systémů. Metodiku budování IS/ICT definuje Buchalcevová (2005) jako „*principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení*“.

V dnešní době se můžeme setkat s dvěma hlavními kategoriemi metodik. Jedná se o:

- rigorózní metodiky a
- agilní metodiky.

Rigorózní metodiky jsou označovány jako metodiky těžké a lze je chápat jako předchůdce agilních metodik. Jsou charakteristické tím, že se snaží popsat všechny požadavky IS/ICT na počátku. V důsledku toho pak jen obtížně reagují na změny v požadavcích (Buchalcevová, 2005).

Oproti tomu agilní metodiky žijí z přesvědčení, že požadavky není možné na počátku úplně specifikovat. Pro agilní metodiky je důležité vytvořit rychlé řešení, které je následně možné upravovat na základě měnících se požadavků zákazníka, kterému je řešení průběžně prezentováno. Agilní metodiky jsou označovány jako metodiky lehké, protože na rozdíl od rigorózních nejsou tak obsáhlé a nevzniká v nich takové množství vedlejších produktů (Buchalcevová, 2005).

6.2 Nástroje pro správu metodik

Pro správu metodik neexistuje mnoho nástrojů. Mezi nejvýznamnější zástupce těchto nástrojů patří Eclipse Process Framework Composer (EPFC), který je vytvořen a udržován v rámci open source projektu EPF (Eclipse Process Framework). Projekt je dostupný na webových stránkách http://projects.eclipse.org/projects/technology.epf a mezi hlavní přispěvatele tohoto projektu patří firma IBM.

EPFC je poskytován zdarma, ale IBM nabízí placenou verzi s názvem Rational Method Composer (RMC), obsahující řadu pokročilých funkcí, které EPFC nenabízí.

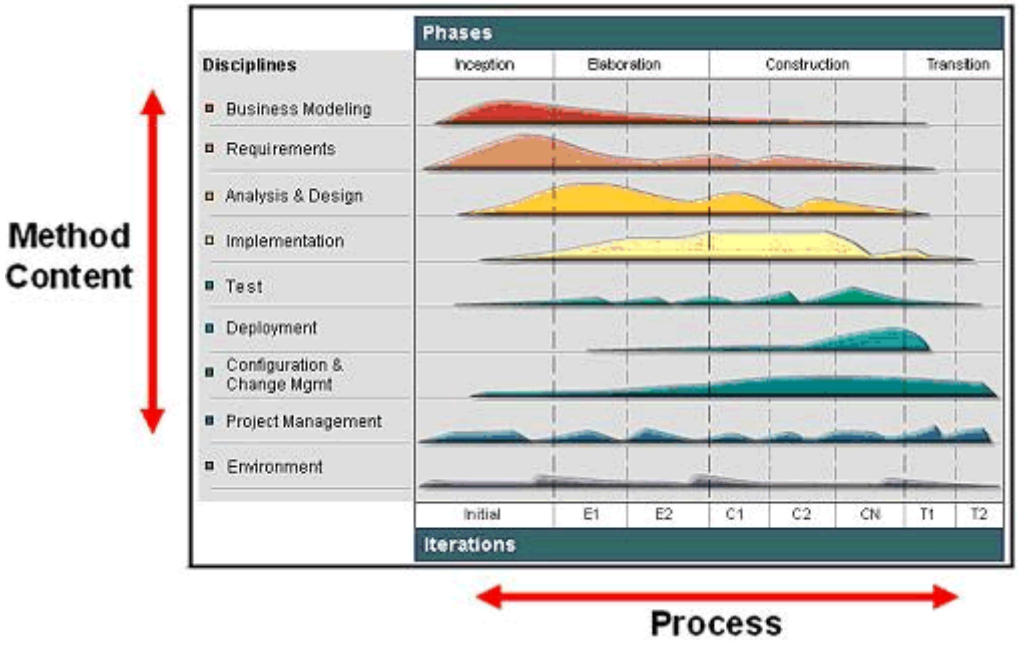
6.2.1 Eclipse Process Framework Composer

EPFC umožňuje implementovat, nasadit a udržovat procesy organizací nebo individuálních projektů (Eclipse, 2014).

Základním principem EPFC je oddělení metodického obsahu od jeho aplikace v procesech.

- **Metodický obsah –** popisuje, co má být vyrobeno, potřebné schopnosti a způsob, jak dosáhnout definovaných cílů. Popisy metodického obsahu jsou nezávislé na vývojovém životním cyklu.
- **Procesy –** popisují vývojový životní cyklus. Procesy přebírají prvky metodického obsahu a přiřazují je do sekvencí, které jsou přizpůsobeny specifickým typům projektů.

Obrázek 5 názorně ukazuje rozdělení metodického obsahu a procesů. Na ose *x* je životní cyklus a na ose *y* jednotlivé disciplíny.



Obrázek 5: *Rozdělení metodického obsahu a procesů v EPFC (Zdroj: Eclipse, 2014)*

Přehled klíčových elementů nástroje EPFC a jejich vztah k metodickému obsahu a procesu je uveden na následujícím obrázku 6. Metodický obsah je primárně vyjádřen elementy:

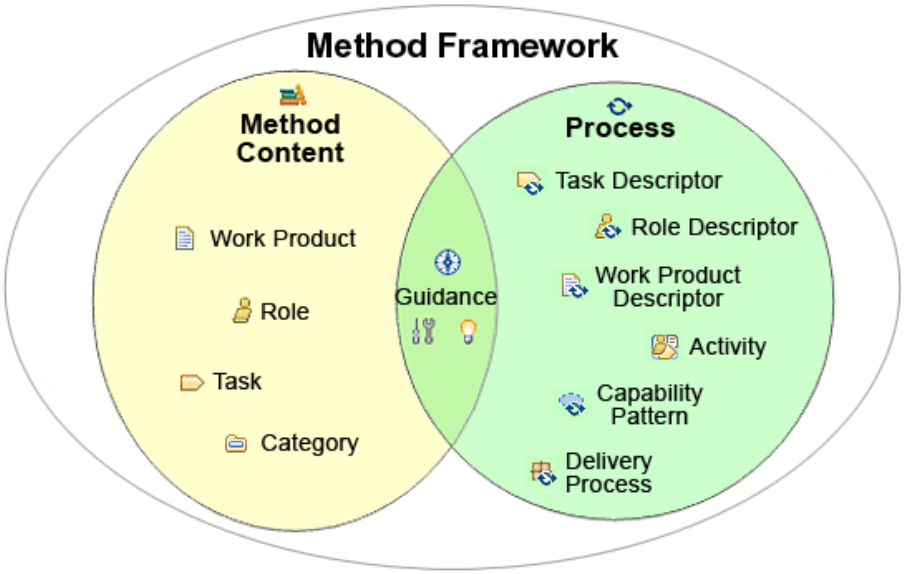
- Work Product (pracovní produkt)
- Role
- Task (úloha)
- Category (kategorie)

Kategorie dále definuje 5 základních typů: *Disciplines* (disciplíny), *Domains* (domény), *Work Product Kind* (druhy pracovního produktu), *Role sets* (skupiny rolí) a *Tools* (nástroje).

Elementy popisující procesy jsou pak následující:

- Task Descriptor (popis úlohy)
- Role Descriptor (popis role)
- Work Product Descriptor (popis pracovního produktu)
- Activity (aktivita)
- Capability Pattern (vzor schopnosti)
- Delivery Process (dodací proces)

Průnikem metodického obsahu a procesu je „Guidance“ jež obsahuje příklady nebo návody.



***Obrázek 6**: Elementy metodického obsahu a procesu (Zdroj: Eclipse, 2014)*

Důkladnější popis nástroje EPFC lze najít v dokumentaci programu nebo v bakalářské práci Marka Pospíšila (2009), který zpracoval příručku k tomuto nástroji.

7 Metodika Testování webových služeb nástrojem SoapUI

Obsahem této kapitoly je popis navržené metodiky Testování webových služeb nástrojem SoapUI vypracované v nástroji EPFC.

Pokládám však za důležité nejprve přiblížit metodiku MMSP, ze které vypracovávaná metodika vychází. Metodika MMSP zde ale není popsána celá, pouze část, která je pro vypracovávanou metodiku podstatná. Úplný popis metodiky MMSP a všech jejích prvků je možné nalézt v diplomové práci Petry Rejnkové (2010) nebo v publikované metodice na stránkách http://mmsp.czweb.org/.

7.1 Metodika MMSP

Metodika MMSP (Metodika pro Malé Softwarové Projekty) je metodikou Petry Rejnkové (2010), která vznikla v rámci diplomové práce „Lokalizace a přizpůsobení metodiky OpenUP“. MMSP je tedy do značné míry inspirována open source metodikou OpenUP, která je odnoží metodiky Unified Process. Jedná se o agilní metodiku se zaměřením na iteraci, inkrementální vývoj, kolektivní práci a komunikaci v týmu. Vzhledem k tomu, že metodika OpenUP není předmětem této práce, bližší informace o této metodice lze získat na adrese: http://epf.eclipse.org/wikis/openup/, kde je metodika publikována, nebo v diplomové práci Petry Rejnkové (2010).

Stejně jako metodika OpenUP, je i metodika MMSP publikována v nástroji EPFC a vychází tak z principů nástroje (principy jsou uvedeny v kapitole o nástroji EPFC - 6.2.1). Aktuální verze metodiky MMSP je dostupná na internetových stránkách http://kitscm.vse.cz/ a jejímu popisu se věnuje i již zmiňovaná publikace „Příklady modelů analýzy a návrhu aplikace v UML“ od Buchalcevové a Stanovské (2013).

Metodika MMSP definuje 4 hlavní elementy: *Role*, *Disciplíny*, *Pracovní produkty* a *Životní cyklus*. V následujících kapitolách jsou tyto čtyři elementy blíže popsány s tím, že je důraz kladen na popis disciplíny testování, protože z těchto prvků metodika vypracovávaná v této diplomové práci vychází.

7.1.1 Role

Metodika MMSP definuje následující role: *Architekt*, *Projektový manažer*, *Analytik*, *Tester*, *Nedefinovaná role*, *Vývojář* a *Zainteresovaná strana*. U každé role se nachází její podrobný popis obsahující její účel, odpovědnosti za úlohy a pracovní produkty, schopnosti a dovednosti. Pro disciplínu testování je nejpodstatnější role *Tester*.

Tester

Role testera v této metodice vykonává veškeré činnosti spojené s disciplínou testování (viz obrázek 7**)**. Jedná se o úlohy:

- Plánování testů
- Provedení testů
- Příprava testů

Role testera je odpovědná za následující pracovní produkty:

- Plán testů
- Seznam testovacích nápadů
- Testovací data
- Testovací případ
- Testovací sada
- Záznam výsledků testů

Obrázek 7: *Úlohy a pracovní produkty role Tester (Zdroj: Rejnková, 2010)*

7.1.2 Disciplíny

Pojem „Disciplína“ v metodice MMSP označuje seskupení více úloh. MMSP rozlišuje: *Řízení projektu*, *Požadavky*, *Architektura*, *Vývoj*, *Řízení změn a konfigurace* a *Testování*.

Testování

Jedná se o disciplínu, která popisuje, jak efektivně získávat zpětnou vazbu o vytvářeném IS/ICT (Rejnková, 2010).

Obsahem disciplíny „Testování“ jsou následující úlohy (Rejnková, 2010):

- **Plánování testů** – určení základního přístupu k testování (testovací techniky, harmonogram, zodpovědnosti).
- **Příprava testů –** jedná se především o přípravu testů a testovacích podkladů.

- **Provedení testů –** vlastní testování vyvíjeného systému na základě naplánovaných testů (zaznamenání chyb a vyhodnocení výsledků).

Tyto úlohy by dle metodiky MMSP měly být prováděny téměř ve všech iteracích životního cyklu metodiky.

7.1.3 Pracovní produkty

Pracovní produkty disciplíny testování jsou následující (Rejnková, 2010):

- **Plán testů** – obsahuje důležité informace o testování pro danou verzi vyvíjeného IS/ICT.
- **Seznam testovacích případů** – zachycuje možné problémové oblasti vyvíjeného IS/ICT, které by měly být otestovány.
- **Testovací data –** množina dat, která se používá při testování.
- **Testovací případ –** popisuje kroky a očekávané výsledky pro otestování specifické funkcionality aplikace.
- **Testovací sada –** uspořádání testovacích případů do logické posloupnosti za účelem otestování určité části aplikace.
- **Záznam výsledků testů –** informace o provedených testech a jejich výsledcích.

7.1.4 Životní cyklus

Životní cyklus se stejně jako v případě metodiky OpenUP dělí na 4 fáze.

1. **Zahájení** - fáze zahájení odpovídá fázi „Inception“ metodiky OpenUP. V této fázi životního cyklu dochází k zahájení projektu. Předmětem je porozumět požadavkům na vyvíjený systém, identifikovat klíčové funkce a rizika, které by mohla projekt ohrozit (Rejnková, 2010). V této fázi se předpokládá již schválený projektový záměr, jenž vymezuje základní charakteristiky projektu.

   Fáze zahájení probíhá v iteraci a je ukončena dosažením milníku „Předmět a rozsah milníku“.

2. **Rozpracování** - hlavním cílem fáze rozpracování je porozumět požadavkům, vymezit architekturu řešení, přijmout opatření pro zmírnění dopadů a prevenci rizik a upřesnit plán projektu (Rejnková, 2010).

   Fáze rozpracování opět probíhá v iteraci a je zakončena milníkem „Definice architektury“.

3. **Konstrukce** - cílem této fáze je vytvoření funkční verze systému. Tato verze by měla být v takovém stavu, že je ji možné nasadit pro účely testování a akceptace (Rejnková, 2010).

Fáze konstrukce je tvořena jednou nebo větším počtem iterací. Fáze je ukončena dosažením milníku „Provozní způsobilost“.

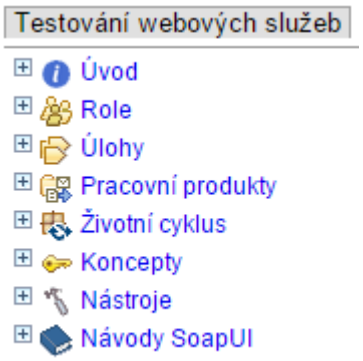
4. **Zavedení** - fáze zavedení je poslední fází životního cyklu metodiky MMSP. V této fázi dochází k uvedení systému do provozu (Rejnková, 2010).

   Fáze zavedení probíhá v iteraci a je zakončena milníkem „Nasazení“.

7.2 Charakteristika metodiky Testování webových služeb nástrojem SoapUI

Metodika Testování webových služeb nástrojem SoapUI je zaměřena na oblast testování a vychází z výše zmíněné metodiky MMSP, kterou upravuje a rozšiřuje o prvky podstatné pro testování webových služeb. Metodika vypracovaná v této práci začleňuje role, úlohy a pracovní produkty související s testováním webových služeb do disciplíny testování, která je definována metodikou MMSP. Metodika se detailněji nevěnuje dalším disciplínám, které zmiňuje metodika MMSP (řízení projektu, požadavky, architektura, vývoj a řízení změn a konfigurace). V rámci vypracovávané metodiky jsou vytvořeny nové elementy (koncepty, nástroje a příručka SoapUI), které jsou důležité pro testování webových služeb nástrojem SoapUI.

Metodika je vytvořena jako samostatná metodika v nástroji EPFC a následující kapitoly, jsou jejím popisem. Metodika se skládá z 8 prvků. Jednotlivé prvky, tak jak jsou vytvořeny v EPFC, jsou znázorněny na obrázku 8. V následujících kapitolách jsou jednotlivé prvky detailně popsány.



Obrázek 8: *Prvky metodiky testování webových služeb nástrojem SoapUI (Zdroj: autor)*

Prvky metodiky Testování webových služeb nástrojem SoapUI:

- **Úvod** – obsahuje základní informace o metodice.
- **Role –** všechny role metodiky, jejich popis a vztahy k dalším prvkům metodiky.
- **Úlohy –** jednotlivé úlohy testování webových služeb, které spadají pod disciplínu testování, jejich popis a návaznosti na další prvky metodiky.

- **Pracovní produkty –** artefakty, které jsou v rámci metodiky vyvářeny, jejich popis a návaznosti na další prvky metodiky.
- **Životní cyklus** – životní cyklus, jak je definován metodikou MMSP. Do životního cyklu jsou zařazeny jednotlivé úlohy, role a pracovní produkty metodiky Testování webových služeb nástrojem SoapUI.
- **Koncepty –** základní pojmy a informace důležité pro testováním webových služeb.
- **Nástroje –** popis nástrojů, které jsou pro testování webových služeb touto metodikou podstatné.
- **Příručka SoapUI** – návody a ukázky použití nástroje SoapUI. Návody pokrývají oblast od instalace samotného nástroje, tvorby testů až po možnosti automatizace spouštění testů.

V následujících kapitolách je uveden strukturovaný popis jednotlivých prvků metodiky Testování webových služeb nástrojem SoapUI, který vychází z návrhu Rejnkové (2010).

7.3 Role

Předmětem této kapitoly je představení rolí, které vstupují do procesu testování webových služeb. U každé role je na konci uveden seznam úloh a pracovních produktů, které s danou rolí souvisej.

Ve vypracovávané metodice dochází oproti MMPS k vyčlenění více rolí z jediné dostupné role *Tester* (v rámci disciplíny testování). Důvodem je zřetelnější oddělení odpovědností jednotlivých členů týmu za jednotlivé části (úlohy a pracovní produkty) v procesu testování. Specifickou rolí pro tuto metodiku je role *Tester služeb*, která má primární odpovědnost za testování webových služeb. Mezi další nové role patří *Analytik testování* a *Manažer testování*, se kterými se lze v procesu testování obvykle setkat. Role *Tester* zůstává přítomna s tím, že některé úlohy a pracovní produkty byly přesunuty do jiných rolí. Z role *Zainteresovaná strana* dochází k vyčlenění role *Zákazník*, která má odpovědnost především za požadavky a provedení akceptačního testování. Role vývojáře zůstává s tím, že dochází k rozšíření o specifické prvky související s webovými službami.

Přehled všech rolí používaných v rámci metodiky Testování webových služeb nástrojem SoapUI:

- Tester služeb
- Analytik testování
- Manažer testování
- Tester
- Vývojář

- Zákazník

7.3.1 Tester služeb

Oproti roli testera jsou na testera webových služeb kladeny vyšší nároky na znalosti a schopnosti (především související s webovými službami). Je třeba, aby takový tester měl nejen povědomí o oblasti webových služeb (která je z podstaty webových služeb velmi rozsáhlá), ale aby měl i širší povědomí o dalších technologiích v IT jako je programování, databáze, architektura či sítě. Zároveň by takovýto tester měl být dobře seznámen i s architekturou v podniku, aby byl schopen komplexně otestovat webovou službu, která je předmětem testování.

Hlavním úkolem testera webových služeb je tvorba testovacích skriptů v SoapUI a jejich provádění. Tester webových služeb je dále zodpovědný za vytvoření a údržbu dokumentu *Standardy tvorby testů v SoapUI*, který je vytvářen již během fáze plánování testů.

Tvorba, správa, spouštění testů a reportování nalezených chyb webových služeb je časově velmi náročná. Je proto vhodné, aby tuto roli v týmu zastávala alespoň jedna osoba. V případě, že to není uskutečnitelné (obvykle v menších týmech), je možné roli testera webových služeb sloučit s rolí testera.

Obrázek 9: *Úlohy a pracovní produkty role Tester služeb (Zdroj: autor)*

Úlohy

- Implementace testů webových služeb
- Provádění testů webových služeb
- Údržba testů

Pracovní produkty

- Standardy tvorby testů v SoapUI
- Testovací data
- Testovací skript

- Záznam výsledků testů

Může vykonávat

- Analýza a návrh testů
- Plánování testů
- Provedení akceptačních testů
- Provedení unit testů
- Tvorba unit testů

7.3.2 Analytik testování

Hlavním úkolem analytika testování je analýza požadavků a návrh testů tak, aby byla plně pokryta testovaná funkcionalita. Stejně jako v případě testera služeb je potřeba, aby i analytik testování měl dostatečné znalosti o oblasti webových služeb a architektuře v podniku. V případě, že požadované znalosti analytik testování nemá, tuto roli může zastávat tester služeb.

Analytik testování je zodpovědný za vytvoření testovacích sad a testovacích případů, na základě kterých bude schopen tester služeb vytvořit testovací skripty v SoapUI.

V případě menších týmů může tuto roli plnit tester, respektive tester služeb za oblast webových služeb.

Obrázek 10: *Úlohy a pracovní produkty role Analytik testování (Zdroj: autor)*

Úlohy

- Analýza a návrh testů
- Údržba testů

Pracovní produkty

- Seznam testovacích nápadů
- Testovací data

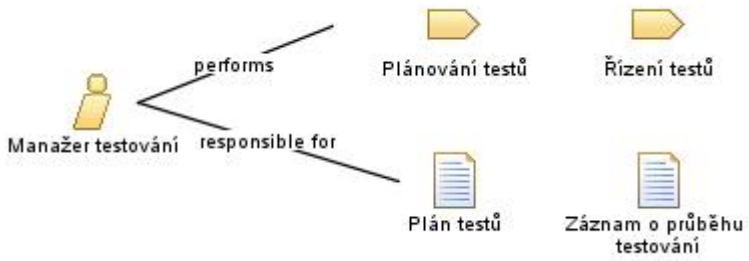
- Testovací případ
- Testovací sada

Může vykonávat

- Plánování testů

7.3.3 Manažer testování

Manažer testování má na starosti vedení celého testovacího týmů. Jeho hlavním úkolem je tvorba testovacího plánu a řízení testů. Manažer testování sleduje stanovené metriky a kontroluje plnění testovacího plánu. Výsledky komunikuje se zástupci vedení projektu a dalšími osobami podílejícími se na projektu.



Obrázek 11: *Úlohy a pracovní produkty role Manažer testování (Zdroj: autor)*

Úlohy

- Plánování testů
- Řízení testů

Pracovní produkty

- Plán testů
- Záznam o průběhu testování

Může vykonávat

Role nemá přiřazeny úlohy, které by mohla dále vykonávat.

7.3.4 Tester

Role testera v procesu testování webových služeb přímo nezastává žádnou úlohu. Tester v rámci procesu testování webových služeb však může vykonávat (vypomáhat) s úlohami, které mají na starosti primárně jiné role.

Úlohy

Role nezastává žádnou úlohu.

Pracovní produkty

Role není odpovědná za žádný pracovní produkt.

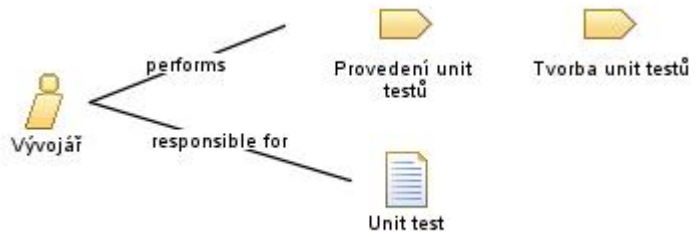
Může vykonávat

- Analýza a návrh testů
- Implementace testů webových služeb
- Provedení akceptačních testů
- Provedení testů webových služeb
- Údržba testů

7.3.5 Vývojář

Ačkoli tato role přímo nespadá do testovacího týmu, je potřeba ji zde zmínit. Hlavním úkolem této role v procesu testování je tvorba a spouštění unit testů webových služeb. Unit testy jsou první fází testování webových služeb a vývojář, který službu implementoval, by měl být zodpovědný i za vytvoření a otestování webové služby pomocí unit testů.

V případě, že by tuto roli nemohl vývojář plnit, může ji zastoupit tester webových služeb.



Obrázek 12: *Úlohy a pracovní produkty role Vývojář (Zdroj: autor)*

Úlohy

- Provedení unit testů
- Tvorba unit testů

Pracovní produkty

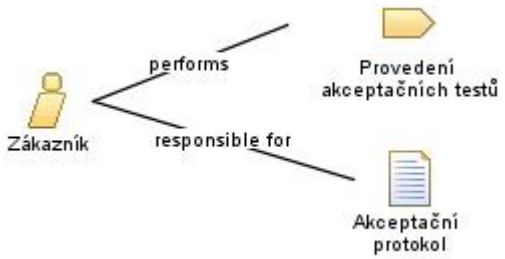
- Unit test

Může vykonávat

- Údržba testů

7.3.6 Zákazník

Role zákazníka stejně jako role vývojáře nepatří k testovacímu týmu, nicméně je v procesu testování klíčová. Hlavním úkolem této role je provedení akceptačních testů. Na základě provedených testů (zákazníkem nebo uživateli za stranu zákazníka) je následně vypracován akceptační protokol, který obsahuje zprávu o tom, zda vyvinutý produkt je zákazníkem akceptován nebo nikoli.



Obrázek 13: *Úlohy a pracovní produkty role Zákazník (Zdroj: autor)*

Úlohy

- Provedení akceptačních testů

Pracovní produkty

- Akceptační protokol

Může vykonávat

Role nemá přiřazeny úlohy, které by mohla dále vykonávat.

