České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Václav Rechtberger**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Modul do systému PCTgen pro generování testovacích scénářů pro kontrolu datové konzistence**

Pokyny pro vypracování:

Navrhněte a implementujte modul do aplikace PCTgen pro generování testovacích scénářů pro kontrolu konzistence datových objektů v aplikaci. Modul bude umožňovat vytváření CRUD matice v tabulce, import a export matice ve formátu CSV a XML a generování testovacích scénářů z CRUD matice podle postupu definovaného v metodice TMAP a rozšířeného algoritmu dodaného vedoucím práce. Zároveň bude jeho součástí funkcionalita pro kontrolu konzistence mezi CRUD maticemi a workflow modelovanými v jiném (již implementovaném) modulu aplikace PCTgen. Funkcionalitu implementovaného modulu ověřte sadou vhodných testů.

Seznam odborné literatury:

Koomen T. et al.: TMap Next, for result-driven testing. UTN Publishers, 2007.
Pecinovský, R.: Návrhové vzory, Computer Press, 2007.
Ray, E. T.: Learning XML, Second Edition. O'Reilly, 2003.

Vedoucí: Ing. Miroslav Bureš, Ph.D.

Platnost zadání: do konce zimního semestru 2016/2017

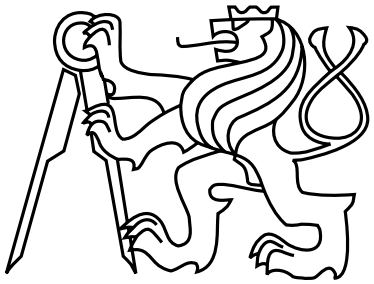
doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 22. 10. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Modul do systému PCTgen pro generování testovacích scénářů pro kontrolu datové konzistence

Václav Rechtberger

Vedoucí práce: Ing. Miroslav Bureš, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

27. května 2016

Poděkování

Zde bych chtěl poděkovat panu Ing. Miroslavu Burešovi, Ph.D. za odborné vedení práce, věcné připomínky, dobré rady a vstřícnosti při konzultacích a vypracováníbakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 27. 5. 2016 .....

Abstract

In this thesis a module to system PCTgen is going to be developed. The sytem now provides generating of process cycle tests. The module is going to serve the purpose of generating test scenarios for the control of data consistency. Possibly every application works with data and these data can be damaged during working with them. It is necessary to find this damage in time. We cannot always wait untill the full effect of the damage, because this finding could cause great problems and consequences. It is necessary to choose a technique how to find these damages. The product I am developing is using the DcyT technique and its extended version EDCyT. The product of these techniques as well as of my module are the testing scenarios thanks to which it is possible to find the data damaging the operations.

Abstrakt

V rámci práce bude vyvýjen modul do systému PCTgen, který nyní poskytuje generování process cycle testů. Modul bude sloužit ke generování testovacích scénářů pro kontrolu datové konzistence. Snad každá aplikace pracuje s daty a tato data mohou být během práce s nimy poškozena. Toto poškození je potřeba včas odhalit. Odhalení chyb nemůže vždy čekat na první projev této chyby v plném nasazení, neboť by takové odhalení sebou mohlo nést velké problémy a drtivé následky. K odhalení chyb je potřeba zvolit nějakou techniku. Mnou vyvýjený produkt bude využívat techniky DCyT a jeho rozšířenou variantu EDCyT. Produktem těchto technik a tedy i mého modulu jsou testovací scénáře pomocí nichž lze odhalit zmíněné data poškozující operace.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Tato bakalářská práce se zabývá realizací modulu do systému PCTgen. Modul slouží ke generování testovacích scénářů pro kontrolu konzistence datových objektů. Systém PCTgen je open source freeware nástroj sloužící k testování software, v aktuální verzi slouží ke generování testovacích případů pro aplikační procesy a workflow. Rozšíření jeho funkcionalit je úkolem této práce.

Systémy pracující s daty nad těmito daty provádějí, krom čtecích, různé editující operace. Při těchto operacích mohou být tato data poškozena, respektive špatně vytvořena či smazána. Tento defekt je nutné hlídat a takovéto operace odhalit již ve fázi vývoje. Jelikož tento defekt může být zjistitelný/projevující se pouze za specifických okolností, je zapotřebí přistupovat k této problematice sofistikovanějšími způsoby než-li náhodným testovaním. Tento přístup nevolíme pouze za účelem účinnějšího odhalení těchto defektů, ale i z důvodu jejich následného znovunasimulování a lokalizace při jejich odstraňování.

Defekty v software nehledáme pouze z důvodu nezklamání zákazníka. V dnešní době provádějí počítače i velmi kritické úkony od řízení dopravy, přez manipulaci s financemi, po řízení vojenské či vesmírné techniky. V těchoto odvětvích by mohla mít i na první pohled drobná chyba fatální následky.

Jak již bylo řečeno, k odhalení těchto chyb je potřeba zvolit nějakou techniku. Mnou vyvýjený modul využije techniky DCyT, definované v metodice TMAP, a EDCyT, jež je rozšířením techniky DCyT. Tyto techniky uvádějí postup ke generování testovacích scénářů, jejichž provedení by mělo odhalit případné chyby v software.

Ptáte se proč jen mělo? Jednak vždy závisí na kvalitě aplikování konkrétní techniky, ale také na testovacích datech. Byť počítače representují i nekonečné množiny pomocí konečných množin, není vždy možné otestovat všechny možné kombinace vstupních dat. Existují sice techniky, které se snaží tyto kombinace redukovat bez ztráty šance na odhalení chyby, avšak zde opět závisí na kvalitě jejich aplikace. Právě z těchto důvodů jsou testovací techniky užitečné k odhalení chyb, nikoli však dostatečné k dokázání jejich absence.

1.1 Struktura práce

V následujících kapitolách nejprve provedu analýzu, v které uvedu testovací techniky, které budou realizovány ve vyvíjeném modulu. Poté uvedu požadavky na systém, které v dostatečné míře vysvětlím. Na konec uvedu případné změny/úpravy oproti zadání, které náležitě vysvětlím.

Dále uvedu návrh modulu. Pro ohraničení rozsahu práce sestavím případy užití, které popíši, doplním o uživatelské scénáře a pro zajištení úplnosti