7.4 Úlohy

Předmětem této kapitoly je popis jednotlivých úloh, které jsou vykonávány při testování webových služeb. Na konci každé úlohy je uvedena tabulka mapující vztahy dané úlohy k rolím a pracovním produktům.

Úlohy zmíněné v této kapitole spadají do disciplíny *Testování*, kterou definuje metodika MMSP. V rámci ni jsou definovány tři úlohy této disciplíny: plánování testů, příprava testů a provedení testů.

Ve vypracovávané metodice je zachována úloha *plánování testů.* Úlohy *provedení testů* a *příprava testů* jsou dekomponovány na více úloh, které lépe zachycují vztahy jednotlivých rolí a produktů.

Ve vytvářené metodice dochází k formulaci úloh nových. Jedná se o úlohy: *analýza a návrh testů*, *implementace testů webových služeb*, *údržba testů*, *provedení testů webových služeb*, *řízení testů*, *provedení akceptačních testů*.

Z metodiky MMSP jsou dále do procesu testování webových služeb zařazeny úlohy *tvorba unit testů* a *provedení unit testů*, které provádí vývojář a jsou v metodice MMSP zařazeny pod disciplínu *Vývoj*.

Seznam všech úloh, které jsou v metodice Testování webových služeb nástrojem SoapUI definovány:

- Analýza a návrh testů
- Implementace testů webových služeb
- Údržba testů
- Provedení testů webových služeb
- Plánování testů
- Tvorba unit testů
- Provedení unit testů
- Řízení testů
- Provedení akceptačních testů

7.4.1 Analýza a návrh testů

Analýza požadavků na webové služby předchází návrhu testů na webové služby a jedná se o důležitou část procesu testování. Vychází se z dostupné dokumentace jako je specifikace od zákazníka (požadavky), případy užití, analýzy provedené v rámci analýzy a vývoje, popis rozhraní webové služby (například WSDL) a z dalších dostupných informací vytvořených

v rámci životního cyklu vývoje softwaru. V této části je potřeba identifikovat i jaké metody a techniky budou při návrhu testů využity.

Po provedení analýzy dochází k navržení testů (testovacích sad a testovacích případů). Cílovým stavem je situace, kdy testy budou pokrývat všechny definované požadavky kladené na webové služby.

V této úloze zároveň dochází k přípravě testovacích dat. Požadavky na testovací data by měly být již v plánu testování a v požadavcích na testovací prostředí.

Odpovědnost za provedení analýzy a návrhu má analytik testování.

Tabulka 4: *Vztahy úlohy Analýza a návrh testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	• Analytik testování	• Tester • Tester služeb
Vstupy	**Primární:**	**Doplňkové:**
	• Požadavky • Případy užití	---
Výstupy	• Seznam testovacích nápadů • Testovací sada • Testovací případ • Testovací data	

7.4.2 Implementace testů webových služeb

V této úloze dochází k vlastní tvorbě testovacích skriptů v SoapUI. Východiskem pro testovací skripty jsou testovací sady, respektive testovací případy vytvořené během úlohy *Analýza a návrh testů*.

Při tvorbě testovacích skriptů je třeba dbát na to, aby byla stanovena jasná pravidla tvorby testovacích skriptů (z pohledu jejich srozumitelnosti, budoucí udržovatelnosti a znovu použitelnosti). Tato pravidla specifikuje pracovní produkt *Standardy tvorby testů v SoapUI*, který je vytvářen během fáze plánování testů.

Nástroj SoapUI umožňuje vytvářen funkční i nefunkční testy:

- **Tvorba funkčních testů v SoapUI –** pravidlům a konceptům tvorby těchto testů se věnuje již zmíněný pracovní produkt *Standardy tvorby testů v SoapUI*. Možnostem tvorby těchto testů v nástroji SoapUI je věnována příručka v příloze A. V rámci této

příručky jsou například vytvořeny návody pro založení testovacího skriptu, tvorbu kontrol, práci s proměnnými a v neposlední řadě pro práci se skriptovacím jazykem Groovy.

- **Tvorba nefunkčních testů v SoapUI** - SoapUI umožňuje vytvářet především testy zátěžové a bezpečnostní. Pravidlům a konceptům tvorby těchto testů se opět věnuje pracovní produkt *Standardy tvorby testů v SoapUI*.
    - **Zátěžové testy –** výhodou tvorby zátěžových testů v SoapUI je možnost přepoužití již existujících částí funkčních testů. Zároveň SoapUI nabízí předdefinované sady strategií zátěžových testů, které jsou okamžitě k použití a umožňují snadnou úpravu. Tvorbě zátěžových testů se věnuje příručka k tvorbě testů v SoapUI, kapitola A.4.6 v příloze A.
    - **Bezpečnostní testy –** SoapUI nabízí předpřipravenou sadu bezpečnostních kontrol, které jsou blíže popsány v kapitole A.4.7 přílohy A. V případě verze „Pro“ umožňuje navíc SoapUI generovat bezpečnostní testy automaticky. Stejně jako v případě zátěžových testů je možné přepoužít již existující části funkčních testů.

Při tvorbě testovacích skriptů se používají testovací data připravená v rámci úlohy *Analýza a návrh testů*. Příprava testovacích dat ale není odpovědností pouze analytika testování, ale i testera služeb. Zde záleží na dohodě, kdo se jak a z jaké části bude na přípravě dat podílet.

Primární odpovědnost za implementaci testovacích skriptů v SoapUI má tester služeb. Výstupem této úlohy jsou testovací skripty v SoapUI.

Tabulka 5: *Vztahy úlohy Implementace testů webových služeb (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	● Tester služeb	● Tester
Vstupy	**Primární:**	**Doplňkové:**
	● Standardy tvorby testů v SoapUI ● Testovací případ ● Testovací sada	● Testovací data
Výstupy	● Testovací data ● Testovací skript	

7.4.3 Údržba testů

S úlohou *Implementace testů webových služeb* úzce souvisí *Údržba testů*. Údržba testovacích případů, respektive testovacích skriptů v SoapUI může být časově náročná. Je obvykle vyvolána zastaralou testovací dokumentací a potřebou ji aktualizovat. Může se jednat například o změnu v požadavcích, kterou je nutné promítnout do testů webových služeb.

Údržbu testovacích případů má na starosti analytik testování, údržbu testovacích skriptů pak tester služeb.

Tabulka 6: *Vztahy úlohy Údržba testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	• Tester služeb • Analytik testování	• Tester • Vývojář
Vstupy	**Primární:**	**Doplňkové:**
	• Testovací skript • Testovací případ • Testovací sada • Testovací data • Unit test	• Požadavky • Případy užití • Seznam testovacích nápadů
Výstupy	• Testovací skript • Testovací případ • Testovací sada • Testovací data • Unit test	

7.4.4 Provedení testů webových služeb

V rámci této úlohy probíhá spouštění testovacích skriptů vytvořených v nástroji SoapUI. K otestování webové služby je možné přistoupit nezávisle na jiných webových službách či dalších komponentách (není třeba čekat na zbývající části aplikace jako například na dovyvinutý front-end aplikace).

Vytvořené testy je možné spouštět manuálně nebo automatizovaně např. za pomocí nástroje Test Runner, jež je součástí SoapUI (více o nástroji Test Runner v kapitole A.5). Díky možnostem nástroje SoapUI spouštět testy vzdáleně pomocí příkazu je možné integrovat testy

vytvořené v tomto nástroji do některého z nástrojů pro kontinuální integraci. Pomocí těchto nástrojů je možné automaticky spouštět testy pokaždé, kdy je vytvořeno nové sestavení aplikace nebo webové služby.

V případě nalezení chyby je třeba ji zaznamenat do nástroje pro správu chyb (např. JIRA nebo Redmine, ale k zaznamenání je možné využít i Excel).

Po provedení testů a zaznamenání chyb je potřeba výsledky reportovat. Řada nástrojů pro řízení testů tuto funkcionalitu v sobě zahrnuje a tím tuto úlohu usnadňuje. Nástroj SoapUI není výjimkou a obsahuje sadu přednastavených reportů pro reportování průběhu testů.

Za provedení testů webových služeb primárně odpovídá tester služeb.

Tabulka 7: *Vztahy úlohy Provedení testů*

Role	**Primárně vykonává:**	**Může vykonávat:**
	• Tester služeb	• Tester
Vstupy	**Primární:**	**Doplňkové:**
	• Testovací skript	• Testovací sada • Testovací případ • Testovací data
Výstupy	• Záznam výsledků testů	

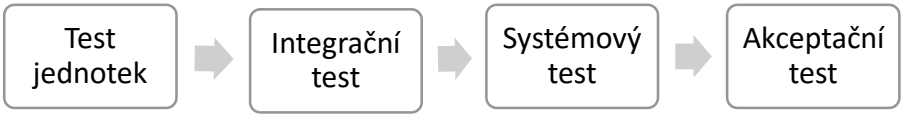
7.4.5 Plánování testů

Plánování testů je úlohou převzatou z MMSP. V průběhu plánování testů je specifikována část vyvíjeného systému, která by měla být testována, jsou určeny základní testovací techniky, vytvořen orientační harmonogram testování a rozdělena zodpovědnost za přípravu a provedení různých typů testů mezi jednotlivé členy týmu (Rejnková, 2010).

V rámci plánování testů dochází k tvorbě pracovního produktu *Plán testů*. Tvorbu tohoto dokumentu má na starosti manažer testování. Jedná se o stěžejní dokument v procesu testování, s jehož tvorbou je vhodné začít v dostatečném časovém předstihu.

Z pohledu testování webových služeb v rámci této úlohy také dochází k tvorbě pracovního produktu *Standardy tvorby testů v SoapUI*, který je součástí plánu testování a má ho na starosti především tester služeb.

Při testování webových služeb lze sledovat proces testování, který je definován V-modelem (viz kapitola 3.2). Jednotlivé fáze testování tak, jak jdou za sebou, znázorňuje obrázek 14.



Obrázek 14: *Fáze testování definované V-modelem (Zdroj: Roudenský a Havlíčková, 2013)*

Test jednotek

Během této fáze dochází k provádění unit testů, které by měl provádět vývojář. Předmětem je individuální testování operací a metod webové služby. Hlavním cíle je otestování toho, zda se jednotlivé komponenty webové služby chovají v izolovaném prostředí tak, jak je požadováno, a zda je možné webovou službu zkompilovat. Pro tvorbu unit testů je možné využít frameworky, které poskytuje daný programovací jazyk, v němž je webová služba vytvářena (v případě Javy to může být například framework JUnit). Vzhledem ke své povaze jsou tyto testy často automatizované.

Integrační test

V rámci této fáze se provádějí integrační testy, které by měl realizovat tester webových služeb. Integrační testy obvykle následují po jednotkovém testování – tj. po ověření, že jednotlivé komponenty služby se chovají, jak je požadováno. V rámci integračních testů se ověřuje, zda integrování s dalšími komponentami systému nenarušilo funkcionalitu webové služby. Testuje se tak rozhraní služby, napojení na externí webové služby, ESB a další komponenty, s kterými může webová služba komunikovat. V případě, že část, na kterou se webová služba integruje, není ještě dostupná nebo dovyvinutá, je možné využít tzv. mocky služeb (více viz kapitola B.7).

Systémový test

V rámci systémových testů probíhají testy funkční (testy funkcionality, regresní testy) a nefunkční (performance testy, bezpečnostní testy), které má na starosti tester webových služeb. Tyto testy obvykle následují po integračním testování, ale testování funkcionalit webové služby může proběhnout i po fázi jednotkového testování, ještě před integračním testování za účelem ověření kvalit webové služby. Předmětem systémových testů je ověření, zda služba naplňuje stanovené byznys požadavky (funkční i nefunkční), které byly identifikované v rámci analýzy. V případě návazností na jiné služby nebo komponenty, které stále nejsou dostupné nebo dovyvinuté, lze opět použít mocky služeb.

Akceptační test

V rámci této fáze probíhají především akceptační testy, které provádí zákazník (respektive uživatelé určení zákazníkem). Akceptační testování obvykle následuje po úspěšném ukončení systémových testů. Akceptační kritéria, respektive testy jsou připraveny předem (obvykle se připravují ve fázi analýzy požadavků) a jsou schváleny oběma stranami (jak dodavatelem, tak zadavatelem).

Tabulka 8: *Vztahy úlohy Plánování testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	● Manažer testování	● Tester služeb ● Analytik testování
Vstupy	**Primární:**	**Doplňkové:**
	● Plán projektu	● Záznam o průběhu testování
Výstupy	● Plán testů ● Standardy tvorby testů v SoapUI	

7.4.6 Tvorba unit testů

Tvorba unit testů je úlohou převzatou z MMSP, v této úloze dochází k tvorbě unit testů webových služeb.

Tvorba unit testů je první úroveň testování, kterou má na starosti vývojář. Ten v této úloze vytváří a spouští unit testy, které testují jednotlivé části webové služby. Jednotkové testy webových služeb se v principu neliší od běžných jednotkových testů. Dochází k testování izolovaných částí (jednotek) programu a k ověřování jejich správného chování. Při jednotkovém testování se často využívá tzv. mocků (maket). Tyto objekty simulují chování prvků jako například databáze nebo další webová služba.

Tabulka 9: *Vztahy úlohy Tvorba unit testů webové služby (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	● Vývojář	● Tester služeb
Vstupy	**Primární:**	**Doplňkové:**
	---	---
Výstupy	● Unit test	

7.4.7 Provedení unit testů

Provedení testů je taktéž úlohou převzatou z MMSP, v této úloze dochází k provádění unit testů webových služeb. Za provádění jednotkových testů webových služeb má primární odpovědnost vývojář.

Tabulka 10: *Vztahy úlohy Provedení unit testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	● Vývojář	● Tester služeb
Vstupy	**Primární:**	**Doplňkové:**
	● Unit test	---
Výstupy	● Záznam výsledků testů	

7.4.8 Řízení testů

Řízení testů je úloha, která se prolíná celým testovacím procesem. V rámci ní dochází ke kontrole naplňování testovacího plánu, případně k úpravám v testovacím plánu na základě zjištěných rozdílů.

Za řízení testů je odpovědný manažer testování.

Tabulka 11: *Vztahy úlohy Řízení testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	● Manažer testování	---
Vstupy	**Primární:**	**Doplňkové:**
	● Plán testů ● Plán projektu ● Záznam výsledků testů	● Akceptační protokol
Výstupy	● Záznam o průběhu testování	

7.4.9 Provedení akceptačních testů

V této úloze dochází k realizaci akceptačních testů. Jejich předmětem je ověření, zda dodaný produkt splňuje akceptační kritéria, která byla společně se zákazníkem definována na počátku projektu. Výsledkem této úlohy je akceptační protokol, obsahující zprávu o tom, zda je vyvinutý produkt zákazníkem akceptován nebo nikoli.

Za provedení akceptačních testů je zodpovědný zákazník (respektive uživatelé vystupující za stranu zákazníka).

Tabulka 12: *Vztahy úlohy Provedení akceptačních testů (Zdroj: autor)*

Role	**Primárně vykonává:**	**Může vykonávat:**
	• Zákazník	• Tester služeb • Tester
Vstupy	**Primární:**	**Doplňkové:**
	• Testovací případ • Testovací sada • Testovací skript	• Testovací data
Výstupy	• Akceptační protokol	

7.5 Pracovní produkty

V této kapitole jsou představeny jednotlivé pracovní produkty. Vztahy pracovních produktů k rolím a úlohám shrnuje tabulka na konci každého pracovního produktu.

Některé pracovní produkty byly převzaty z MMSP a případně upraveny pro potřeby metodiky testování webových služeb pomocí nástroje SoapUI. Jedná se následující: *plán testů*, *seznam testovacích nápadů*, *testovací data*, *testovací případ*, *testovací sada*, *záznam výsledků testů*, *unit test*, *plán projektu*, *požadavky* a *případy použití*.

Ve vytvářené metodice byly formulovány nové pracovní produkty: *testovací skript*, *záznam o průběhu testování* a *akceptační protokol*.

Seznam všech pracovních produktů používaných v metodice Testování webových služeb pomocí nástroje SoapUI:

- Testovací sada
- Testovací případ

- Testovací skript
- Testovací data
- Záznam výsledků testů
- Plán testů
- Záznam o průběhu testování
- Unit test
- Požadavky
- Případy užití
- Plán projektu
- Akceptační protokol
- Seznam testovacích nápadů
- Standardy tvorby testů v SoapUI

7.5.1 Testovací sada

Testovací sada je pracovní produkt metodiky MMSP, který je tvořen uspořádanými testovacími případy do logické posloupnosti s cílem otestovat určitou část vyvíjeného systému (Rejnková, 2010).

Z pohledu testování webových služeb se jedná o sadu testovacích případů sdružených za účelem otestování určité části webové služby.

Tabulka 13: *Vztahy pracovního produktu Testovací sada (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Analytik testování	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Implementace testů webových služeb • Údržba testů • Provedení akceptačních testů • Provedení testů webových služeb	• Analýza a návrh testů • Údržba testů

7.5.2 Testovací případ

Pracovní produkt definovaný v MMSP, který popisuje kroky, jež by měl tester provést, a způsob, jakým by měl systém zareagovat (Rejnková, 2010).

V rámci metodiky Testování webových služeb nástrojem SoapUI slouží testovací případ jako předpis pro tvorbu testovacího skriptu v nástroji SoapUI.

Tabulka 14: *Vztahy pracovního produktu Testovací případ (Zdroj: autor)*

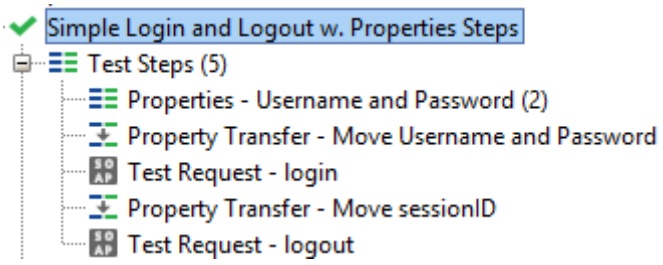
<table>
<tr><td rowspan="2">Role</td><td colspan="2">Odpovědná role:</td></tr>
<tr><td colspan="2">• Analytik testování</td></tr>
<tr><td rowspan="2">Úlohy</td><td>Vstupuje do:</td><td>Výstup z:</td></tr>
<tr><td>• Implementace testů webových služeb
• Údržba testů
• Provedení akceptačních testů
• Provedení testů webových služeb</td><td>• Analýza a návrh testů
• Údržba testů</td></tr>
</table>

7.5.3 Testovací skript

Test vytvořený v SoapUI na základě testovacího případu. Jedná se o zápis kroků, které musí provést nástroj SoapUI, aby otestoval daný testovací případ.

Při tvorbě testovacího skriptu se používá prostředků nástroje SoapUI – jedná se například o dotaz typu SOAP nebo REST, kontroly, proměnné, skriptovací jazyk Groovy atd. (více o tvorbě testů v nástroji SoapUI obsahuje příloha A).

Ukázku SoapUI testovacího skriptu zobrazuje obrázek 15.



Obrázek 15: *Ukázka testovacího skriptu v nástroji SoapUI (Zdroj: autor)*

Tabulka 15: *Vztahy pracovního produktu Testovací skript (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Tester služeb	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Provedení testů webových služeb • Údržba testů • Provedení akceptačních testů	• Implementace testů webových služeb • Údržba testů

7.5.4 Testovací data

Pracovní produkt metodiky MMSP. Testovací data slouží jako vstup pro testovací skripty, respektive testovací případy. Odpovědnost za testovací data má analytik testování a tester služeb.

V případě, že testovací data nejsou k dispozici, není možné testy provést. Zároveň, pokud jsou použity nesprávné testovací data (data nejsou například aktuální nebo skript volá jiné testovací data než má), může dojít ke zkreslení výsledků testu (test, který by neprošel, projde a naopak).

Tabulka 16: *Vztahy pracovního produktu Testovací data (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Analytik testování • Tester služeb	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Implementace testů webových služeb • Údržba testů • Provedení akceptačních testů • Provedení testů webových služeb	• Analýza a návrh testů • Implementace testů webových služeb • Údržba testů

7.5.5 Záznam výsledků testů

Pracovní produkt MMSP, jenž sjednocuje informace o provedených testech a jejich výsledcích. Měl by obsahovat i podrobný report chyb (Rejnková, 2010).

Za výsledky testů webových služeb je zodpovědný tester webových služeb. Report obsahuje podrobné výsledky testů webových služeb a nalezené chyby. Tento report slouží manažerovi testování jako podklad k vytvoření zprávy o výsledcích testování.

Tabulka 17: *Vztahy pracovního produktu Záznam výsledků testů (Zdroj: autor)*

Role	**Odpovědná role:**	
	● Tester služeb	
Úlohy	**Vstupuje do:**	**Výstup z:**
	● Řízení testů	● Provedení testů webových služeb ● Provedení unit testů

7.5.6 Plán testů

Produkt MMSP, měl by obsahovat veškeré důležité informace o testování, které by mělo být pro danou verzi vyvíjeného IS/ICT provedeno (Rejnková, 2010).

Stěžejní dokument procesu testování, vytvářený test manažerem v rámci úlohy plánování testů. Dokument není pro testy webových služeb unikátní, ale jedná se o dokument, který pokrývá celou disciplínu testování. Je v něm však třeba definovat strategii testování webových služeb.

Tabulka 18: *Vztahy pracovního produktu Plán testů (Zdroj: autor)*

Role	**Odpovědná role:**	
	● Manažer testování	
Úlohy	**Vstupuje do:**	**Výstup z:**
	● Řízení testů	● Plánování testů

7.5.7 Záznam o průběhu testování

Záznam o průběhu testování obsahuje seznam provedených testů (testovacích sad, případů a skriptů) s jejich výsledky, záznam nalezených a nahlášených chyb a případné problémy, s kterými bylo nutné se vypořádat.

Tento dokument je předáván projektovému vedení, kde je společně s dalšími odpovědnými osobami analyzován a konzultován. Na základě tohoto dokumentu může dojít k úpravě plánu testů.

Šablonu dokumentu je možné nalézt v příloze D.

Za vytvoření záznamu o průběhu testování odpovídá manažer testování. Dokument vzniká v rámci úlohy řízení testů.

Tabulka 19: *Vztahy pracovního produktu Záznam o průběhu testování (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Manažer testování	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Plánování testů	• Řízení testů

7.5.8 Unit test

Pracovní produkt MMSP. Unit testy jsou vytvářeny vývojářem. V jejich rámci dochází k individuálnímu testování operací a metod webové služby s cílem otestovat, zda se jednotlivé komponenty webové služby chovají v izolovaném prostředí, jak je požadováno, a zda je možné webovou službu zkompilovat. Unit testy se obvykle skládají ze sady automatizovaných testů.

Tabulka 20: *Vztahy pracovního produktu Unit test (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Vývojář	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Provedení unit testů • Údržba testů	• Tvorba unit testů • Údržba testů

7.5.9 Požadavky

Požadavky jsou v MMSP definovány jako pracovní produkt, který zachycuje veškeré požadavky (funkční i nefunkční) na vyvíjený systém. Funkční požadavky jsou následně rozpracovány v pracovním produktu *Případy užití* (Rejnková, 2010).

Pracovní produkt *Požadavky* zahrnuje i požadavky na webové služby. Pro analytika testování jsou požadavky a případy užití důležité z hlediska návrhu testovací dokumentace (testovacích sad a případů). Požadavky, respektive případy užití jsou dále používány i pro údržbu testů webových služeb.

Tabulka 21: *Vztahy pracovního produktu Požadavky (Zdroj: autor)*

Role	**Odpovědná role:**	

Úlohy	**Vstupuje do:**	**Výstup z:**
	• Analýza a návrh testů • Údržba testů	---

7.5.10 Případy užití

Dokument případy užití vychází z metodiky MMSP a jedná se o souhrn detailně popisující funkční požadavky na systém (Rejnková, 2010).

Stejně jako požadavky, tak i případy užití jsou pro analytika testování klíčové při tvorbě testovacích sad a případů webových služeb.

Tabulka 22: *Vztahy pracovního produktu Případy užití (Zdroj: autor)*

Role	**Odpovědná role:**	

Úlohy	**Vstupuje do:**	**Výstup z:**
	• Analýza a návrh testů • Údržba testů	---

7.5.11 Plán projektu

Pracovní produkt MMSP. Na základě plánu projektu je schopen manažer testování zpracovat plán testů. Plán projektu obsahuje všechny dokumenty, které jsou v rámci daného projektu postupem času vytvářeny.

Tabulka 23: *Vztahy pracovního produktu Plán projektu (Zdroj: autor)*

Role	**Odpovědná role:**	

Úlohy	**Vstupuje do:**	**Výstup z:**
	● Plánování testů ● Řízení testů	---

7.5.12 Akceptační protokol

Akceptační protokol je výsledkem úlohy provedení akceptačních testů. Protokol obsahuje zprávu o tom, zda je vyvinutý produkt zákazníkem akceptován.

Šablona pracovního produktu je uvedena v příloze E.

Tabulka 24: *Vztahy pracovního produktu Akceptační protokol (Zdroj: autor)*

Role	**Odpovědná role:**	
	● Zákazník	
Úlohy	**Vstupuje do:**	**Výstup z:**
	● Řízení testů	● Provedení akceptačních testů

7.5.13 Seznam testovacích nápadů

Pracovní produkt MMSP. Seznam testovacích nápadů zachycuje možné problémové oblasti vyvíjeného systému, které by bylo vhodné otestovat. Vychází primárně ze specifikovaných případů užití, u kterých by měly být testovány úspěšné a chybové průchody (Rejnková, 2010).

Seznam testovacích nápadů vypracovává analytik testování a je výsledkem úlohy *Analýza a návrh testů*.

Tabulka 25: *Vztahy pracovního produktu Seznam testovacích nápadů (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Analytik testování	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Údržba testů	• Analýza a návrh testů

7.5.14 Standardy tvorby testů v SoapUI

Jedná se o dokument, který vypracovává tester služeb a který definuje standardy a metody přístupu k tvorbě testů v SoapUI. Obsahem tohoto dokumentu by měly být následující body:

- **Verze nástroje** – pro jakou verzi nástroje jsou testy vytvářeny. To má například dopad na možnosti kontrol, které mohou být v rámci testovacích skriptů použity.
- **Struktura testovacích skriptů** – definování struktury testovacích skriptů (dělení na logické celky, stanovení granularity testů).
- **Jmenná konvence** – pravidla pojmenovávání TestSetů, TestCasů, TestStepů, proměnných a dalších objektů, které během tvorby testovacích skriptů mohou vzniknout.
- **Pravidla používání komentářů** – definování pravidel pro komentování částí testovacích skriptů.
- **Pravidla používání proměnných** – proměnné lze v SoapUI definovat na několika úrovních (více viz příručka k SoapUI kapitola A.4.3). Dokument by měl stanovit, kde a jakým způsobem jednotlivé typy proměnných používat.
- **Pravidla konfigurace připojení na externí systémy** – v rámci testovacích skriptů lze definovat napojení na externí systémy. Dokument by měl stanovit, jak tato napojení vytvářet a používat napříč projektem (může se jednat např. o JDBC).
- **Testovací data** – stanovení, jak přistupovat k testovacím datům. Testovací dat lze definovat vně skriptu nebo uvnitř skriptu. Mohou být staticky nebo dynamicky generována.
- **Znovupoužitelnost –** pravidla, jak psát testovací skripty, aby jejich části mohly být přepoužity v jiném testu.
- **Zátěžové testy –** pravidla a koncepty zátěžových testů v SoapUI.
- **Bezpečnostní testy** – pravidla a koncepty bezpečnostních testů v SoapUI.
- **Automatizované spouštění** – určení, zda testovací skripty budou spouštěny automatizovaně nebo manuálně. V případě automatizovaného spouštění by měl dokument

obsahovat i základní popis návrhu a popis toho, jak (technologie), kdy (periodicita) a kde (server, prostředí) se budou testy webových služeb spouštět.

Šablonu dokumentu je možné nalézt v příloze F.

Standardy tvorby testů v SoapUI vypracovává tester služeb a je výsledkem úlohy *Plánování testů*.

Tabulka 26: *Vztahy pracovního produktu Standardy tvorby testů v SoapUI (Zdroj: autor)*

Role	**Odpovědná role:**	
	• Tester služeb	
Úlohy	**Vstupuje do:**	**Výstup z:**
	• Implementace testů webových služeb	• Plánování testů

7.6 Životní cyklus

Předmětem této části je, jak jednotlivé činnosti, role a pracovní produkty popsané výše zapadají do životní cyklu vývoje systému metodiky MMSP.

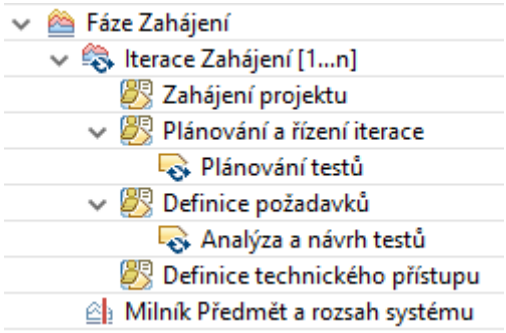
Životní cyklus se skládá ze 4 fází, přičemž jednotlivé fáze mohou dle potřeby projít libovolným počtem iterací. Jednotlivé fáze metodiky:

1. Zahájení
2. Rozpracování
3. Konstrukce
4. Zavedení

7.6.1 Fáze Zahájení

V této fázi z pohledu metodiky testování webových služeb probíhá plánování testů a prvotní analýza a návrh testů.

Fáze zahájení probíhá v iteraci a je zakončena milníkem *Předmět a rozsah systému*. Proces fáze *Zahájení* je znázorněn na obrázku 16:

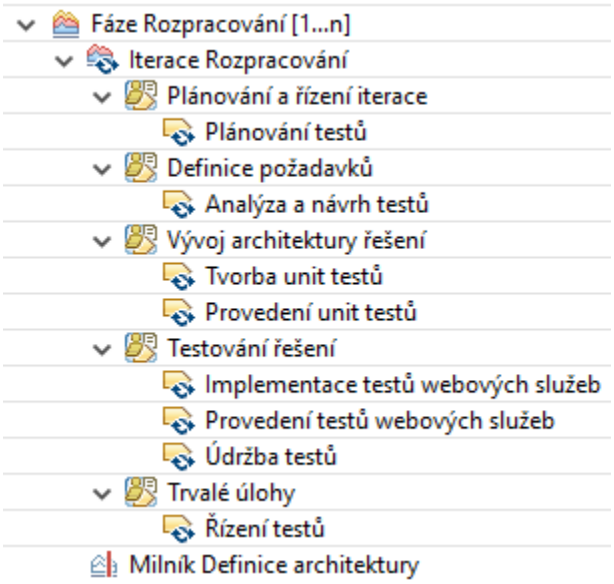


Obrázek 16: *Fáze Zahájení (Zdroj: autor)*

7.6.2 Fáze Rozpracování

V této fázi se z pohledu metodiky testování webových služeb začíná s testováním – probíhá implementace a provádění testů webových služeb a unit testů. Zároveň probíhá údržba testů a z pohledu řízení jsou tyto aktivity podpořeny aktivitou *Řízení testů*.

Fáze zahájení probíhá v iteraci a je zakončena milníkem *Definice architektury*. Proces fáze *Rozpracování* je znázorněn na obrázku 17:

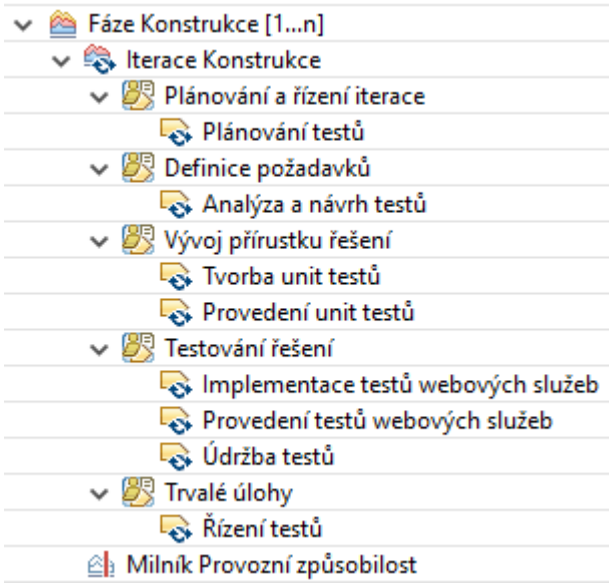


Obrázek 17: *Fáze Rozpracování (Zdroj: autor)*

7.6.3 Fáze Konstrukce

Tato fáze z pohledu metodiky testování webových služeb je hlavní fází testování. Během ní probíhá spouštění unit testů a testů webových služeb. Zároveň na základě úprav a zpřesňování požadavků probíhá tvorba nových testů a revize stávajících.

Fáze zahájení probíhá v iteraci a je zakončena milníkem *Provozní způsobilost*. Proces fáze *Konstrukce* je znázorněn na obrázku 18:

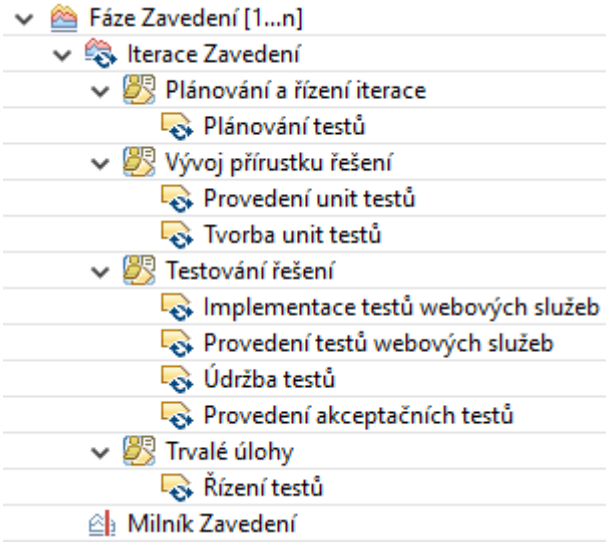


Obrázek 18: *Fáze Konstrukce (Zdroj: autor)*

7.6.4 Fáze Zavedení

V rámci této fáze z pohledu metodiky testování webových služeb probíhají poslední testy se zaměřením na akceptační testy prováděné zákazníkem.

Fáze zahájení probíhá v iteraci a je zakončena milníkem *Zavedení*. Proces fáze *Zavedení* je znázorněn na obrázku 19:



Obrázek 19: *Fáze Zavedení (Zdroj: autor)*

7.7 Příručka SoapUI

Obsahem metodiky Testování webových služeb nástrojem SoapUI je příručka s návody k tvorbě testů v nástroji SoapUI. Vzhledem ke svému rozsahu je příručka uvedena zvlášť v příloze A.

7.8 Koncepty

Sekce *Koncepty* obsahuje vysvětlení několika klíčových pojmů souvisejících s testováním webových služeb. Obsahem jsou některé pojmy, které byly vysvětleny v rámci teoretické části této práce, a proto je sekce *Koncepty* uvedena zvlášť v příloze B.

7.9 Nástroje

Kapitola se věnuje popisu nástrojů stěžejních pro testování webových služeb. Obsahem je popis nástroje SoapUI, který vychází z kapitoly 5.2, a proto je sekce *Nástroje* uvedena zvlášť v příloze C.

8 Závěr

Hlavním cílem této diplomové práce bylo vytvořit metodiku testování webových služeb, která by poskytla ucelený návod, jak k testování webových služeb přistupovat. Tohoto cíle bylo dosaženo vytvořením metodiky Testování webových služeb nástrojem SoapUI společně s příručkou pro tvorbu testů v nástroji SoapUI.

Hlavnímu cíli nejprve předcházelo několik cílů dílčích. Prvním dílčím cílem bylo zpracování rešerše z oblasti testování webových služeb, v rámci níž byly identifikovány podstatné zdroje, ze kterých tato práce vycházela. Následovalo objasnění základních pojmů, které s oblastí testování webových služeb souvisí. Těmto pojmům se věnovaly především kapitoly 3 a 4, které objasňovaly termíny testování a webová služba. Dalším dílčím cílem bylo představení nástroje SoapUI, kterému se věnovala kapitola 5.2. Mezi další dílčí cíle patřilo navržení metodiky a její vytvoření v nástroji EPFC. Těmto dílčím cílům předcházelo objasnění pojmu metodika a představení nástroje EPFC v kapitole 6. Popisu vytvořené metodiky se věnovala kapitola 7. V rámci této kapitoly byly popsány hlavní prvky metodiky Testování webových služeb nástrojem SoapUI. Zbylé prvky metodiky byly popsány zvlášť v přílohách. Nejvýznamnější přílohou pro vypracovanou metodiku je příloha A, která obsahuje příručku pro tvorbu testů v nástroji SoapUI. V příloze B jsou následně vysvětleny hlavní pojmy podstatné pro testování webových služeb. Příloha C obsahuje popis nástrojů, které jsou pro metodiku důležité. Zbylé přílohy (D, E a F) obsahují vybrané šablony pracovních produktů.

Za hlavní přínos této diplomové práce považuji vytvoření metodiky Testování webových služeb nástroje SoapUI a vytvoření příručky k nástroji SoapUI, ve které je uvedena řada návodů a postupů pro práci s tímto nástrojem. Tato metodika může sloužit pro zavedení testování webových služeb tam, kde s testováním webových služeb nejsou zatím žádné nebo malé zkušenosti. Zároveň příručka uvedená v příloze A může sloužit jako výukový materiál pro zaškolení testera s prací v nástroji SoapUI.

Vzhledem k tomu, že je metodika publikována pomocí nástroje EPFC, může být jednoduše upravována pro potřeby konkrétních jedinců nebo organizací a dále rozšiřována například o další návody v SoapUI či jiném nástroji pro testování webových služeb.

9 Terminologický slovník

Tabulka 27: *Terminologický slovník*

Termín	Zkratka	Význam [zdroj]
Application Program Interface	API	Rozhraní pro programování aplikací, které nabízí sbírku procedur, funkcí či tříd nějaké knihovny (IT-Biz, 2015).
Assertion		V rámci SoapUI označuje kontrolu, kterou je možné při tvorbě testů vytvořit (autor).
Cloud Computing	CC	Cloud computing je sdílení hardwarových i softwarových prostředků pomocí sítě (Cloud Computing, 2010).
Continual Integration (kontinuální integrace)	CI	Spočívá v průběžném provádění úkonů integrace zdrojového kódu, testování, sestavení a nasazení v reakci na každou vývojářem odevzdávanou změnu zdrojového kódu projektu a to za využití nástrojů na podporu vývoje a testování dodržováním stanoveného postupu automatizovaně (Ducho a Zajíček, 2015).
Eclipse Process Framework Composer	EPFC	Nástroj pro správu metodik (autor).
Endpoint		Označuje cílovou adresu, na kterou bude dotaz zaslán (autor).
Enterprise Service Bus	ESB	ESB je architektonický model pro propojení služeb například v SOA architektuře (Rouse, 2007).
Extensible Markup Language	XML	Rozšiřitelný značkovací jazyk. Dnes se používá především pro snadnou výměnu informací a komunikaci nezávislou na konkrétní aplikaci či platformě (Adaptic, 2015).
Front-end	FE	Viditelná část aplikace uživateli (autor).
Geo Services		Webová služba, která umožňuje vrátit informace o místě na základě PSČ, vzdálenost mezi dvěma lokacemi, vrátit lokace na základě PSČ nebo vrátit města a místa blízko určité lokace (autor).
Graphical User Interface	GUI	Jedná se o uživatelské prostředí, které je obohaceno o grafické elementy jako např. okno, tlačítko nebo ikona (TechTerms, 2015).
Groovy Script		Skriptovací jazyk (autor).

Termín	Zkratka	Význam [zdroj]
Hypertext Transfer Protocol	HTTP	Internetový protokol pro přenos dat mezi webovým serverem a klientem (Techopedia, 2015).
Java Database Connectivity	JDBC	Poskytuje základní rozhraní pro unifikovaný přístup k databázím (Šeda, 2003).
Java Message Service	JMS	API, které umožňuje vytvářet, posílat a číst zprávy (JavaTpoint, 2014).
JavaScript	JS	Skriptovací jazyk, který se obvykle používá při vývoji internetových stránek (TechTerms, 2014).
JavaScript Object Notation	JSON	Formát pro výměnu dat. Během posledních několika let se vedle XML zařadil mezi nejdůležitější formáty pro výměnu dat na webu (Hassman, 2008).
Methodology Framework for IS/ICT Systems	MeFIS	Metodický rámec IS/ICT, který je vytvářen na katedře informačních technologií VŠE v Praze (Buchalcevová, 2005)
Metodika budování IS/ICT		Principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení (Buchalcevová, 2005).
Metodika pro Malé Softwarové Projekty	MMSP	Metodika Petry Rejnkové (2010), která vznikla v rámci diplomové práce „Lokalizace a přizpůsobení metodiky OpenUP“ (autor).
Mock služba		Typ zástupného objektu, který simuluje chování určitého objektu, v tomto případě webové služby (Kankanamge, 2012).
Open Source	OS	Software, který je poskytován zdarma s dostupným zdrojovým kódem (autor).
Open Unified Process	OpenUP	Agilní metodika se zaměřením na iteraci, inkrementální vývoj, kolektivní práci a komunikaci v týmu (autor).
Quality Assurence (zajišťování kvality)	QA	Definuje standardy, metodiky a metriky, které by měly vést k vyšší kvalitě produktu (Roudenský a Havlíčková, 2013).
Regular expression (regulární výraz)		Slouží k vyhledání části řetězce, kterou předem (úplně) neznáme nebo která může mít více podob (Watzke, 2008).

Termín	Zkratka	Význam [zdroj]
Remote Procedure Call	RPC	Jedná se o komunikace typu klient-server pro distribuované výpočty a další. Požadavek je obvykle zaslán vzdálenému systému, který vykoná označenou proceduru s dodanými argumenty a výsledek vrátí volajícímu (Říha a Klaška, 2015).
Representational State Transfer	REST	Alternativa k protokolu SOAP. Jedná se o architekturu postavenou na principech popisující lokalizaci a přístup ke zdrojům (Barry a Dick, 2013).
Request (dotaz)		Zpráva odeslaná na webovou službu (autor).
Response (odpověď)		Zpráva přijatá od webové služby (autor).
Secure Sockets Layer	SSL	Bezpečnostní protokol pro šifrovanou komunikaci mezi klientem a serverem (Dicert, 2015).
Selenium		Nástroj pro automatické testování webových aplikací (autor).
Service Oriented Architecture (služebně orientovaná architektura)	SOA	Služebně orientovaná architektura, je popisována jako strategie budování služebně orientovaných aplikací (cílem je poskytovat služby, které mohou být užívány jinými službami) (Bozkurt, Harman a Hassoun, 2010).
Service Oriented Computing (služebně orientované výpočty)	SOC	Služebně orientované výpočty jsou výpočetním konceptem, který využívá služeb jako lehkého stavebního prvku, který umožňuje vývoj rychlých, levných a jednoduchých kompozicí distribuovaných aplikací (Bozkurt, Harman a Hassoun, 2010).
Simple Mail Transfer Protocol	SMTP	Protokol pro přenos elektronických zpráv (KORRA'TI, 2005).
SOAP		Protokol pro zasílání zpráv mezi webovými službami založený na XML (Breshne, Fuhrer a Pasquier, 2009).
SoapUI		Nástroj pro tvorbu komplexních funkčních a nefunkčních testů, především webových služeb (autor).
Software	SW	Programové vybavení počítače (ITBiz, 2015).
TestRunner		Nástroj pro automatizované spouštění testů v SoapUI (autor).
Uniform Resource Identifier	URI	Obecně použitelná množina všech jmen/adres, které se vztahují k nějakému zdroji (Gála, 2006).

Termín	Zkratka	Význam [zdroj]
Universal Description, Discovery and Integration	UDDI	Adresář pro registraci a vyhledávání webových služeb (Barry a Dick, 2013).
User Interface (uživatelské rozhraní)	UI	Uživatelským rozhraním se označuje rozhraní, prostřednictvím kterého uživatel komunikuje se softwarovou aplikací nebo hardwarovým zařízením (TechTerms, 2009).
V-model		Model životního cyklu vývoje softwaru (autor).
Web Application Description Language	WADL	Dokument, který stejně jako WSDL slouží k popisu webové služby (Kankanamge, 2012).
Web Service (webová služba)	WS	Stavební blok pro vytváření distribuovaných aplikací, které mohou být publikovány a konzumovány prostřednictvím Internetu či intranetových sítí (Papazoglou, 2008).
Web Services Description Language	WSDL	XML dokument, který slouží k popisu rozhraní webové služby (Kankanamge, 2012).
Workspace (pracovní prostor)		V rámci SoapUI pracovní prostor sloužící k organizaci, nastavování a správě jednotlivých projektů (autor).
World Wide Web Consorcium	W3C	Mezinárodní sdružení zabývající se webovými standardy (autor).
XML Path Language	XPath	Jazyk, který slouží k procházení XML dokumentu (autor).

10 Seznam literatury

Adaptic. *XML* [online]. 2015 [cit. 2015-11-15]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/xml/

ANDERSON, Rupert. SoapUI Cookbook. Birmingham: Packt Publishing, 2015. ISBN 978-1-78439-421-9.

AZZAM, Sana, Izzat ALSMADI a Mohammed AL-KABI. *WEB SERVICES TESTING CHALLENGES AND APPROACHES* [online]. In: . 2012, s. 291-296 [cit. 2015-11-12]. Dostupné z: https://www.researchgate.net/publication/241194924_WEB_SERVICES_TESTING_CHALLENGES_AND_APPROACHES

BARRY, Douglas K. a with David DICK. *Web services, service-oriented architectures, and cloud computing: the savvy manager's guide*. 2nd ed. San Francisco, Calif: Morgan Kaufmann, 2013. ISBN 9780123983572.

BOROVCOVÁ, Anna. *Testování webových aplikací* [online]. Praha, 2008 [cit. 2015-11-04]. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/46536/. DP. Univerzita Karlova v Praze. Vedoucí práce Vladan Majerech.

BOZKURT, Mustafa, Mark HARMAN a Youssef HASSOUN. Testing web services: a survey. In: *Testing web services: a survey* [online]. 2010 [cit. 2015-09-26]. Dostupné z: https://www.researchgate.net/publication/228920258_Testing_web_services_a_survey

BRUCKNER, Tomáš. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada, 2012, 357 s. Management v informační společnosti. ISBN 9788024741536.

BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. *Příklady modelů analýzy a návrhu aplikace v UML*. Vyd. 1. Praha: Oeconomica, 2013, 197 s. ISBN 9788024519227.

BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005, 163 s. Management v informační společnosti. ISBN 8024710757.

Cloud Computing. *Co to je Cloud computing?* [online]. 2010 [cit. 2015-11-15]. Dostupné z: http://www.cloudcomputing.cz/index.html

Codeception: Testing WebServices. *Codeception: Testing WebServices* [online]. 2015 [cit. 2015-10-19]. Dostupné z: http://codeception.com/docs/10-WebServices

DASARI, Suresh. *What is the difference between Unit testing, Assembly testing, Integration Testing and Regression testing*. Aspdotnet-Suresh [online]. 2010 [cit. 2015-11-04]. Dostupné z: http://www.aspdotnet-suresh.com/2010/09/whats-difference-between-unit.html

Dicert. *What Is SSL (Secure Sockets Layer)* [online]. 2015 [cit. 2015-11-15]. Dostupné z: https://www.digicert.com/ssl.htm

DUCHO, Pavel a Tomáš ZAJÍČEK. *Výhody kontinuální integrace a automatizovaného deploymentu z pohledu webové integrace* [online]. 2015 [cit. 2015-11-21]. Dostupné z: http://www.web-integration.info/cs/blog/vyhody-kontinualni-integrace-a-automatizovaneho-deploymentu-z-pohledu-webove-integrace/

ECLIPSE, *Eclipse Process Framework Composer Help* [program]. Version 1.5.1.7. 2014.

ELKSTEIN. *Learn REST: A Tutorial* [online]. 2010 [cit. 2015-11-11]. Dostupné z: http://rest.elkstein.org/2008/02/documenting-rest-services-wsdl-and-wadl.html

ERL, Thomas. *SOA: principles of service design*. Upper Saddle River: Prentice Hall, 2008, xxxii, 573 s. ISBN 9780132344821.

GÁLA, Libor. Kurz 4it383 Web design. *URI, URL a URN* [online]. 2006 [cit. 2015-11-21]. Dostupné z: http://nb.vse.cz/~gala/4it383/pomucky/uri.htm

GURU99,. SoapUI Tutorials for Beginners. *SoapUI Tutorials for Beginners* [online]. 2015 [cit. 2015-09-26]. Dostupné z: http://www.guru99.com/soapui-tutorial.html

HASSMAN, Martin. *JSON : jednotný formát pro výměnu dat* [online]. 2008 [cit. 2015-11-15]. Dostupné z: https://www.zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat/

HAUPTVOGEL, Roman. *Metodika budovania IS v architektúre orientovanej na služby*. Praha, 2013. Disertační práce. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

HUW, James. PLANIT. *Web Services Testing* [online]. 2013 [cit. 2015-11-11]. Dostupné z: http://www.planit.net.au/resource/web-services-testing/

ITBiz. *API* [online]. 2015 [cit. 2015-11-21]. Dostupné z: http://www.itbiz.cz/slovnik/informacni-technologie-it/api

ITBiz. *SW* [online]. 2015 [cit. 2015-11-21]. Dostupné z: http://www.itbiz.cz/slovnik/informacni-technologie-it/software

JavaTpoint. *JMS Tutorial* [online]. 2014 [cit. 2015-11-15]. Dostupné z: http://www.javatpoint.com/jms-tutorial

JOSUTTIS, Nicolai. *SOA in practice*. Beijing: O'Reilly, c2007, xv, 324 s. ISBN 9780596529550.

KANKANAMGE, Charitha. Web services testing with soapUI. Birmingham, [Eng.]: Packt Publishing, 2012, 310 p. ISBN 9781849515665.

KORRA'TI, R'ykandar. TechNet Magazine. *How Simple is SMTP?* [online]. 2005 [cit. 2015-11-21]. Dostupné z: https://technet.microsoft.com/en-us/magazine/2005.11.howitworkssmtp.aspx

KUBA, Martin. Web Services [online]. In: . Brno, 2006, s. 93-112 [cit. 2015-11-12]. Dostupné z: http://dior.ics.muni.cz/~makub/soap/MartinKuba_WebServices_Datakon2006_clanek.pdf

MALÝ, Martin. Zdroják: REST: architektura pro webové API. *REST: architektura pro webové API* [online]. 2009 [cit. 2015-10-21]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

MARTINA, Hanzlíková. *Testování bezpečnosti webových aplikací* [online]. Praha, 2013 [cit. 2015-11-04]. Dostupné z: https://www.vse.cz/vskp/36619_testovani_bezpecnosti_webovych_aplikaci. BP. Vysoká škola ekonomická v Praze. Vedoucí práce Jan Mittner.

MUELLER, John. Software Quality Matters Blog: Understanding SOAP and REST Basics And Differences. *Software Quality Matters Blog* [online]. 2013 [cit. 2015-10-21]. Dostupné z: http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/

PAPAZOGLOU, M. *Web services and SOA: principles and technology*. 2nd ed. New York: Pearson Education, 2012, xliv, 812 p. ISBN 9780273732167.

PAPAZOGLOU, Michael P. *Web services: principles and technology*. Harlow: Pearson Education, 2008, xxxii, 752 s. ISBN 9780321155559.

PATTON, Ron. *Testování softwaru*. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 8072266365.

POSPÍŠIL, Marek. *Eclipse Process Framework Composer - nástroj na správu metodiky*. Praha, 2009. Dostupné také z: http://www.vse.cz/vskp/eid/20153. Bakalářská práce. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

PTÁČEK, Martin. *Web Services* [online]. 2007 [cit. 2015-11-12]. Dostupné z: https://cw.fel.cvut.cz/wiki/_media/courses/x33eja/2010-zs-prednaska-8-web-services.pdf

REJNKOVÁ, Petra. *Lokalizace a přizpůsobení metodiky OpenUP*. Praha, 2010. Dostupné také z: https://www.vse.cz/vskp/28286_lokalizace_a%C2%A0prizpusobeni_metodiky_openup. Diplomová práce. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

REPÁŇOVÁ, Zdeňka. *Testování webových služeb*. Pardubice, 2014. Dostupné také z: https://dk.upce.cz/handle/10195/58874. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Zdeněk Šilar.

ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce testováním. 1. vyd. Brno: Computer Press, 2013, 208 s. ISBN 9788025138168.

ROUSE, Margaret. *Enterprise service bus (ESB) definition* [online]. 2007 [cit. 2015-11-15]. Dostupné z: http://searchsoa.techtarget.com/definition/enterprise-service-bus

ŘÍHA, Petr a Luboš KLAŠKA. *RPC, Remote Procedure Call* [online]. 2015 [cit. 2015-11-15]. Dostupné z: http://www.svetsiti.cz/slovnik.asp?hid=RPC-Remote-Procedure-Call

SLÁDKOVÁ, Lenka. *Testování atomických softwarových služeb*. Praha, 2013. Dostupné také z: https://www.vse.cz/vskp/36885_testovani_atomickych_softwarovych_sluzeb. Bakalářská práce. Vysoká škola ekonomická v Praze. Vedoucí práce Libor Gála.

SMARTBEAR. *SoapUI 101: Beginner's Guide to Functional Testing* [online]. 2013, 55 s. [cit. 2015-09-26]. SB-SUI-20130925-V1-SoapUI101. Dostupné z: http://www2.smartbear.com/rs/smartbear/images/SmartBear-SoapUI-101-eBook.pdf

SMARTBEAR. *What is soapUI? | About SoapUI* [online]. 2015 [cit. 2015-09-26]. Dostupné z: http://www.soapui.org/about-soapui.html

SoapUI vs. SoapUI NG Pro. SMARTBEAR. *Compare SoapUI and SoapUI NG Pro | About SoapUI NG Pro* [online]. 2015 [cit. 2015-09-28]. Dostupné z: http://www.soapui.org/about-soapui-pro/product-comparison/soapui-and-soapui-pro.html

SW Testování: SoapUI prakticky [online]. 2009 [cit. 2015-11-08]. Dostupné z: http://www.swtestovani.cz/index.php/testovaci-nastroje

ŠEDA, Jan. *Úvod do JDBC* [online]. 2003 [cit. 2015-11-15]. Dostupné z: https://www.interval.cz/clanky/uvod-do-jdbc/

Techopedia. *Hypertext Transfer Protocol (HTTP)* [online]. 2015 [cit. 2015-11-15]. Dostupné z: https://www.techopedia.com/definition/2336/hypertext-transfer-protocol-http

TechTerms. *GUI* [online]. 2015 [cit. 2015-11-15]. Dostupné z: http://techterms.com/definition/gui

TechTerms. *JavaScript* [online]. 2014 [cit. 2015-11-15]. Dostupné z: http://techterms.com/definition/javascript

TechTerms. *User Interface* [online]. 2009 [cit. 2015-11-15]. Dostupné z: http://techterms.com/definition/user_interface

VACHALEC, Vladan. *Testování a kvalita softwaru v metodikách vývoje softwaru*. Praha, 2014. Dostupné také z: https://www.vse.cz/vskp/40451_testovani_a%C2%A0kvalita_softwaru_v%C2%A0metodikach_vyvoje_softwaru. Diplomová práce. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

VODRÁŽKA, Michal. *Způsoby definování požadavků pro výkonnostní testování softwaru* [online]. Praha, 2015 [cit. 2015-11-04]. Dostupné z: https://www.vse.cz/vskp/45290_zpusoby_definovani_pozadavku_pro_vykonnostni_testovani_softwaru. DP. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

W3C. *Web Services Architecture* [online]. 2004 [cit. 2015-11-11]. Dostupné z: http://www.w3.org/TR/ws-arch/#whatis

W3C. *XML Path Language (XPath) 3.1* [online]. 2014 [cit. 2015-11-14]. Dostupné z: http://www.w3.org/TR/xpath-31/

WATZKE, David. Regulární výrazy. *Abc Linux* [online]. 2008 [cit. 2015-11-14]. Dostupné z: http://www.abclinuxu.cz/clanky/programovani/regularni-vyrazy

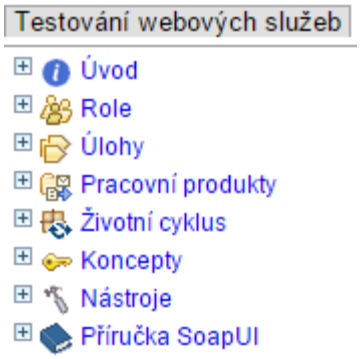
11 Seznam obrázků a tabulek

11.1 Seznam obrázků

11.2 Seznam tabulek

Příloha A: Příručka SoapUI

Příloha popisuje práci s programem SoapUI. Příručka SoapUI je element, který je součástí metodiky Testování webových služeb nástrojem SoapUI, viz obrázek 20.



Obrázek 20: *Prvky metodiky Testování webových služeb nástrojem SoapUI (Zdroj: autor)*

Veškeré ukázky jsou z nástroje SoapUI Open Source verze 5.2.1 a z operačního systému Microsoft Windows.

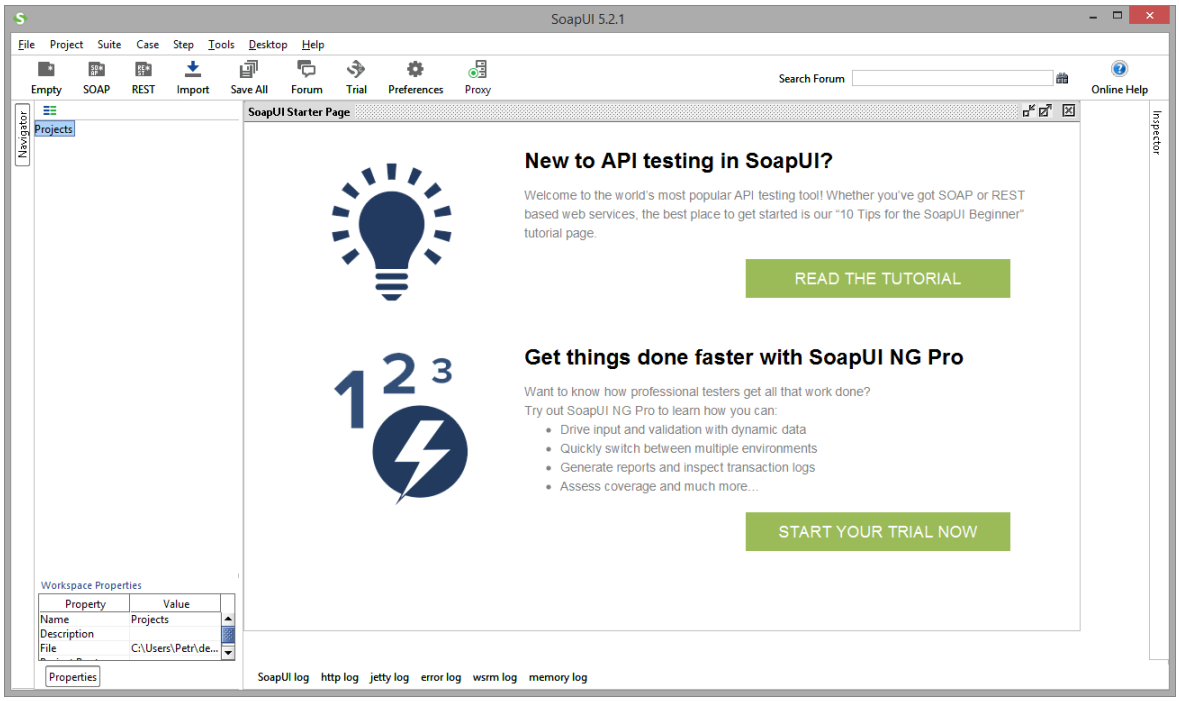
A.1 Instalace a úvod k SoapUI

Před samotnou tvorbou testů webových služeb je nutné nainstalovat nástroj SoapUI Open Source. Nástroj je zdarma ke stažení na webových stránkách:

- http://www.soapui.org/downloads/soapui/open-source.html.

K dispozici je 32bitová nebo 64bitová verze a spustitelný instalátor (exe) nebo archiv (zip), po jehož rozbalení je možné program bez instalace spustit (souborem v \SoapUI-5.2.1\bin\soapui.bat). Po spuštění nástroje je zobrazeno uživatelské prostředí, viz obrázek 21. Uživatelské prostředí nástroje SoapUI se skládá ze tří hlavních sekcí.

V horní části je hlavní nabídka, která umožňuje zakládat nové SOAP a REST projekty, měnit nastavení programu a obsahuje další nabídky a odkazy na hlavní funkce nástroje. V levé části programu je pracovní prostor (workspace), který slouží k organizaci, nastavování a správě jednotlivých projektů. Největší část uživatelského prostředí zabírá prostor, ve kterém je po prvním spuštění uvítací stránka SoapUI (SoapUI Starter Page). Tento prostor je určen pro zobrazování jednotlivých projektů, dotazů (request), odpovědí (response) a testů webových služeb.



Obrázek 21: *Uživatelské prostředí SoapUI OS (Zdroj: autor)*

A.2 Založení projektu

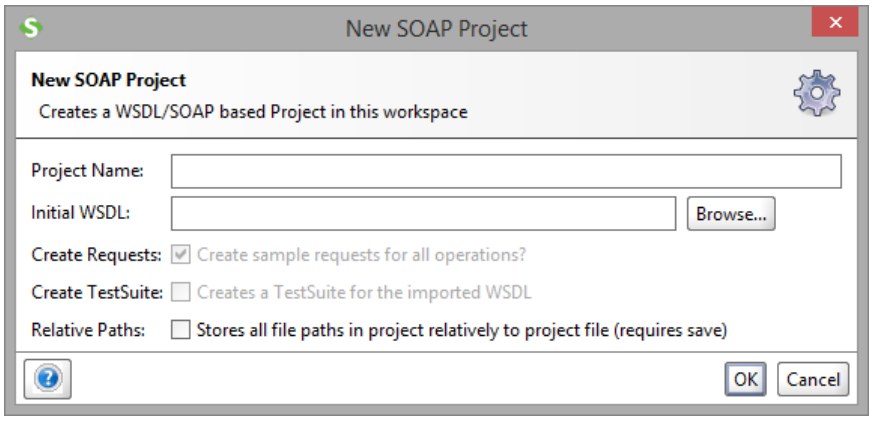
Projekt lze založit buď jako nový projekt nebo již jednou vytvořený projekt naimportovat. V případě, že chceme projekt naimportovat, zvolíme z kontextové nabídky *File* a *Import Project*. Projekty se ukládají ve formátu *XML*.

V opačném případě zvolíme *File* a podle toho, zda chceme SOAP nebo REST webovou službu, zvolíme *New SOAP Project*, respektive *New REST Project*.

A.2.1 SOAP

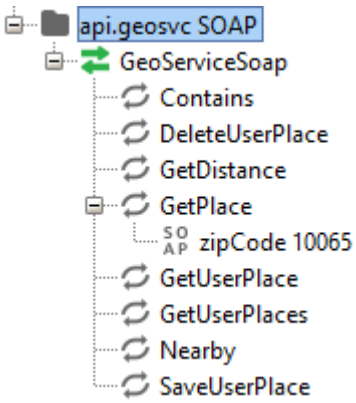
V případě zvolení SOAP je potřeba zadat WSDL webové služby, viz obrázek 22. WSDL webové služby Geo Services se nachází na:

- http://api.geosvc.com/geosvc_api/soap?singleWsdl.



Obrázek 22: *Založení nového SOAP projektu (Zdroj: autor)*

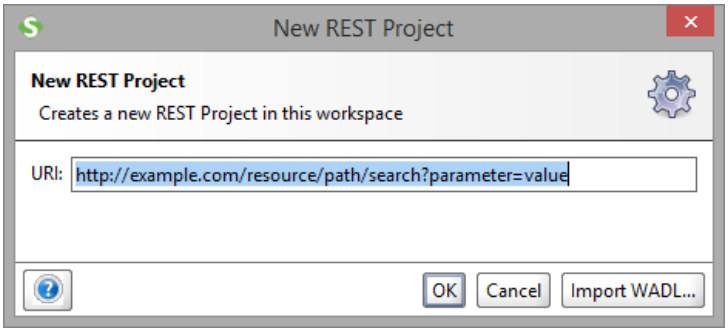
Na základě WSDL je následně vygenerován projekt, obsahující veškeré metody, které je možné v rámci webové služby dotazovat. Na obrázku 23 je projekt webové služby Geo Services s jedním dotazem na metodu *GetPlace*.



Obrázek 23: *SOAP projekt webové služby Geo Services (Zdroj: autor)*

A.2.2 REST

Jak je patrno na obrázku 24, v případě REST je nutné zadat URI konkrétního zdroje webové služby (případně je možné zadat i WADL, ale na rozdíl od SOAP to není povinné).

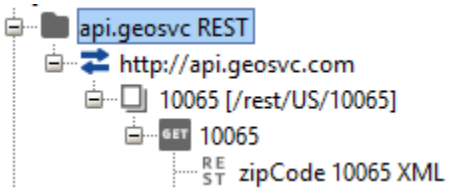


Obrázek 24: *Založení nového REST projektu (Zdroj: autor)*

Příklad URI webové služby Geo Services (*Single Place* s parametry {COUNTRYCODE} pro kód lokace (US), {POSTALCODE} pro PSČ (10065) a {APIKEY} pro ověření (1a0d749e646e46b1a459b10afb559112)):

- http://api.geosvc.com/rest/US/10065?apikey=1a0d749e646e46b1a459b10afb559112

Na obrázku 25 je projekt v SoapUI pro webovou službu Geo Services vygenerovaný na základě uvedené URI.



Obrázek 25: *REST projekt v SoapUI (Zdroj: autor)*

A.3 Odeslání dotazu na webovou službu v SoapUI

Odeslání dotazu na webovou službu se liší podle toho, zda komunikujeme se službou SOAP nebo REST.

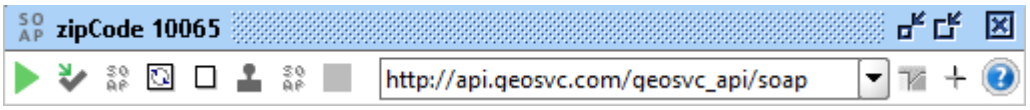
A.3.1 SOAP

V projektu SOAP je třeba vybrat metodu, pro kterou chceme vytvořit dotaz (například *GetPlace*). Následně klineme pravým tlačítkem myši a vybereme *New request* (nebo stiskneme CTRL + N). Dotaz pojmenujeme, přičemž je vhodné volit takový název, který koresponduje s obsahem dotazu. Na základě WSDL je vygenerován ukázkový dotaz, který nemá vyplněné parametry. Dotaz vyplníme například dle výpisu 8 níže.

Výpis 8: SOAP dotaz na službu Geo Services

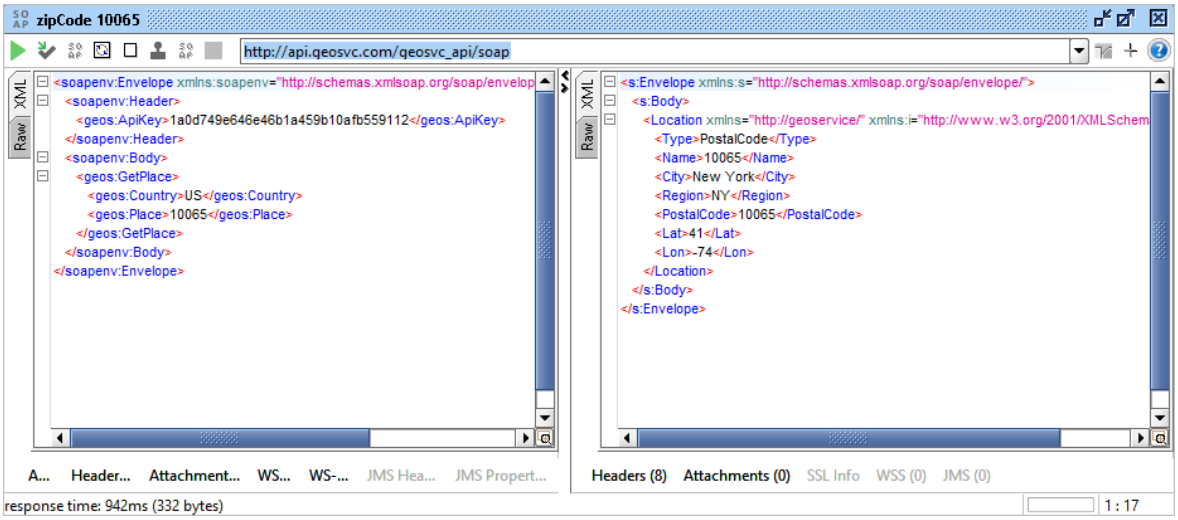
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
   <soapenv:Header>
      <geos:ApiKey>1a0d749e646e46b1a459b10afb559112</geos:ApiKey>
   </soapenv:Header>
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>10065</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```

Po vyplnění dotazu je možné stisknutím tlačítka nebo zkratkou ALT + Enter dotaz odeslat. Dotaz bude odeslán na endpoint, který je vyplněný v poli pro adresu, jak to zobrazuje obrázek 26 (v tomto případě *http://api.geosvc.com/geosvc_api/soap)*.



Obrázek 26: *SOAP dotaz, který obsahuje vyplněný endpoint (Zdroj: autor)*

Po odeslání dotazu je webovou službou vrácena odpověď, která se zobrazí v pravém okně vedle zadaného dotazu (viz obrázek 27).



Obrázek 27: *Dotaz a odpověď na službu Geo Services (Zdroj: autor)*

V okně s dotazem a odpovědí lze přepnout zobrazení z *XML* na *Raw*, kde lze vidět mimo samotného dotazu či odpovědi webové služby i detail komunikace (použitý protokol, kódování atd.). Ukázkový dotaz a odpověď z tohoto přehledu jsou uvedeny ve výpisech 9 a 10.

Výpis 9: SOAP dotaz na službu Geo Services zobrazený jako *Raw*

```
POST http://api.geosvc.com/geosvc_api/soap HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "http://geoservice/GetPlace"
Content-Length: 398
Host: api.geosvc.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
   <soapenv:Header>
      <geos:ApiKey>1a0d749e646e46b1a459b10afb559112</geos:ApiKey>
```

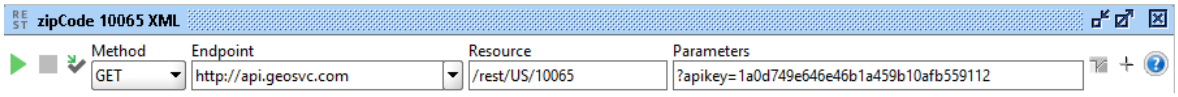
```
   </soapenv:Header>
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>10065</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```

Výpis 10: SOAP odpověď služby Geo Services zobrazena jako *Raw*

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 332
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
Date: Sat, 24 Oct 2015 11:46:32 GMT
Connection: close

<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body><Location
xmlns="http://geoservice/" xmlns:i="http://www.w3.org/2001/XMLSchema-
instance"><Type>PostalCode</Type><Name>10065</Name><City>New
York</City><Region>NY</Region><PostalCode>10065</PostalCode><Lat>41</Lat><Lon>-
74</Lon></Location></s:Body></s:Envelope>
```

A.3.2 REST

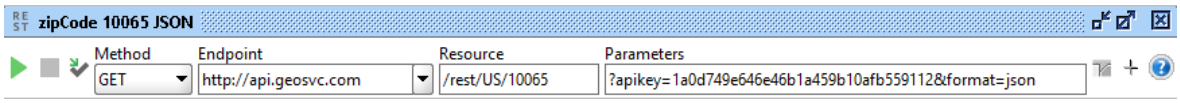
Při vytváření projektu bylo třeba zadat již konkrétní URI zdroje, které slouží jako dotaz na webovou službu. SoapUI tuto URI rozložil na jednotlivé prvky – *Endpoint* (adresu), *Resource* (zdroj) a *Parameters* (parametry), viz obrázek 28.



Obrázek 28: *REST dotaz na Geo Services (Zdroj: autor)*

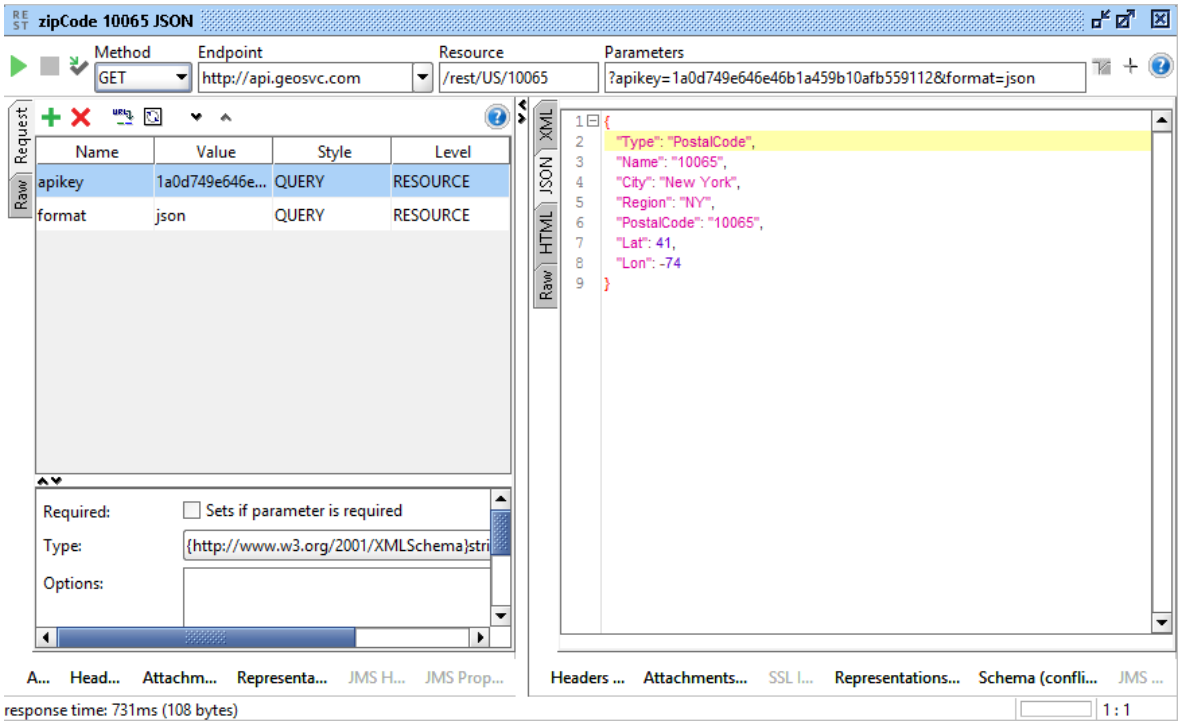
Dotaz lze přímo v SoapUI editovat. Můžeme změnit například metodu – v tomto případě používáme *GET* pro získání informací o lokaci identifikovanou parametry výše. Můžeme změnit i zbylé údaje jako adresu, zdroj nebo parametry.

Například přidáním parametru `format=json` získáme odpověď ve formátu JSON místo XML. Dotaz po přidání parametru pro získání odpovědi ve formátu JSON je na obrázku 29.



Obrázek 29: *REST dotaz na Geo Services s přidaným parametrem format=json (Zdroj: autor)*

Po stisknutí tlačítka nebo zkratkou ALT + Enter je dotaz odeslán na webovou službu Geo Services. Po přepnutí z pohledu XML na JSON je zobrazena odpověď webové služby na daný dotaz, viz obrázek 30.



Obrázek 30: *REST odpověď služby Geo Services s přidaným parametrem format=json (Zdroj: autor)*

Stejně jako v případě SOAP, tak i v případě REST webové služby je možné po přepnutí pohledu do zobrazení *Raw* vidět dotaz a odpověď s dodatečnými informacemi.

Výpis 11: REST dotaz na službu Geo Services zobrazený jako *Raw*

```
GET
http://api.geosvc.com/rest/US/10065?apikey=1a0d749e646e46b1a459b10afb559112&form
at=json HTTP/1.1
Accept-Encoding: gzip,deflate
Host: api.geosvc.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Výpis 12: REST odpověď služby Geo Services zobrazený jako *Raw*

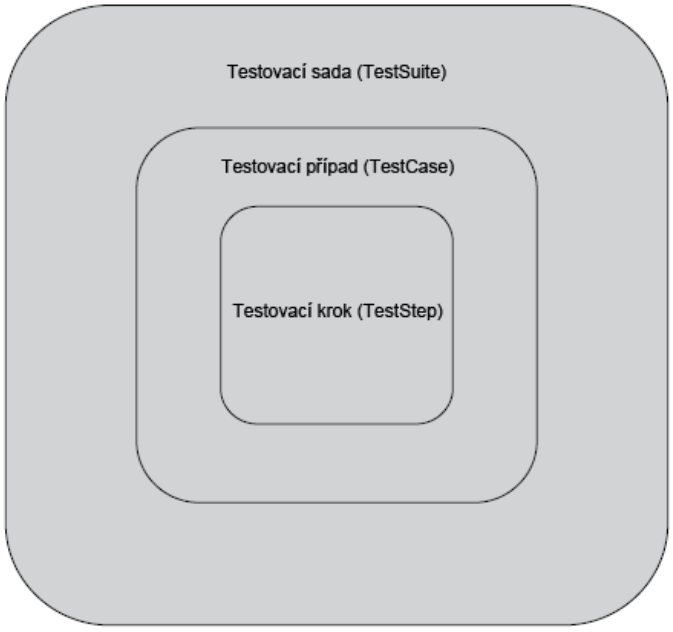
```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 108
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
Date: Sat, 24 Oct 2015 12:23:24 GMT
Connection: close

{"Type":"PostalCode","Name":"10065","City":"New
York","Region":"NY","PostalCode":"10065","Lat":41,"Lon":-74}
```

A.4 Tvorba testů webové služby v SoapUI

V nástroji SoapUI je možné vytvořit tři základní typy testů:

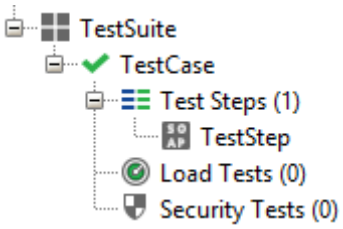
- Funkční testy (Functional Tests)
- Zátěžové testy (Load Tests)
- Bezpečnostní testy (Security Tests)

Testy v SoapUI se vytváří do struktury, která se skládá ze tří úrovní (viz obrázek 31), přičemž struktura je shodná jak pro webové služby SOAP, tak pro webové služby REST (Kankanamge, 2012).



Obrázek 31: *Struktura testů v SoapUI (Zdroj: Kankanamge, 2012)*

1. **Testovací sada (TestSuit)** – souhrn testovacích případů (TestCase) složených do logických celků za účelem otestování specifické funkcionality.
2. **Testovací případ (TestCase)** – souhrn testovacích kroků (TestStep) organizovaný za účelem otestování specifické funkcionality. V rámci testovacího případu (TestCase) lze také definovat zátěžové testy (Load Tests) a bezpečnostní testy (Security Tests), viz obrázek 32.
3. **Testovací krok (TestStep)** – základní prvek funkčních testů v SoapUI. Slouží k tvorbě dotazů, kontrol, proměnných, skriptů a dalších prvků, které souvisejí s tvorbou funkčních testů v SoapUI. Množina testovacích kroků se nazývá *Test Steps* a patří přímo pod testovací případ (TestCase), viz obrázek 32.

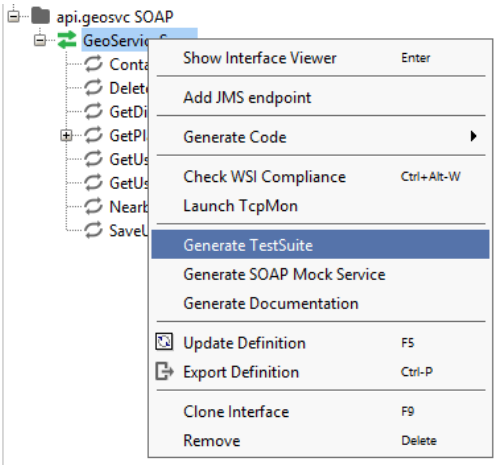


Obrázek 32: *Ukázka struktury testů v SoapUI (Zdroj: autor)*

Vzhledem k tomu, že tvorba testů v SoapUI se pro webové služby SOAP a REST příliš neliší, budou následující návody psány již pouze pro webové služby SOAP pracující v XML.

A.4.1 Vytvoření testovací sady (TestSuite)

Novou testovací sadu lze vytvořit kliknutím pravým tlačítkem myši na rozhraní (interface) webové služby, viz obrázek 33.



Obrázek 33: *Založení testovací sady (TestSuite) (Zdroj: autor)*

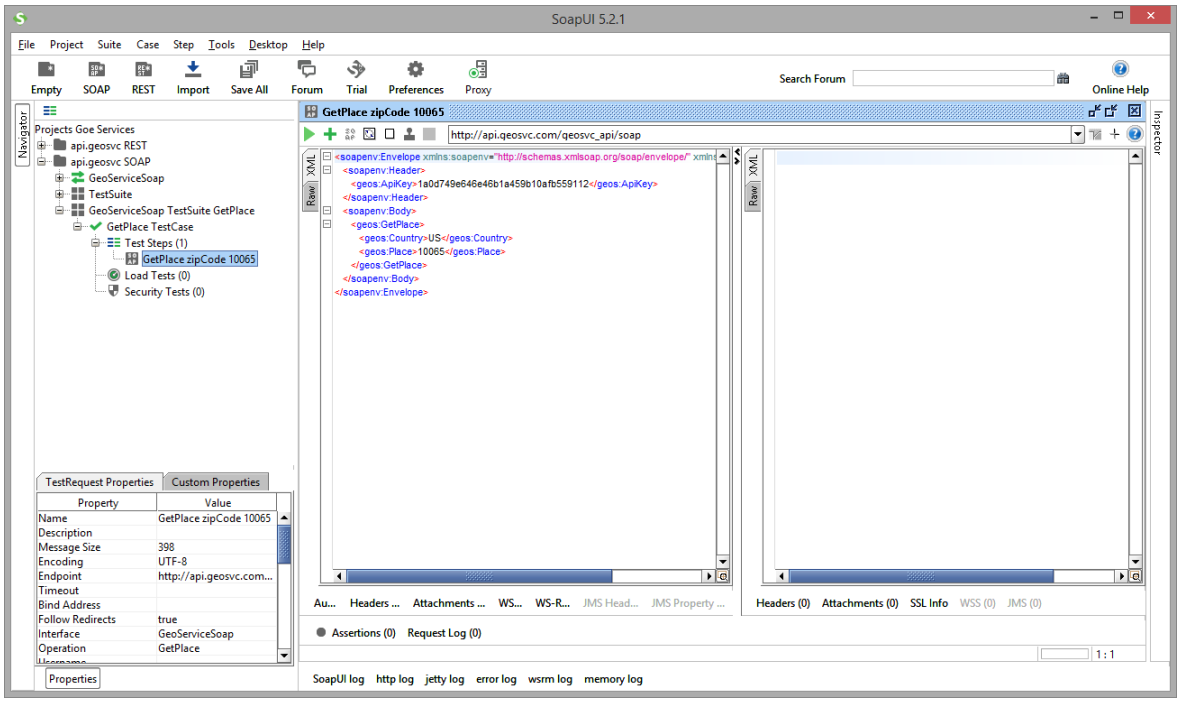
V následném dialogovém okně je třeba zvolit, které operace webové služby budou obsahem testovací sady, zda se má vytvářet testovací případ pro každou operaci zvlášť nebo se má

vytvořit pouze jeden testovací případ a zda se mají vygenerovat prázdné ukázkové dotazy pro vybrané operace nebo se mají použít již vytvořené (pokud existují).

Po potvrzení vygeneruje SoapUI testovací sadu, testovací případ(y) a případně dotaz(y) na webovou službu.

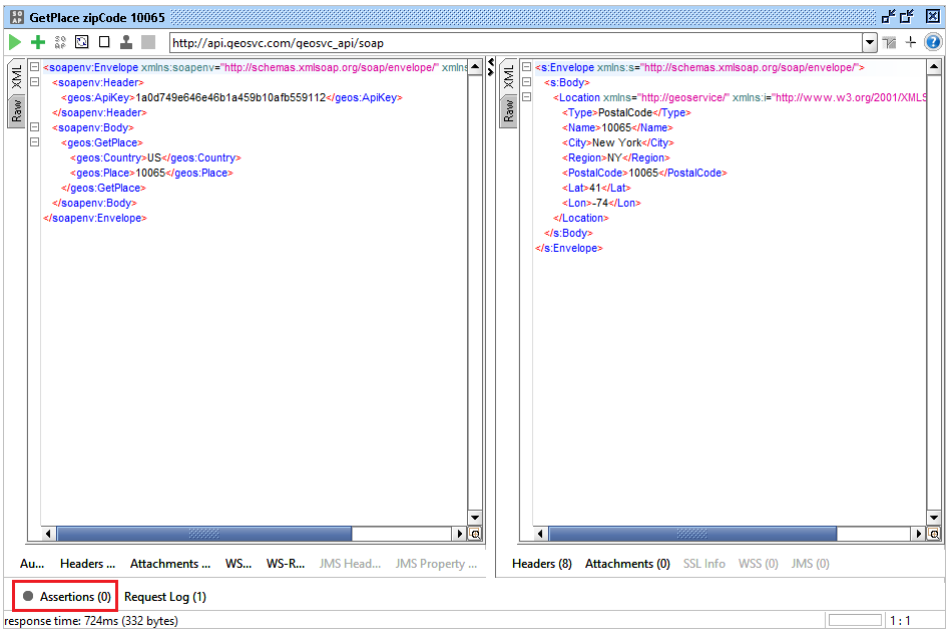
A.4.2 Tvorba kontroly (Assertion)

Jestliže máme v testovacím případu dotaz na webovou službu (viz například obrázek 34 obsahující dotaz z výpisu 8), můžeme dotaz odeslat na webovou službu a následně vytvořit kontrolu (Assertion).



Obrázek 34: *SOAP dotaz v testovacím případu GetPlace TestCase (Zdroj: autor)*

Kontrola se v SoapUI vytvoří tak, že se v dolní části okna s dotazem a odpovědí kline na *Assertions,* viz obrázek 35**,** a následně se klikne na tlačítko ✚.



Obrázek 35: *Tvorba kontroly (Assertion) v SoapUI (Zdroj: autor)*

Po kliknutí na tlačítko plus je zobrazen seznam kontrol, které SoapUI umožňuje. Mezi nejčastěji používané patří:

- **XPath Match**
- **Contains**

Následující ukázky budou vycházet z odpovědi služby Geo Services uvedené ve výpisu 13.

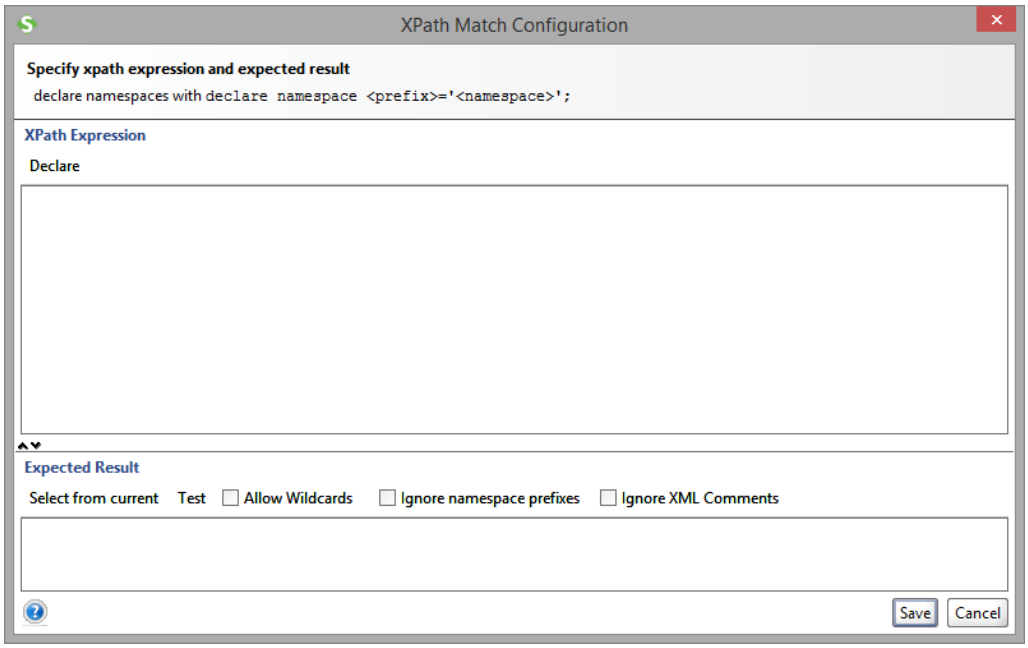
Výpis 13: SOAP odpověď služby Geo Services

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Body>
      <Location xmlns="http://geoservice/"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <Type>PostalCode</Type>
         <Name>10065</Name>
         <City>New York</City>
         <Region>NY</Region>
         <PostalCode>10065</PostalCode>
         <Lat>41</Lat>
         <Lon>-74</Lon>
      </Location>
   </s:Body>
</s:Envelope>
```

XPath Match

Tento typ kontroly slouží k porovnání výsledku řetězce XPath s předdefinovanou hodnotou. XPath lze kombinovat s regulárním výrazem nebo například s proměnnými definovanými v rámci testu SoapUI.

V části *XPath Expression* se definuje řetězec XPath a v části *Expected Result* se definuje očekávaná hodnota. Okno pro definici XPath je na obrázku 36.



Obrázek 36: *Tvorba kontroly typu XPath Match v SoapUI (Zdroj: autor)*

Kontrola obsahu elementu

Pro porovnání hodnoty elementu je nutné nejdříve deklarovat řetězec XPath. Řetězec XPath v odpovědi nalezne element a jeho hodnotu porovná s hodnotou definovanou v očekávaném výsledku.

Příklad deklarace XPath řetězce, který nalezne hodnotu v elementu *City*.

Výpis 14: Deklarace XPath řetězce pro nalezení elementu City

```
declare namespace gs='http://geoservice/';
//gs:Location/gs:City
```

První řádek deklarace definuje jmenný prostor. Tento jmenný prostor má prefix *gs*. V řetězci XPath se dále pracuje již jen s tímto prefixem. Ve druhém řádku je řetězec XPath, který hledá element *City*, jenž je potomkem elementu *Location*.

Po deklaraci XPath řetězce následuje stanovení očekávaného výsledku. Hodnotu lze do pole ručně vepsat nebo stisknutím možnosti *Select from current* automaticky dotáhnout z aktuální odpovědi webové služby.

Výpis 15: Deklarace očekávaného výsledku

```
New York
```

Možností *Test* lze pak ověřit právě definovanou kontrolu.

Kontrola pomocí zástupných znaků

SoapUI umožňuje v kontrole obsahu elementů využívat zástupné znaky (Wildcards). Zástupným znakem je v SoapUI znak hvězdičky (*). Hvězdička znamená, že obsah elementu může být libovolný (například i prázdný). Při používání zástupných znaků je třeba v SoapUI zaškrtnout volbu *Allow Wildcards*.

Příklad níže kontroluje vracené elementy službou Geo Services, ale již ne jejich obsah.

Výpis 16: Deklarace XPath řetězce pro získání struktury odpovědi

```
declare namespace gs='http://geoservice/';
//gs:Location
```

Výpis 17: Deklarace očekávaného výsledku

```
<Location xmlns="http://geoservice/" xmlns:i="http://www.w3.org/2001/XMLSchema-
instance" xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <Type>*</Type>
  <Name>*</Name>
  <City>*</City>
  <Region>*</Region>
  <PostalCode>*</PostalCode>
  <Lat>*</Lat>
  <Lon>*</Lon>
</Location>
```

Kontrola obsahu elementu matematickými operátory

Hodnotu elementu lze také porovnávat operátory větší, menší nebo rovno. V tomto případě se do deklarace řetězce XPath zadává podmínka. Do očekávaného výsledku se zadává *true* nebo *false* v závislosti nad tím, zda očekáváme splnění nebo nesplnění definované podmínky.

Níže je uveden příklad kontroly hodnoty elementu *Lat.* Ověřuje se, zda je hodnota pole větší než 0.

Výpis 18: Deklarace XPath řetězce pro ověření velikosti hodnoty *Lat*

```
declare namespace gs='http://geoservice/';
//gs:Location/gs:Lat/text() > 0
```

Výpis 19: Deklarace očekávaného výsledku

```
true
```

Kontrola existence elementu

V řetězci XPath lze použít klíčové slovo *exists* pro ověření, zda určitý element v odpovědi existuje.

Příklad níže kontroluje, zda se v odpovědi nachází element *City*.

Výpis 20: Deklarace XPath řetězce pro ověření existence elementu *City*

```
declare namespace gs='http://geoservice/';
exists( //gs:Location/gs:City)
```

Výpis 21: Deklarace očekávaného výsledku

```
true
```

Kontrola elementu pomocí regulárního výrazu

Při kontrole hodnoty elementu lze využít i regulárních výrazů. Při použití klíčového slova *Matches* ověřujeme, zda hodnota pole je rovna deklarované hodnotě. V deklaraci lze následně použít regulárních výrazů.

Příklad kontroly hodnoty elemente *Name,* zda obsahuje číslo přesně o pěti znacích.

Výpis 22: Deklarace XPath řetězce pro ověření elementu pomocí regulárního výrazu

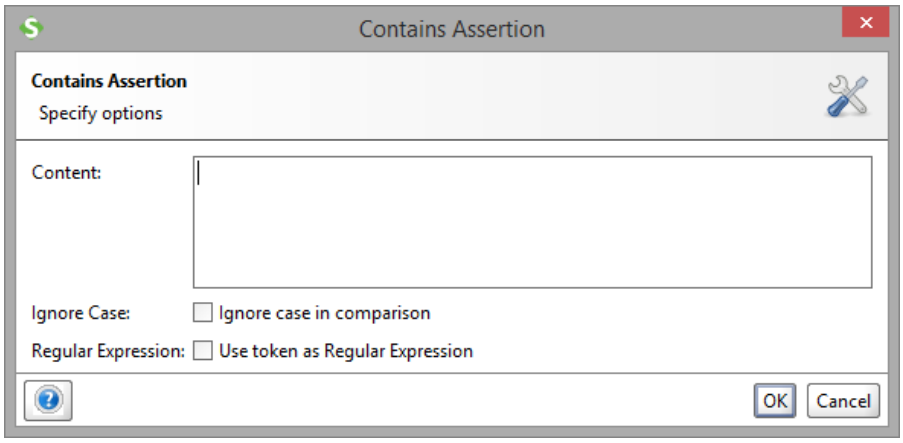
```
declare namespace gs='http://geoservice/';
matches(//gs:Location/gs:Name/text(),'^\d{5}$')
```

Výpis 23: Deklarace očekávaného výsledku

```
true
```

Contains

Kontrola odpovědi webové služby, zda obsahuje daný text. Při kontrole lze využít regulární výrazy.



Obrázek 37: *Tvorba kontroly typu Contains v SoapUI (Zdroj: autor)*

Hledání řetězce

V případě, že chceme v ukázkové odpovědi, viz výpis 13, najít řetězec „New York“, stačí do pole *Content* napsat:

Výpis 24: Deklarace *Content*

```
New York
```

Hledání řetězce na základě regulárního výrazu

V případě, že chceme opět najít řetězec „New York“, ale tentokrát pomocí regulárního výrazu, je potřeba zaškrtnout volbu *Use token as Regular Expression* a do pole *Content* napsat:

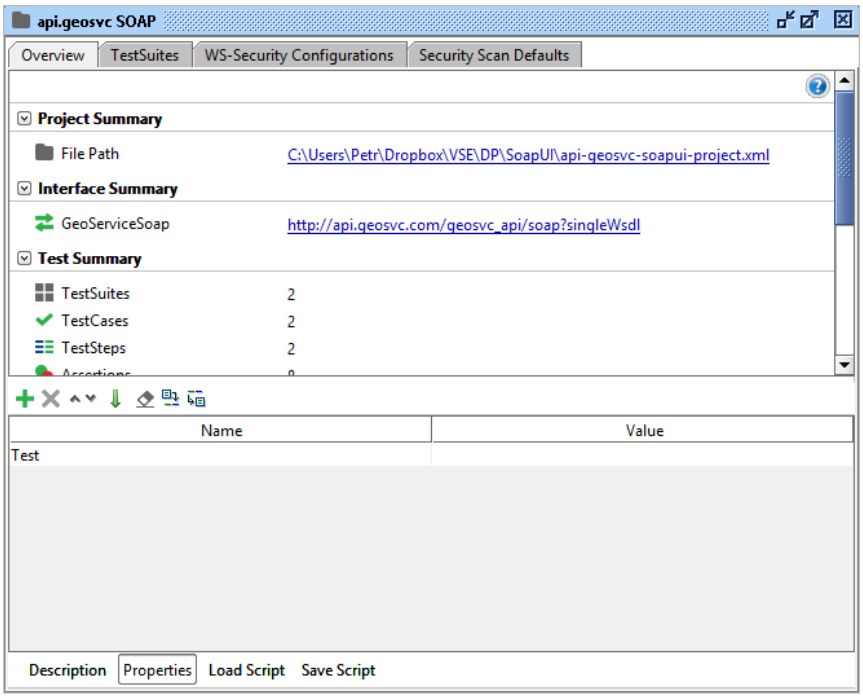
Výpis 25: Deklarace *Content* pomocí regulárního výrazu

```
(?s).*New York.*
```

A.4.3 Práce s proměnnými v SoapUI

Proměnné v SoapUI slouží k uchování hodnot a může být na ně v rámci testů odkazováno. Výhodou používání proměnných je, že díky nim je možné testy jednoduše parametrizovat. Proměnné v SoapUI lze definovat na několika úrovních:

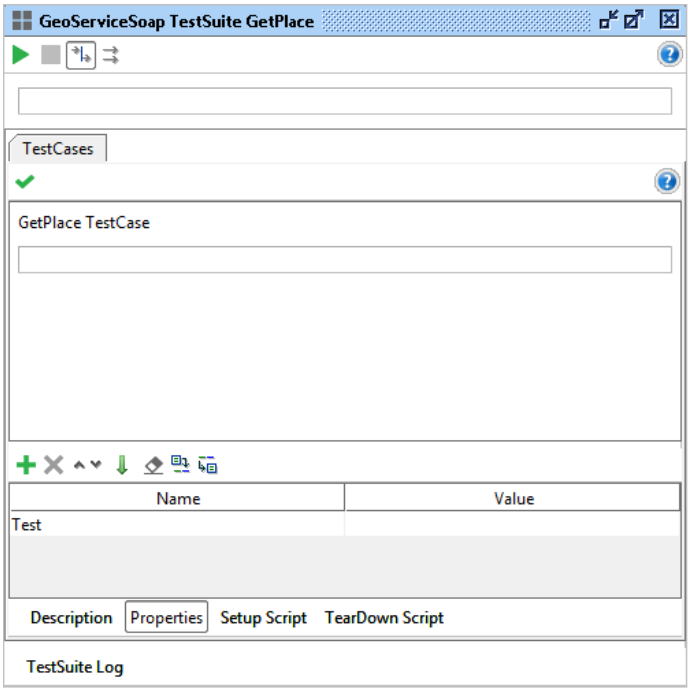
1. **Projektová proměnná** – proměnná definovaná pro celý projekt v SoapUI. Projektovou proměnnou lze vytvořit po otevření detailu projektu a kliknutí na *Properties,* viz obrázek 38.



Obrázek 38: *Definice projektové proměnné (Zdroj: autor)*

Zápis pro získání projektové proměnné s názvem *Test*: `${#Project#Test}`.

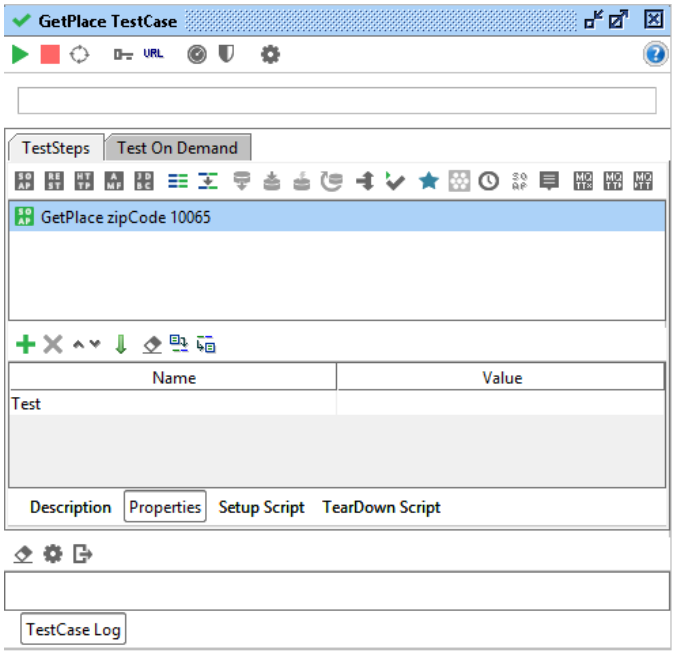
2. **Proměnná v TestSuit** – proměnná definovaná pro danou testovací sadu. Proměnnou lze vytvořit po otevření detailu testovací sady a kliknutí na *Properties,* viz obrázek 39.



Obrázek 39: *Definice proměnné v testovací sadě (TestSuite)(Zdroj: autor)*

Zápis pro získání proměnné z testovací sady s názvem *Test*: `${#TestSuite#Test}`.

3. **Proměnná v TestCase** – proměnná definovaná pro daný testovací případ. Proměnnou lze vytvořit po otevření detailu testovacího případu a kliknutí na *Properties,* viz obrázek 40**.**

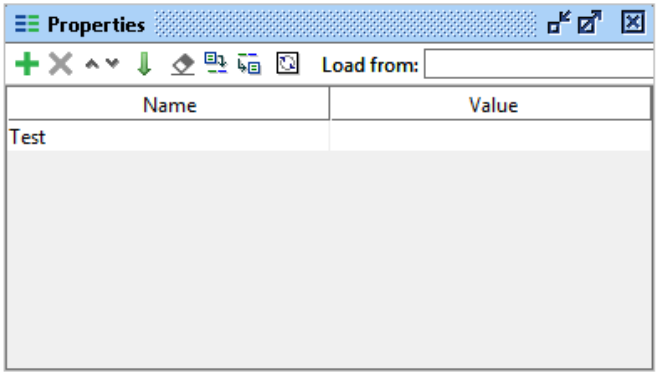


Obrázek 40: *Definice proměnné v testovacím případu (TestCase)(Zdroj: autor)*

Zápis pro získání proměnné z testovacího případu *Test*: `${#TestCase#Test}`.

4. **Proměnná jako TestStep** – proměnné lze definovat i na úrovni testovacího kroku. Testovací krok s proměnnými se nazývá *Properties*. V rámci této struktury je možné také uchovávat jednotlivé proměnné.

   Properties lze založit kliknutím pravým tlačítkem myši na *Test Steps* a vybráním volby *Add Steps* a *Properties*.



Obrázek 41: *Definice proměnné v konstrukci „Properties“ (Zdroj: autor)*

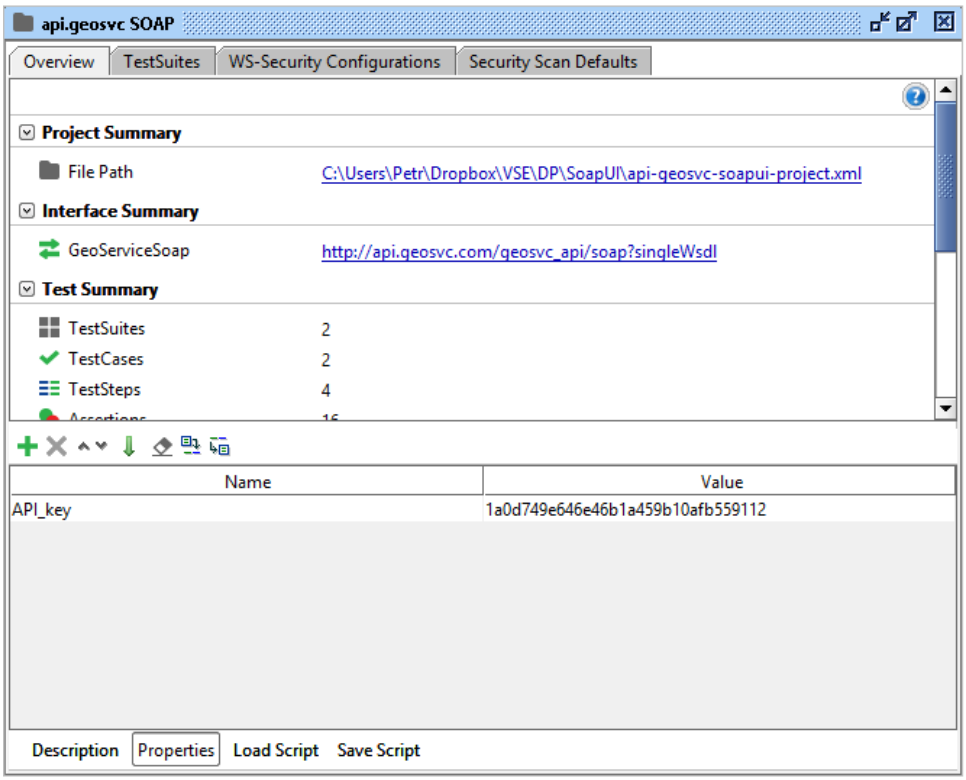
Zápis pro získání proměnné z testovacího kroku *Properties* s názvem *Test*: `${Properties#Test}`.

Použití proměnné v dotazu

Místo konkrétní hodnoty elementu se do elementu vloží název proměnné s reflektováním toho, v jaké úrovni je proměnná definována.

Příklad níže zobrazuje použití projektové proměnné *API_key*, která obsahuje klíč pro ověření identity. Proměnná je v tomto příkladu deklarována již na projektové úrovni, protože se jedná o hodnotu, která je stejná pro všechny dotazy na webovou službu Geo Services.

Deklaraci proměnné zachycuje obrázek 42.



Obrázek 42: *Deklarace proměnné API_key na projektové úrovni (Zdroj: autor)*

Konkrétní použití proměnné *API_key* zachycuje dotaz níže. Proměnná je vložena mezi elementy `<geos:ApiKey>` a `</geos:ApiKey>` na místo konkrétního API klíče *1a0d749e646e46b1a459b10afb559112*.

Výpis 26: Použití proměnné *API_key* v dotazu na službu Geo Services

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
   <soapenv:Header>
      <geos:ApiKey>${#Project#API_key}</geos:ApiKey>
   </soapenv:Header>
```

```
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>10065</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```

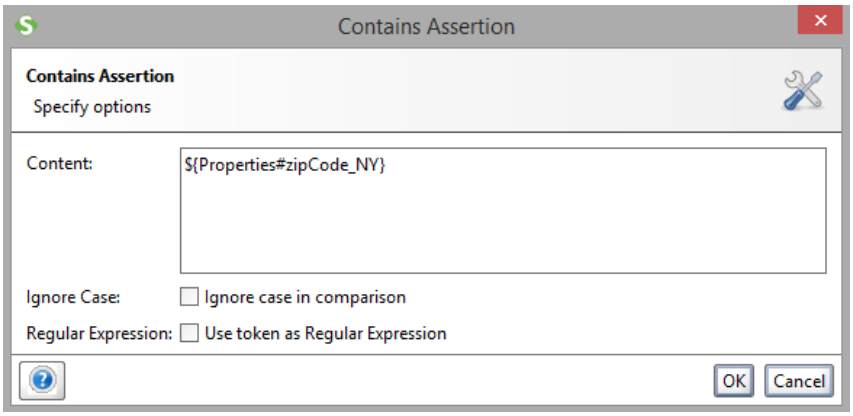
Použití proměnné při kontrole

S proměnnou lze pracovat i v případě kontrol. Místo konkrétní očekávané hodnoty se použije název proměnné s reflektováním toho, v jaké úrovni je proměnná definována. V případě kontroly pomocí *Contains Assertion* je však potřeba použít proměnou definovanou na úrovni testovacího kroku (Properties), jinak může dojít k chybnému vyhodnocení kontroly.

Příklad níže znázorňuje použití proměnné *zipCode_NY* v dotazu a i v následné kontrole. Kontrola tedy ověřuje, že PSČ, které bylo voláno, je zároveň vráceno v odpovědi webové služby. V případě, že se PSČ změní, není potřeba měnit dotaz ani kontrolu, ale pouze hodnotu proměnné.

Výpis 27: Použití proměnné *zipCode_NY* v dotazu na službu Geo Services

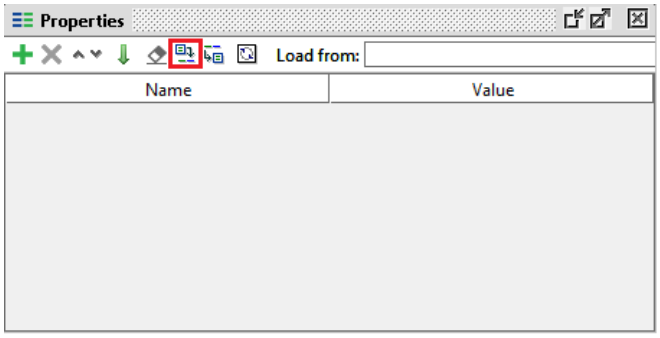
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
   <soapenv:Header>
      <geos:ApiKey>1a0d749e646e46b1a459b10afb559112</geos:ApiKey>
   </soapenv:Header>
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>${Properties#zipCode_NY}</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```



Obrázek 43: *Použití proměnné zipCode_NY v kontrole typu Contains (Zdroj: autor)*

Načtení proměnné ze souboru

Proměnné je možné načítat a ukládat z externího souboru. Pro načtení proměnných ze souboru stačí kliknout na tlačítko v okně deklarace proměnných, viz obrázek 44.



Obrázek 44: *Načtení proměnné ze souboru (Zdroj: autor)*

Soubor, ze kterého jsou proměnné načítány, by měl obsahovat seznam proměnných a jejich hodnot ve formátu *nazev_promanne=hodnota_promenne* oddělené od sebe novým řádkem.

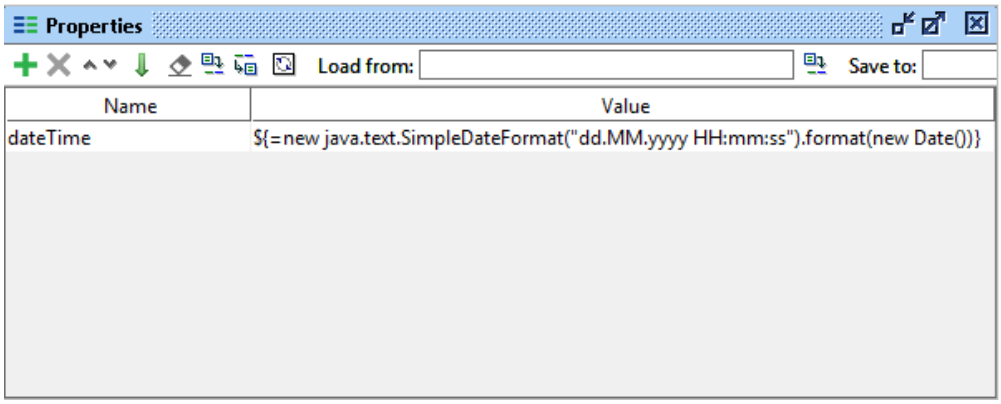
Výpis 28: Příklad hodnot souboru s dvěma proměnnými: *API_key* a *zipCode_NY*

```
API_key=1a0d749e646e46b1a459b10afb559112
zipCode_NY=10065
```

Dynamické generování hodnoty proměnné

Hodnota proměnné nemusí být statická, ale může být generována dynamicky.

Příklad níže ukazuje vygenerování aktuálního datum s časem pomocí Java funkce *Date* a provede naformátování pomocí funkce *SimpleDateFormat* na *dd.MM.yyyy HH:mm:ss*.



Obrázek 45: *Deklarace dynamické proměnné (Zdroj: autor)*

Výpis 29: Příklad vygenerování data a času ve formátu *dd.MM.yyyy HH:mm:ss*

```
${=new java.text.SimpleDateFormat("dd.MM.yyyy HH:mm:ss").format(new Date())}
```

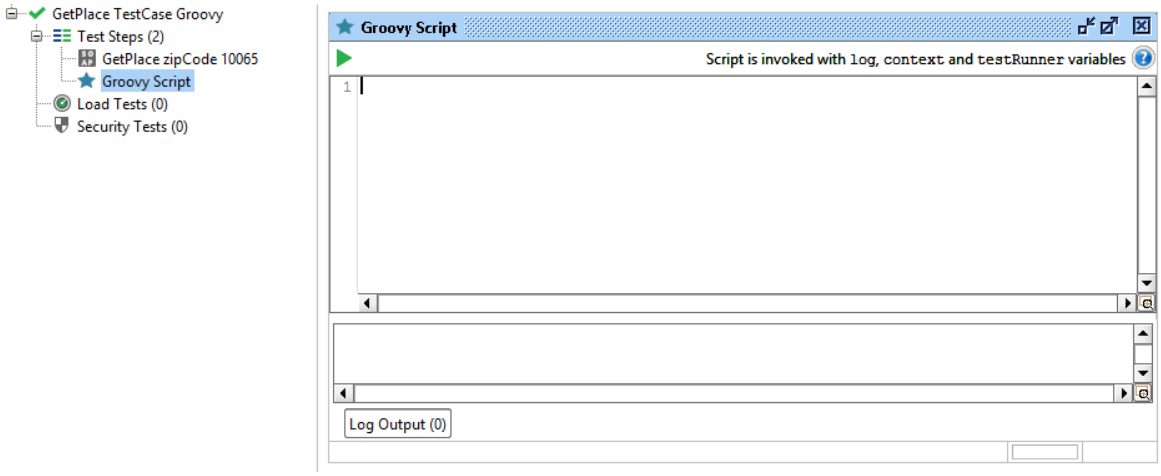
Příklad níže znázorňuje vygenerování náhodného pětimístného čísla pomocí matematické funkce *Random*.

Výpis 30: Příklad vygenerování náhodného pětimístného čísla

```
${=(int)(Math.random()*100000)}
```

A.4.4 Tvorba skriptu v Groovy Script

Groovy Script se v SoapUI vytváří jako nový krok v testovacím případu, viz obrázek 46.



Obrázek 46: *Nový Groovy Script (Zdroj: autor)*

Po založení Groovy Scriptu je již zobrazen editor, ve kterém je možné psát kód a tlačítkem  jej spustit. Vzhledem k tomu, že se jedná o skriptovací jazyk, je kód prováděn okamžitě a není tak třeba čekat na jeho zkompilování.

Načtení proměnné

V Groovy skriptu lze pracovat s proměnnými definovanými v SoapUI.

Výpis 31: Příklad načtení projektové proměnné *API_key*

```
def API_key = context.expand('${#Project#API_key}')
```

Uložení hodnoty do proměnné

Výpis 32: Příklad uložení hodnoty *10065* do proměnné *zipCode_NY* definované v úrovni testovacího případu

```
context.testCase.setPropertyValue( "zipCode_NY","10065" )
```

Získání hodnoty z odpovědi

V rámci Groovy skriptu lze pracovat i s dotazy a odpověďmi webových služeb. S takto získanými hodnotami lze dále ve skriptu pracovat a vytvářet tak například kontroly, které rozhraní SoapUI neumí.

Výpis 33: Příklad načtení hodnoty *City* z odpovědi webové služby

```
def response = context.expand('${GetPlace zipCode 10065#Response#declare
namespace gs=\'http://geoservice/\'; //gs:Location/gs:City}')
```

Kontrola hodnoty elementu

Pomocí Groovy skriptu lze provádět vlastní kontroly.

Příklad níže kontroluje, zda hodnota *City* z odpovědi webové služby obsahuje hodnotu *New York*.

Výpis 34: Kontrola, zda se hodnota elementu *City* rovná *New York*

```
def response = context.expand('${GetPlace zipCode 10065#Response#declare
namespace gs=\'http://geoservice/\'; //gs:Location/gs:City}')
assert response == 'New York';
```

Konverze String na Integer

Pro konverzi řetězce na číslo lze použít metodu *toInteger()*.

Tento postup najde uplatnění v případě, kdy potřebujeme provádět například matematické operace s hodnotami, které jsou vráceny z odpovědi služby.

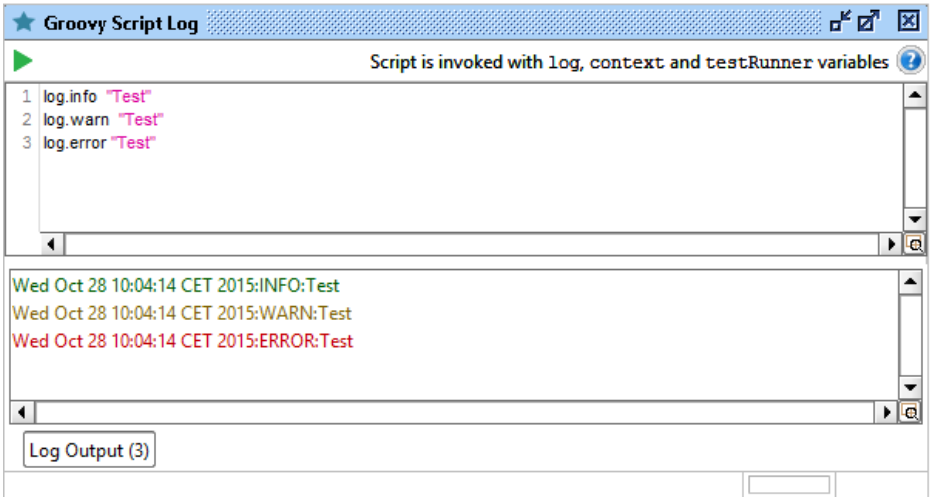
Výpis 35: Konverze proměnných *lat* a *lon* na Integer a provedení matematické operace odečtení

```
def lat = context.expand('${GetPlace zipCode 10065#Response#declare namespace
gs=\'http://geoservice/\'; //gs:Location/gs:Lat}').toInteger()
```

```
def lon = context.expand('${GetPlace zipCode 10065#Response#declare namespace
gs=\'http://geoservice/\'; //gs:Location/gs:Lon}').toInteger()
return lat - lon;
```

Logování

V Groovy skriptu lze využít také základní logování, viz obrázek 47. Logování může být využito pro zaznamenávání průchodu skriptem nebo například pro zaznamenávání výsledků prováděných kontrol.



Obrázek 47: *Logování v Groovy skriptu (Zdroj: autor)*

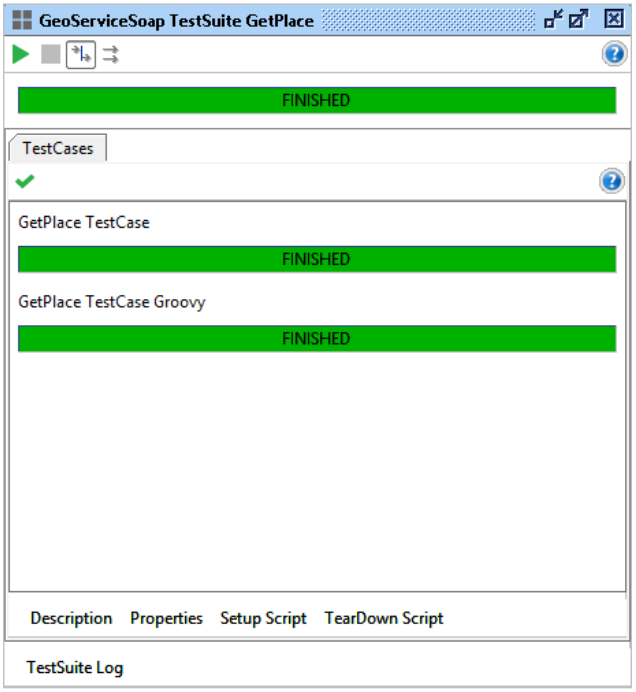
Výpis 36: Ukázka logování do tří úrovní *info*, *warn* a *error*

```
log.info  "Test"
log.warn  "Test"
log.error "Test"
```

A.4.5 Spuštění testů v SoapUI

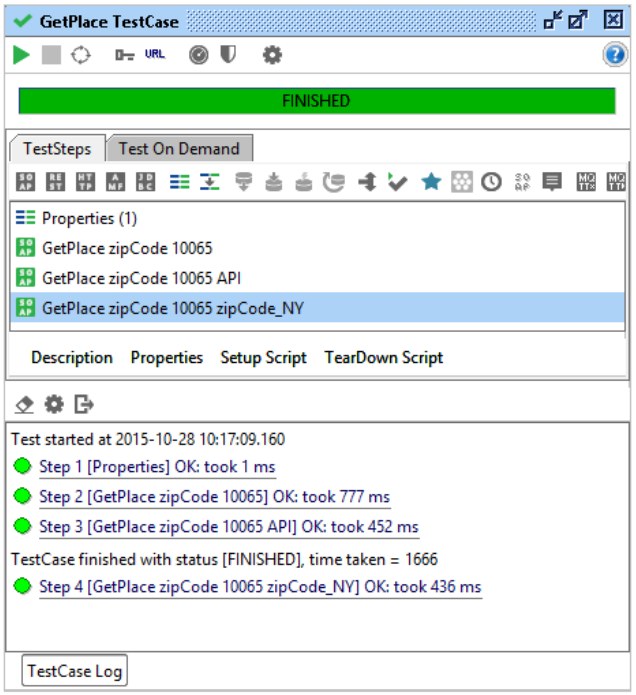
Testy v SoapUI je možné spustit tlačítkem ▶ z detailu testovací sady, viz obrázek 48 nebo z detailu testovacího případu, viz obrázek 49.

V případě spuštění testů z testovací sady dojde k provedení všech testovacích případů. Obrázek 48 zobrazuje úspěšné provedení dvou testovacích případů.



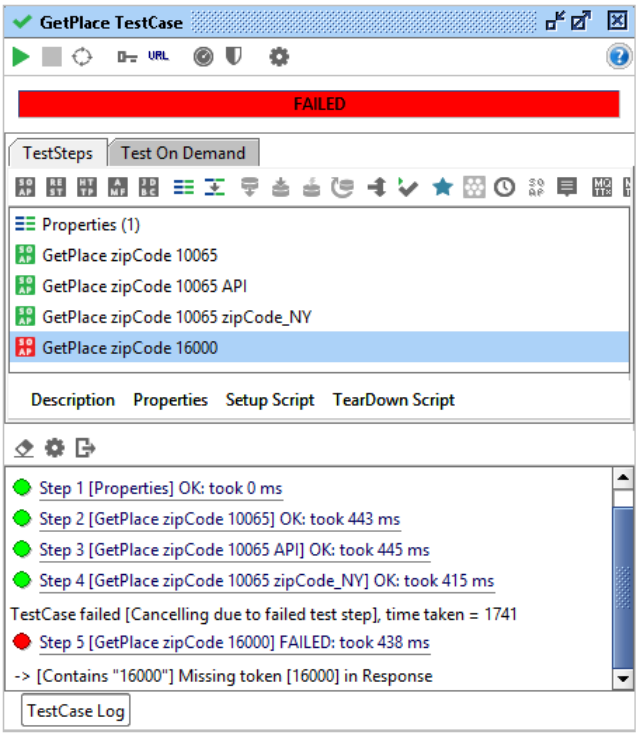
Obrázek 48: *Spuštění testů SoapUI z testovací sady (Zdroj: autor)*

V případě spuštění testů z testovacího případu dojde k provedení všech aktivních testovacích kroků definovaných v testovacím případu. Obrázek 49 zobrazuje úspěšné provedení 4 kroků testovacího případu s informací o tom, jak dlouho trvalo provedení celého testovacího případu, ale i jednotlivých kroků.



Obrázek 49: *Spuštění testů SoapUI v testovacím případu (Zdroj: autor)*

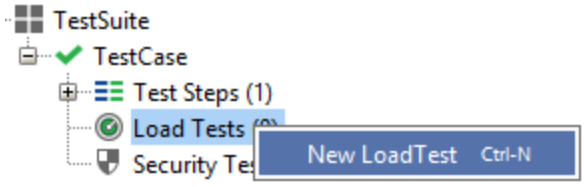
V případě neúspěšné kontroly v některém z kroků se ve výsledku testu zobrazí *FAILED,* viz obrázek 50**,** společně s důvodem, proč daný testovací krok neprošel kontrolou.



Obrázek 50: *Spuštění testů SoapUI v testovacím případu s jedním neúspěšným testem (Zdroj: autor)*

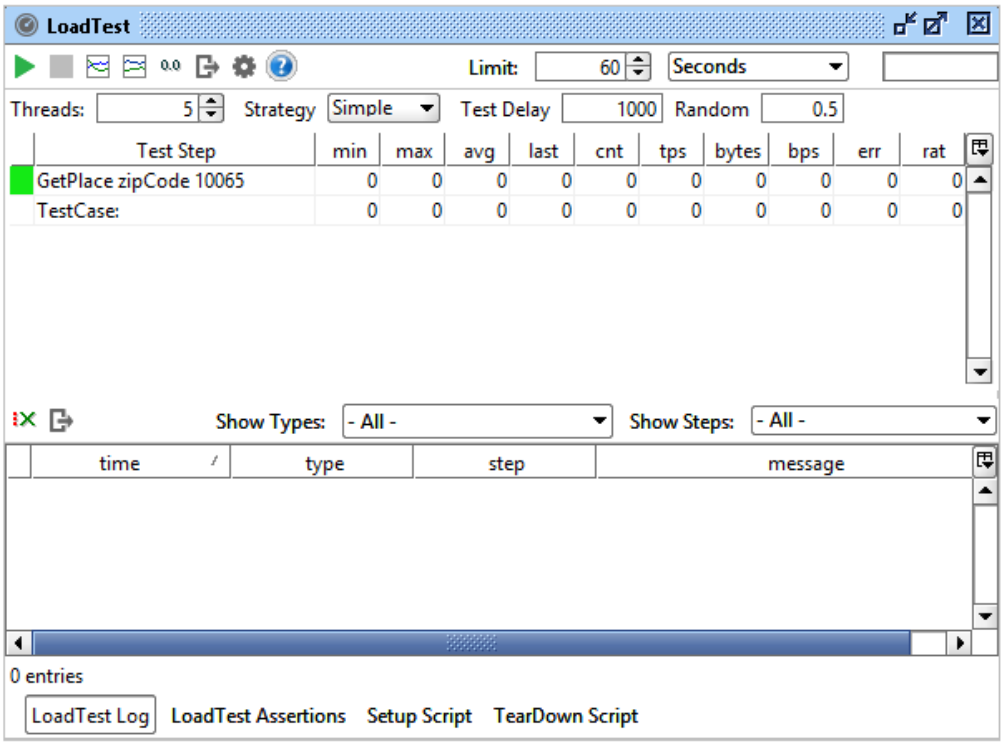
A.4.6 Vytvoření zátěžového testu

SoapUI umožňuje vytvářet zátěžové testy z již vytvořených funkčních testů. Zátěžové testy se v SoapUI vytváří na úrovni testovacího případu. Pro tvorbu testu je potřeba mít vytvořenou testovací sadu a testovací případ, který obsahuje alespoň jeden dotaz na testovanou webovou službu. Nový zátěžový test se vytvoří volbou *New LoadTest* nebo stisknutím klávesové zkratky *CTRL + N,* viz obrázek 51.



Obrázek 51: *Vytvoření bezpečnostního testu v SoapUI (Zdroj: autor)*

Po založení nového zátěžového testu je zobrazeno okno, které umožňuje vytvářet, spouštět a spravovat zátěžové testy pro jednotlivé dotazy na webovou službu, viz obrázek 52.



Obrázek 52: *Okno zátěžového testu v SoapUI (Zdroj: autor)*

SoapUI automaticky vygeneruje zátěžový test, který je možné upravit parametry v horní části okna.

Přednastaven je limit běhu zátěžového testu na 60 vteřin. Místo vteřin je možné použít počet opakování nebo počet běhů na jedno vlákno. V případě zvolení počtu opakování bude každý krok opakován právě *N*krát.

V další části okna je možná nastavit počet vláken (Threads). Vlákno zde reprezentuje jednoho virtuálního uživatele, který vykonává jednotlivé kroky zátěžového testu. Počet vláken 5 tedy znamená, že je simulována situace, kdy pět uživatelů (každý sám za sebe) provádí definovaný set kroků.

Další volba umožňuje nastavit strategii zátěžového testu. Na výběr jsou následující volby:

- **Burst –** jedná se o strategii, kdy je ve stanovených intervalech generována krátkodobá vysoká zátěž. Interval se definuje v poli *Burst Delay* a délka generování zátěže se definuje v poli *Burst Duration*. Pro tento typ testů je vhodné nastavit vyšší počet vláken a kratší dobu zátěže.

  V případě následující ukázky bude po dobu pěti minut každých 60 vteřin generována zátěž sta uživatelů po dobu deseti vteřin.

Výpis 37: Ukázka nastavení strategie *Burst*

```
Limit: 300 Seconds
```

```
Threads: 100
Strategy: Burst
Burst Delay: 60
Burst duration: 10
```

- **Simple –** jedná se o přednastavenou strategii v SoapUI při vytvoření zátěžového testu. V rámci této strategie dochází k provedení testu s definovaným zpožděním mezi každou iterací. Zpoždění se definuje v poli *Test Delay* (ms). Zpoždění lze také pomocí pole *Random* upravit tak, aby zpoždění nebylo konstantní, ale obsahovalo určitou míru nahodilosti.

  V případě ukázky níže bude po dobu 1 minuty test prováděn pěti virtuálními uživateli s jednovteřinovým zpožděním mezi každým provedením testu.

Výpis 38: Ukázka nastavení strategie *Simple*

```
Limit: 60 Seconds
Threads: 5
Strategy: Simple
Test Delay: 1000
Random: 0
```

- **Thread –** simuluje narůstající zátěž na webovou službu. V případě této strategie se definuje v poli *Start Threads* počáteční počet virtuálních uživatelů a v *End Threads* konečný počet virtuálních uživatelů. Zátěž narůstá pozvolna (lineárně).

  V ukázce níže je definováno, že po dobu 1 minuty narůstat počet virtuálních uživatelů, kteří provádějí test, od jednoho uživatele až po 60 uživatelů. Nárůst je jeden uživatel každou jednu vteřinu.

Výpis 39: Ukázka nastavení strategie *Simple*

```
Limit: 60 Seconds
Threads: 60
Strategy: Threads
Start Threads: 1
End Threads: 60
```

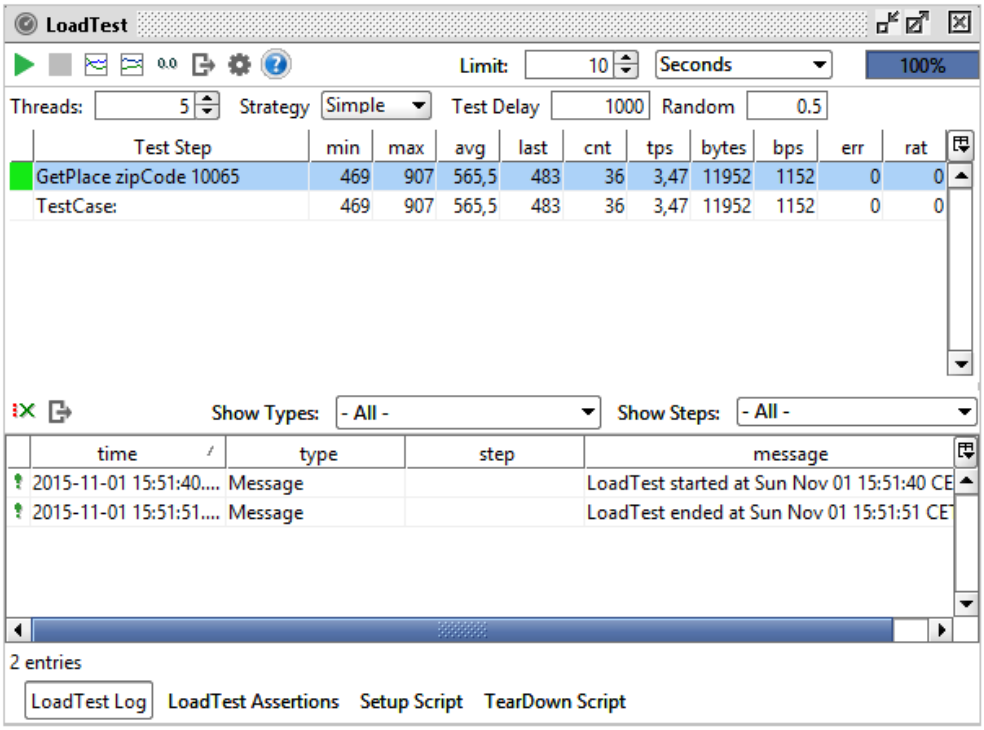
- **Variance –** cílem této strategie je simulovat zátěž, která v průběhu testu kolísá. V poli *Interval* se definuje interval, po kterém dojde ke změně počtu virtuálních uživatelů. V poli *Variance* se definuje, k jak velké změně v počtu virtuálních uživatelů dojde.

  Ukázka níže uvádí test, který bude probíhat 5 minut a každou minutu dojde ke snížení nebo navýšení počtu virtuálních uživatelů o 5.

Výpis 40: Ukázka nastavení strategie *Simple*

```
Limit: 300 Seconds
Threads: 10
Strategy: Threads
Interval: 60
Variance: 0.5
```

Po nadefinování testu je možné zátěžový test spustit tlačítkem . Po jeho spuštění je okamžitě vidět průběh testu se sledovanými parametry, viz obrázek 53.



Obrázek 53: *Spuštění zátěžového testu v SoapUI (Zdroj: autor)*

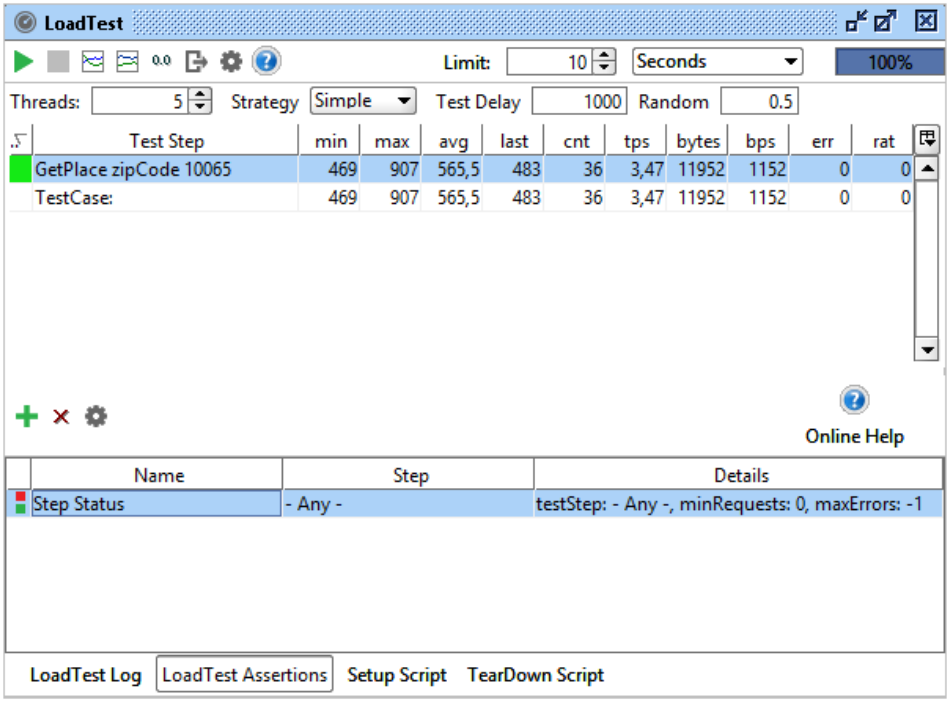
Mezi sledované parametry zátěžového testu patří (Kankanamge, 2012):

- **min** – minimální naměřená doba odpovědi testovacího kroku (v ms),
- **max** – maximální naměřená doba odpovědi testovacího kroku (v ms),
- **avg** – průměrná naměřená doba odpovědi testovacího kroku (v ms),
- **last** – průměrná naměřená doba odpovědi posledního běhu (v ms),
- **cnt** – kolikrát byl daný testovací krok proveden,
- **tps** – počet transakcí za vteřinu,
- **bytes** – kolik bytů bylo přesunuto,
- **bps** – počet bytů za vteřinu,

- **err** – počet chyb, které se běh testu objevily,
- **rat** – procento neúspěšných dotazů.

Přidání kontrol

V zátěžových testech lze také nadefinovat kontroly. Novou kontrolu lze přidat na záložce *LoadTest Assertions* kliknutím na tlačítko ✚, viz obrázek 54.



Obrázek 54: *Vytvoření kontroly v zátěžovém testu (Zdroj: autor)*

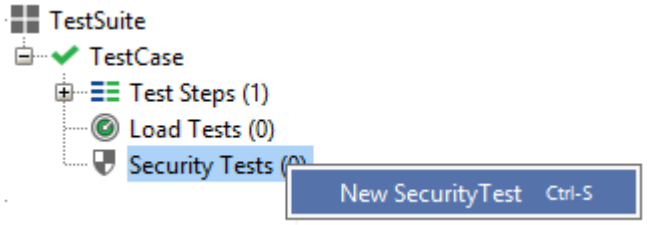
Kontroly lze definovat jak pro jednotlivé testovací kroky zvlášť, tak pro všechny dohromady. Mezi kontroly, které jsou v dostupné SoapUI, patří:

- **Max Errors** – kontroluje maximální počet chyb testovacího kroku.
- **Step Average** – kontroluje průměrnou dobu testovacího kroku.
- **Step TPS** – kontroluje počet transakcí za vteřinu testovacího kroku.
- **Step Maximum** – kontroluje maximální dobu testovacího kroku.
- **Step Status** – přednastavená kontrola, která ověřuje stav testovacího kroku.

A.4.7 Vytvoření bezpečnostního testu

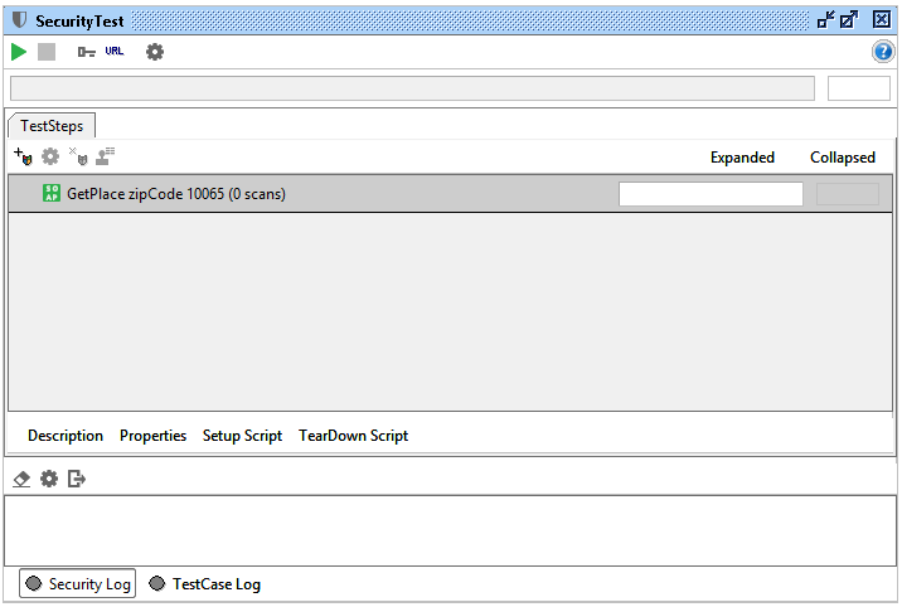
Vedle funkčních a zátěžových testů umožňuje SoapUI provádět i testy zaměřené na bezpečnost webových služeb. SoapUI umožňuje provádět sadu předpřipravených bezpečnostních kontrol, za účelem odhalení potencionálních slabých míst.

Bezpečnostní testy se v SoapUI stejně jako zátěžové testy vytvářej na úrovni testovacího případu. Pro tvorbu testu je třeba mít vytvořenou testovací sadu a testovací případ, který obsahuje alespoň jeden dotaz na testovanou webovou službu. Nový bezpečnostní test se vytvoří volbou *New SecurityTest* nebo stisknutím klávesové zkratky *CTRL + S,* viz obrázek 55.



Obrázek 55: *Vytvoření bezpečnostního testu v SoapUI (Zdroj: autor)*

Po založení nového bezpečnostního testu je zobrazeno okno, které umožňuje vytvářet, spouštět a spravovat bezpečnostní testy pro jednotlivé dotazy na webovou službu, viz obrázek 56.



Obrázek 56: *Okno bezpečnostního testu v SoapUI (Zdroj: autor)*

Tlačítkem  je možné přidat nový test pro dotaz na webovou službu. Typy bezpečnostních testů, které lze v SoapUI vytvořit, jsou:

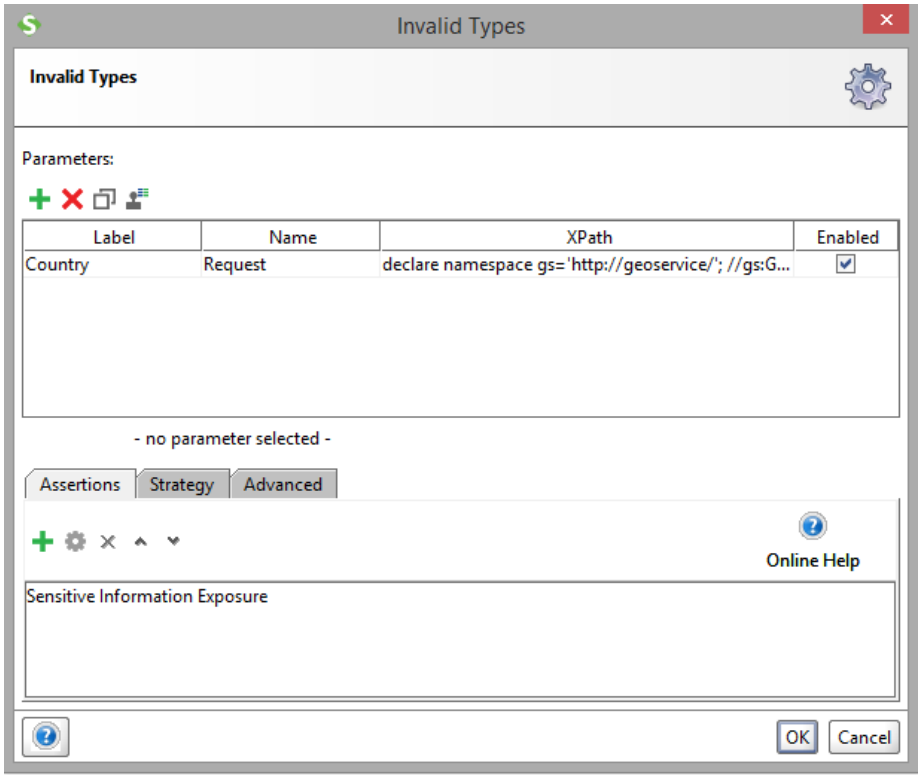
- **SQL Injection** – testy zaměřené na zranitelnost integrace s databází,
- **XPath Injection** – testy zaměřené na chybné zpracování XML webovou službou,
- **Boundary Scan** – testy zaměřené na zasílání hodnot, které jsou mimo definovaný rozsah,
- **Invalid Types –** testy zaměřené na zasílání nevalidních hodnot,

- **Malformed XML** – testy zaměřené na chování serveru nebo webové služby při zasílání. nevalidních XML,
- **XML Bomb** – testy zaměřené na zasílání škodlivých XML dotazů,
- **Malicious Attachment** – testy zaměřené na zasílání škodlivých příloh,
- **Cross Site Scripting** – testy zaměřené na nalezení zranitelností v kódu webové služby,
- **Custom Script –** umožňuje vytvoření vlastních testů.

Vzhledem k rozsahu práce není možné detailně popsat tvorbu jednotlivých bezpečnostních testů. Pro účely této práce bude popsána tvorba testu na zasílání nevalidních hodnot (Invalid Types), na tomto typu bude popsán princip tvorby bezpečnostních testů.

Po zvolení typu testu je potřeba nadefinovat parametry, kontroly a strategii testu. V případě testu *Invalid Types* je třeba nadefinovat, v jakých elementech dotazu se budou zasílat definované nevalidní hodnoty.

Tlačítkem ✚ se přidá nový parametr, pro který se definuje označení (Label), název (Name) a XPath cesta pro identifikaci elementu. Obrázek 57 zobrazuje parametr, který nahrazuje hodnoty v elementu *Country*.



Obrázek 57: *Okno tvorby testu Invalid Types (Zdroj: autor)*

V dalším kroku je třeba vytvořit kontrolu. Ta se vytváří stisknutím tlačítka v sekci *Assertions*. Zde je potřeba vybrat typ kontroly. V tomto případě je typ kontroly *Security* a *Sensitive Information Exposure*.

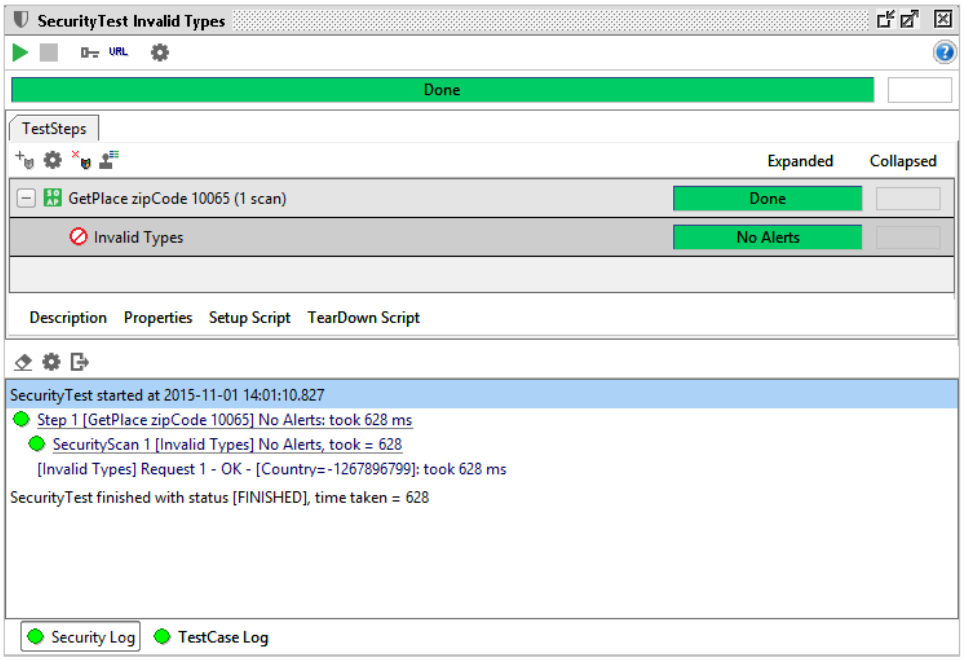
V rámci tohoto typu kontroly je nezvytné zadat řetězec, který se bude v odpovědi webové služby hledat. Ve výpisu níže je příklad možného kontrolního řetězce, který je zapsán pomocí regulárního výrazu. Tento řetězec kontroluje, zda se v odpovědi nezobrazuje výpis *Stacktrace*, pomocí kterého by útočník mohl zjistit citlivé informace o webové službě.

Výpis 41: Příklad kontroly *Sensitive Information Exposure*

```
~(?s).*(s|S)tacktrace:.*
```

Po vytvoření kontroly je možné nadefinovat strategie v záložce *Strategy*. Zde lze nastavit, jakým způsobem se mají dotazy zasílat (postupně, najednou, s odezvou apod.). V posledním kroku je potřeba nadefinovat nevalidní hodnoty, které se budou zasílat. SoapUI v tomto případě již předdefinované nevalidní hodnoty obsahuje. Tyto hodnoty je možné upravit nebo přidat vlastní.

Po nadefinování testu lze sadu bezpečnostních testů spustit tlačítkem , viz obrázek 58. Ve výsledcích testů je možné zjistit průběh testu a jednotlivých kontrol.

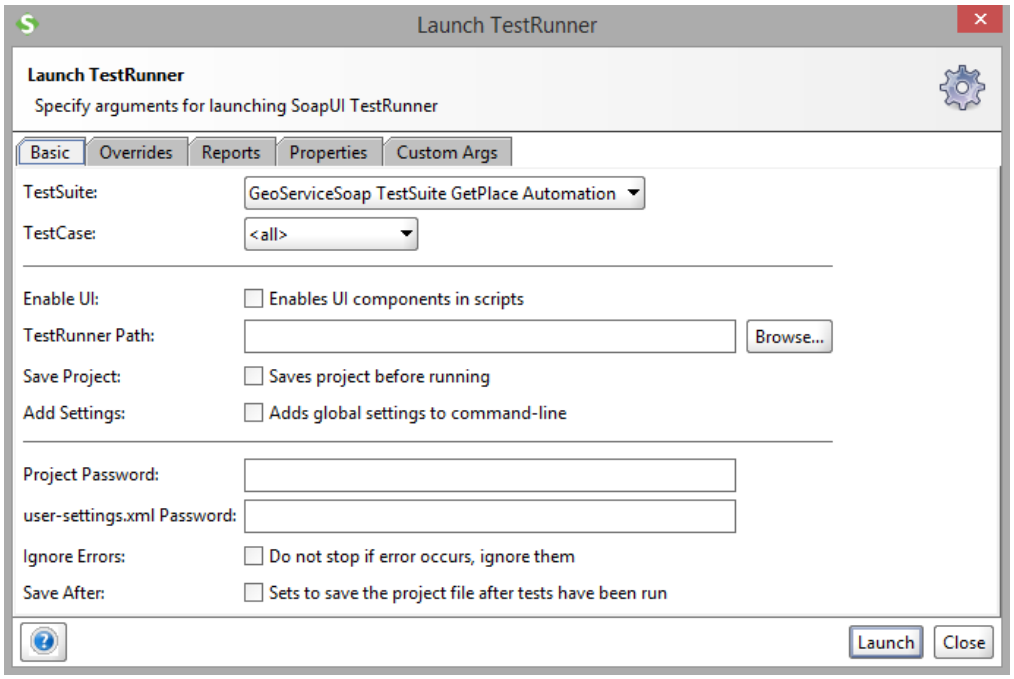


Obrázek 58: *Spuštění bezpečnostních testů v SoapUI (Zdroj: autor)*

A.5 Možnosti automatizace testování v SoapUI

V kapitole A.4.5 byly popsány možnosti manuálního spouštění testů pomocí grafického rozhraní v SoapUI. SoapUI nicméně nabízí i možnosti automatizovaného spouštění testů pomocí nástroje *TestRunner*.

TestRunner je možné spustit přímo z uživatelského prostředí SoapUI nebo z příkazové řádky. Pro spuštění TestRunner z GUI je třeba zvolit v hlavní nabídce *Project* a následně vybrat *Launch TestRunner*. Po vybrání této možnosti se zobrazí okno, které je na obrázku 59**.** V něm je možné navolit, které testy se mají spouštět, určit endpoint, nastavit, kam ukládat výsledky, proměnné a další volitelné parametry.



Obrázek 59: *Grafické rozhraní TestRunner (Zdroj: autor)*

Častěji je však vyžadováno spouštět TestRunner pomocí příkazu z příkazové řádky. V adresáři se SoapUI je adresář *bin*, ve kterém lze nalézt soubory potřebné pro spuštění a běh TestRunneru. Jedná se především o *testrunner.bat*, který slouží ke spouštění testů z příkazové řádky. Níže je uvedena ukázka příkazu, který slouží ke spuštění testů v testovací sadě a k vygenerování stručného reportu.

Výpis 42: Ukázka příkazu pro spuštění testů z testovací sady

```
cmd.exe /C testrunner.bat -s"GeoServiceSoap TestSuite GetPlace Automation"
C:\SoapUI\api-geosvc-soapui-project.xml -r
```

TestRunner nabízí řadu parametrů, se kterými je možné nástroj spustit. V ukázce výše byly použity 2 parametry:

- **s** – pro deklaraci testovací sady, která má být spuštěna,

- **r** – pro vygenerování jednoduchého reportu o výsledcích testu.

  Kompletní seznam parametrů pro TestRunner lze nalézt na http://www.soapui.org/test-automation/running-functional-tests.html.

V ukázce dále je uveden zkrácený výpis takového běhu nástroje TestRunner. Z logu lze vyčíst pouštěné testy a jednotlivé kontroly včetně jejich výsledků. V závěru je vygenerován report, který shrnuje výsledky testu.

Výpis 43: Ukázka logu z běhu TestRunneru

```
SoapUI 5.2.1 TestCase Runner
10:59:58,543 INFO  [WsdlProject] Loaded project from
[file:/C:/Users/Petr/Dropbox/VSE/DP/SoapUI/api-geosvc-soapui-project.xml]
10:59:58,561 INFO  [SoapUITestCaseRunner] Running SoapUI tests in project
[api.geosvc SOAP]
10:59:58,564 INFO  [SoapUITestCaseRunner] Running TestSuite [GeoServiceSoap
TestSuite GetPlace Automation], runType = SEQUENTIAL
10:59:58,597 INFO  [SoapUITestCaseRunner] Running SoapUI testcase [GetPlace
TestCase]
10:59:58,619 INFO  [SoapUITestCaseRunner] running step [GetPlace zipCode 10065]
10:59:59,273 DEBUG [HttpClientSupport$SoapUIHttpClient] Attempt 1 to execute
request
10:59:59,274 DEBUG
[SoapUIMultiThreadedHttpConnectionManager$SoapUIDefaultClientConnection] Sending
request: POST /geosvc_api/soap HTTP/1.1
10:59:59,493 DEBUG
[SoapUIMultiThreadedHttpConnectionManager$SoapUIDefaultClientConnection]
Receiving response: HTTP/1.1 200 OK
10:59:59,507 DEBUG
[SoapUIMultiThreadedHttpConnectionManager$SoapUIDefaultClientConnection]
Connection shut down
11:00:00,256 INFO  [SoapUITestCaseRunner] Assertion [Contains "New York"] has
status VALID
11:00:00,257 INFO  [SoapUITestCaseRunner] Assertion [Contains "New York" RegExp]
has status VALID
11:00:00,257 INFO  [SoapUITestCaseRunner] Assertion [XPath Match "New York"] has
status VALID
11:00:00,257 INFO  [SoapUITestCaseRunner] Assertion [XPath Match *] has status
VALID
11:00:00,258 INFO  [SoapUITestCaseRunner] Assertion [XPath Match Matches] has
status VALID
11:00:00,258 INFO  [SoapUITestCaseRunner] Assertion [XPath Match RegExp] has
status VALID
11:00:00,258 INFO  [SoapUITestCaseRunner] Assertion [XPath Match >] has status
VALID
11:00:00,259 INFO  [SoapUITestCaseRunner] Assertion [XPath Match Exists] has
status VALID
```

```
11:00:00,261 INFO  [SoapUITestCaseRunner] Finished running SoapUI testcase
[GetPlace TestCase], time taken: 1616ms, status: FINISHED
11:00:00,262 INFO  [SoapUITestCaseRunner] TestSuite [GeoServiceSoap TestSuite
GetPlace Automation] finished with status [FINISHED] in 1682ms

SoapUI 5.2.1 TestCaseRunner Summary
----------------------------
Time Taken: 1701ms
Total TestSuites: 1
Total TestCases: 1 (0 failed)
Total TestSteps: 1
Total Request Assertions: 8
Total Failed Assertions: 0
Total Exported Results: 0
```

Příloha B: Koncepty metodiky

B.1 Webová služba

Webové služby slouží jako stavební blok pro vytváření distribuovaných aplikací, které mohou být publikovány a konzumovány prostřednictvím Internetu či intranetových sítí. Webové služby jsou postaveny na otevřených webových standardech a je pro ně typické, že na jedné straně je poskytovatel služby, který vystavuje rozhraní, a na druhé straně příjemce služby, který webovou službu konzumuje (Papazoglou, 2008).

Specifikace webových služeb se skládá celkem ze tří hlavních principů (Barry a Dick, 2013):

- **WSDL** pro popis rozhraní webové služby,
- protokolu **SOAP** pro komunikaci a
- adresáře **UDDI** pro registraci a vyhledávání webových služeb.

B.2 WSDL

WSDL (Web Services Description Language) je XML dokument, který slouží k popisu rozhraní webové služby. V rámci této specifikace by mělo být popsáno rozhraní a další informace o dané webové službě. Mezi tyto informace patří například protokol nebo adresa, kde je webová služba dosažitelná (Kankanamge, 2012; Kuba, 2006).

V dnešní době existují 2 verze WSDL, které se stále používají (Elkstein, 2010).

- **WSDL 1.1** – verze používaná především pro popis SOAP webových služeb. Pro popis REST webových služeb chybí některé HTTP metody jako PUT a DELETE.
- **WSDL 2.0 –** zlepšená podpora pro popis REST webových služeb, díky přidání podpory všech HTTP metod.

Alternativou k WSDL (především pro webové služby REST) je WADL.

B.3 WADL

WADL (Web Application Description Language) stejně jako WSDL slouží k popisu webových služeb. Jedná se o XML popis webových služeb obvykle komunikujících prostřednictvím protokolu HTTP. Z toho důvodu se s WADL lze setkat především u webových služeb REST. Nicméně WADL na rozdíl od WSDL není příliš rozšířen (Kankanamge, 2012; Breshne, Fuhrer a Pasquier, 2009).

B.4 SOAP

SOAP slouží jako obálka pro zprávy webových služeb v XML. O standardizaci se stará World Wide Web Consorcium (W3C)(Kankanamge, 2012).

SOAP obálka se skládá ze dvou částí, hlavičky a těla. Hlavička je volitelná část SOAP obálky a může nést informace jako je autentizace nebo kódování. Tělo SOAP obálky je povinné a obsahuje vlastní XML zprávu definovanou pomocí WSDL (Kankanamge, 2012).

SOAP dotaz na službu Geo Services

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:geos="http://geoservice/">
   <soapenv:Header>
      <geos:ApiKey>1a0d749e646e46b1a459b10afb559112</geos:ApiKey>
   </soapenv:Header>
   <soapenv:Body>
      <geos:GetPlace>
         <geos:Country>US</geos:Country>
         <geos:Place>10065</geos:Place>
      </geos:GetPlace>
   </soapenv:Body>
</soapenv:Envelope>
```

SOAP odpověď služby Geo Services

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Body>
      <Location xmlns="http://geoservice/"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <Type>PostalCode</Type>
         <Name>10065</Name>
         <City>New York</City>
         <Region>NY</Region>
         <PostalCode>10065</PostalCode>
         <Lat>41</Lat>
         <Lon>-74</Lon>
      </Location>
   </s:Body>
</s:Envelope>
```

B.5 UDDI

UDDI (Universal Description, Discovery and Integration) slouží pro registraci a vyhledávání služeb popsaných pomocí WSDL. Vzhledem k tomu, že registry UDDI se v řadě případů neimplementovaly, postupem času o UDDI opadl zájem a dnes se již příliš nepoužívá (Barry a Dick, 2013).

Mezi populární repositáře webových služeb patří například web ProgrammableWeb dostupný na http://www.programmableweb.com/.

B.6 REST

REST (Representational State Transfer) je alternativa k protokolu SOAP. Jedná se o architekturu postavenou na principech popisující lokalizaci a přístup ke zdrojům (Barry a Dick, 2013).

REST je na rozdíl od SOAP orientován datově. Všechny zdroje mají vlastní identifikátor URI a REST určuje, jak se k těmto datům přistupuje. Existují čtyři základní metody, které odpovídají metodám HTTP protokolu (Malý, 2009):

- **GET** – získá zdroj na základě URI (odpovídá operaci READ).
- **POST** – vytvoří nový zdroj (odpovídá operaci CREATE).
- **PUT** – aktualizuje zdroj nebo v případě, že neexistuje, zdroj vytvoří (odpovídá operaci UPDATE).
- **DELETE** – vymaže zdroj identifikovaný URI (odpovídá operaci DELETE).

REST je jednodušší a datově méně náročný než jeho protějšek SOAP. Na rozdíl od SOAP totiž není závislý na XML a odpovědi tak mohou být zaslány v mnohem lehčí formě jako je například JSON (Kankanamge, 2012).

REST dotaz na službu Geo Services pro získání odpovědi v JSON

```
http://api.geosvc.com/rest/US/10065?apikey=1a0d749e646e46b1a459b10afb559112&form
at=json
```

REST odpověď služby Geo Services v JSON (formátovaná)

```
{
   "Type": "PostalCode",
   "Name": "10065",
   "City": "New York",
   "Region": "NY",
   "PostalCode": "10065",
   "Lat": 41,
   "Lon": -74
```

```
}
```

B.7 Mock služba

Mock služba je typ zástupného objektu, který simuluje chování určitého objektu, v tomto případě webové služby. Mock služba se využívá například tehdy, nemáme-li přístup k opravdové webové službě (například ještě není vyvinuta nebo k ní nemáme přístup, protože je mimo naši síť), ale již potřebujeme testovat napojení testované webové služby na tu, na kterou nemáme zatím přístup. Mocková služba tak může simulovat chování opravdové webové služby a umožnit tak otestovat napojení na druhou službu (Kankanamge, 2012).

B.8 XPath

XPath (XML Path Language) je jazyk, který slouží k procházení XML dokumentu. Pomocí XML výrazu lze nalézt ve stromové struktuře XML dokumentu elementy a atributy (W3C, 2014).

V SoapUI se s XPath můžeme setkat například v případě tvorby kontrol elementů pomocí funkce *XPath Match*.

Návody a ukázkové XML výrazy lze najít na stránkách od Jiřího Koska:

- http://www.kosek.cz/xml/xslt/vyrazy.html.

nebo na stránkách Simple Talk:

- https://www.simple-talk.com/dotnet/.net-framework/xpath,-css,-dom-and-selenium-the-rosetta-stone/.

B.9 Regulární výrazy

Regulární výraz (anglicky regular expression, či zkráceně regexp) slouží k vyhledání části řetězce, kterou předem (úplně) neznáme nebo která může mít více podob. Používá se především v programovacích a skriptovacích jazycích (Watzke, 2008).

V případě SoapUI se s regulárními výrazy lze setkat při tvorbě XPath výrazu nebo například při tvorbě skriptu v Groovy skriptu.

Ohledně regulárních výrazu byla napsána řada publikací a článků. Přehledný popis základních konstrukcí regulárních výrazu lze nalézt například na stránkách Interval.cz:

- https://www.interval.cz/clanky/regularni-vyrazy-v-prikladech/

Na internetu také existuje řada aplikací, které umožňují otestovat regulární výraz na dané množině dat pro ověření, zda sestavený regulární výraz je správný. Mezi takové testery regulárních výrazů patří například:

- http://www.regexpal.com/.

B.10 Groovy Script

Groovy Script je skriptovací jazyk, který se nejvíce podobá jazykům jako Ruby, Pearl nebo JavaScript (Kankanamge, 2012).

V SoapUI je možné vytvořit Groovy Script jako krok v testu, pomocí kterého je možné provést například kontrolu, nastavení SoapUI proměnných, generování náhodných dat atd. Praktické příklady použití Groovy Scriptu lze nalézt v příručce k SoapUI v kapitole A.4.4.

Dokumentace k tomuto jazyku je dostupná na domovských stránkách:

- http://groovy-lang.org/.

Příloha C: Nástroje

C.1 SoapUI

Nástroj společnosti SmartBear Software pro tvorbu komplexních funkční a nefunkčních testů. SoapUI podporuje řadu technologií a standardů: SOAP a REST webové služby, JMS nebo RIA. Není to tak nástroj pouze pro testování webových služeb, jak by se na první pohled mohlo zdát. Nicméně oblast testování SOAP a REST webových služeb je pro nástroj stěžejní.

V roce 2014 SmartBear integroval nástroj SoapUI Pro do svého nového nástroje „Ready! API“, zatímco open source verze SoapUI označovaná jako „OS“ nadále pokračuje jako standalone verze.

Mezi hlavní přednosti nástroje patří:

- podpora SOAP a REST webových služeb,
- možnost tvorby funkčních a nefunkčních testů webových služeb (regresní, bezpečností a zátěžové testy) s širokou paletou aplikovatelných kontrol,
- tvorba skriptů (Groovy, JavaScript),
- možnost vytvářet mocky webových služeb,
- automatizace provádění testů.

C.1.1 Verze SoapUI

Nástroj SoapUI je poskytován ve verzi zdarma jako SoapUI OS a v placené verzi jako SoapUI NG Pro.

SoapUI OS

Verze OS (Open Source) je poskytována zdarma pod licencí EUPL, European Union Public Licence. Oproti placené verzi však obsahuje méně funkcionalit.

SoapUI OS. SoapUI OS je možné stáhnout z webových stránek SoapUI:

- http://www.soapui.org/downloads/soapui/open-source.html.

SoapUI NG Pro

Placená verze nástroje SoapUI, dříve poskytována pod názvem SoapUI Pro. Od konce roku 2014 došlo k přejmenování na SoapUI NG Pro (NG = New Generation) a k integrování do nového nástroje „Ready! API“.

Nástroj je možné zdarma vyzkoušet ve 14denní lhůtě (tzv. Trial verze). SoapUI NG Pro je možné stáhnout z webových stránek společnosti SmartBear:

- http://smartbear.com/lp/soapui/download-soapui-ng-pro/.

Ready! API

Nový nástroj firmy Smart Bear. První verze Ready! API byla publikována 15.10.2014. Ready! API v sobě integruje v několik nástrojů pro komplexní testování API. Jedná se o nástroje:

- SoapUI NG Pro,
- LoadUI NG Pro,
- Secure Pro,
- ServiceV Pro.

Ready! API je možné vyzkoušet ve 14denní lhůtě. Nástroj je možné stáhnout ze stránek SmartBear:

- http://smartbear.com/product/ready-api/free-trial/.

C.1.2 Rozdíly mezi SoapUI OS a SoapUI NG Pro

Placená verze SoapUI obsahuje oproti verzi zdarma několik funkcionalit navíc. Jedná se obvykle o funkce, které usnadňují a zpříjemňují práci s nástrojem.

Tabulka níže uvádí některé rozdíly těchto verzí SoapUI. Detailní srovnání poskytuje výrobce a je k dispozici na stránkách:

- http://www.soapui.org/about-soapui-pro/product-comparison/soapui-and-soapui-pro.html.

Je však vhodné poznamenat, že řada věcí, které nabízí SoapUI NG Pro, je možné provést i ve verzi zdarma. Je však třeba více využít skriptovací jazyk Groovy, případně JavaScript, který je k dispozici i ve verzi SoapUI OS.

Srovnání verze SoapUI OS a verze Pro (Zdroj: autor)

Funkcionalita	**SoapUI OS**	**SoapUI NG Pro**
Podpora	Ano (ale pouze komunita)	Ano (oficiální podpora od vývojářů)
Environments (funkcionalita přepínání definovaných prostředí)	Ne	Ano
Tvorba a správa požadavků	Ne	Ano

SOAP	Ano	Ano
REST	Ano	Ano
Tvorba testů (test suits, test cases, assertions, proměnné)	Ano (omezená o některé funkce)	Ano
Tvorba mock-up (SOAP, REST)	Ano	Ano
Groovy, JavaScript	Ano	Ano
WS-Security, WS-I Integration	Ano	Ano
SSL podpora	Ano	Ano
Zátěžové testy (Load Testing)	Ano (omezené)	Ano
Bezpečnostní testy (Security Testing)	Ano	Ano (navíc automatické generování)
HTTP, JDBC, WSDL, JMS	Ano	Ano
Automatizace testů	Ano	Ano
Reporty	Ano (omezené)	Ano

Příloha D: Záznam o průběhu testování

<Název projektu>	
Záznam o průběhu testování	Datum: <dd/mm/rrrr>

<NÁZEV PROJEKTU>

ZÁZNAM O PRŮBĚHU TESTOVÁNÍ

1 ÚVOD

1.1 ÚČEL

[Jaký je účel tohoto dokumentu?]

1.2 ROZSAH

[Jaký je rozsah tohoto dokumentu? Je možné z něj vycházet pouze v rámci tohoto projektu?]

1.3 DEFINICE A ZKRATKY

[Existují nějaké pojmy a zkratky, které by měly být pro srozumitelnost tohoto dokumentu vysvětleny?]

1.4 ODKAZY

[Vychází tento dokument z nějakých jiných výstupů, popř. odkazuje na nějaké další dokumenty?]

1.5 HISTORIE VERZÍ

[Kdo, kdy a jak tento dokument upravoval?]

Datum	Verze	Popis	Autor

2 Testovaný IS/ICT

[Jaký IS/ICT, popřípadě jaká jeho část či verze bude testována?]

3 Stav testování

[Popis průběhu testování a případné problémy a rizika.]

4 Seznam provedených testů

[Seznam všech provedených testů, který vychází z dokumentu Zaznam_vysledku_testu_sablona.docx]

ID testovací sady	ID testovacího případu	Název testovacího případu	Tester	Výsledek

5 Chyby

[Seznam nalezených chyb, který vychází z dokumentu Zaznam_vysledku_testu_sablona.docx nebo reportu chyb]

Závažnost chyby	Počet chyb	
	Celkem	Nových
A		
B		
C		
D		

Příloha E: Akceptační protokol

<Název projektu>	
Akceptační protokol	Datum: <dd/mm/rrrr>

<NÁZEV PROJEKTU>

AKCEPTAČNÍ PROTOKOL

1 ÚVOD

1.1 ÚČEL

[Jaký je účel tohoto dokumentu?]

1.2 ROZSAH

[Jaký je rozsah tohoto dokumentu? Je možné z něj vycházet pouze v rámci tohoto projektu?]

1.3 DEFINICE A ZKRATKY

[Existují nějaké pojmy a zkratky, které by měly být pro srozumitelnost tohoto dokumentu vysvětleny?]

1.4 ODKAZY

[Vychází tento dokument z nějakých jiných výstupů, popř. odkazuje na nějaké další dokumenty?]

1.5 HISTORIE VERZÍ

[Kdo, kdy a jak tento dokument upravoval?]

Datum	Verze	Popis	Autor

2 Testovaný IS/ICT

2.1 Název

[Název a případně další základní informace o testovaném IS/ICT.]

2.2 Zhotovitel

[Informace o zhotoviteli.]

2.3 Zákazník

[Informace o zákazníkovi.]

3 Akceptační kritéria

[Seznam akceptačních kritérií nebo odkaz na dokument, ve kterém jsou akceptační kritéria definována.]

4 Seznam výhrad

[Seznam všech nalezených výhrad.]

ID	Výhrada	Závažnost	Termín a způsob vyřešení výhrady

5 Výsledek akceptace

[Výsledek akceptace - hodící se zaškrtne.]

- Akceptováno bez výhrad
- Akceptováno s výhradami
- Neakceptováno

V ..... dne .....

Zákazník		Zhotovitel	
Jméno		Jméno	
Podpis		Podpis	

Příloha F: Standardy tvorby testů v SoapUI

<NÁZEV PROJEKTU>

STANDARDY TVORBY TESTŮ V SOAPUI

1 ÚVOD

1.1 ÚČEL
[Jaký je účel tohoto dokumentu?]

1.2 ROZSAH
[Jaký je rozsah tohoto dokumentu? Je možné z něj vycházet pouze v rámci tohoto projektu?]

1.3 DEFINICE A ZKRATKY
[Existují nějaké pojmy a zkratky, které by měly být pro srozumitelnost tohoto dokumentu vysvětleny?]

1.4 ODKAZY
[Vychází tento dokument z nějakých jiných výstupů, popř. odkazuje na nějaké další dokumenty?]

1.5 HISTORIE VERZÍ
[Kdo, kdy a jak tento dokument upravoval?]

Datum	Verze	Popis	Autor

2 VERZE NÁSTROJE

[Pro jakou verzi nástroje jsou testy vytvářeny? Pro verzi OS nebo Pro? Toto má například dopad na možnosti kontrol, které mohou být v rámci testovacích skriptů použity.]

3 STRUKTURA TESTOVACÍCH SKRIPTŮ

[Definování struktury testovacích skriptů. Dělení vychází především z potřeb daného projektu.

Základní struktura (vychází ze struktury v SoapUI):

⇨ **TestSuit** (odpovídá testovací sadě – jedná se o sadu testovacích případů sdružených za účelem otestování určité části webové služby)

⇨ **TestCase** (vlastní testovací skript – jedná se o zápis testovacího případu v nástroji SoapUI)

⇨ **Test Steps** (jednotlivé kroky nutné pro provedení testovacího skriptu)

]

4 JMENNÁ KONVENCE

[Pravidla pojmenovávání TestSetů, TestCasů, TestStepů, proměnných a dalších objektů, které během tvorby testovacích skriptů mohou vzniknout.]

5 PRAVIDLA POUŽÍVÁNÍ KOMENTÁŘŮ

[Definování pravidel pro komentování částí testovacích skriptů.]

6 PRAVIDLA POUŽÍVÁNÍ PROMĚNNÝCH

[Proměnné lze v SoapUI definovat na několika úrovních – obsahem je stanovit, kde a jakým způsobem jednotlivé typy proměnných používat.]

7 Pravidla konfigurace připojení na externí systémy

[V rámci testovacích skriptů lze definovat napojení na externí systémy. Obsahem je definice, jak napojení vytvářet a používat napříč projektem (může se jednat například o JDBC).]

8 Testovací data

[Stanovení, jak přistupovat k testovacím datům. Testovací data lze definovat vně nebo uvnitř skriptu. Mohou být staticky nebo dynamicky generována.]

9 Znovupoužitelnost

[Pravidla, jak psát testovací skripty, aby jejich části mohly být například přepoužity v jiném testu. Výchozím předpokladem je parametrizace testovacích skriptů a jejich uspořádání do logických celků.]

10 Zátěžové testy

[Pravidla a koncepty zátěžových testů v SoapUI.]

11 Bezpečnostní testy

[Pravidla a koncepty bezpečnostních testů v SoapUI.]

12 Automatizované spouštění

12.1 Návrh řešení

[Popis řešení automatizovaného spouštění, použité technologie (TestRunner, zapojení do CI ad.), zapojení serverů, testovacích prostředí apod.]

12.2 Pravidla spouštění

[Stanovení periodicity spouštění, výběr testů (testovacích sad, testovacích skriptů).]

12.3 Reportování chyb

[Určení pravidel pro reportování chyb z automatizovaného spouštění testů. Může se jednat o zadávání automatizované nebo manuální.]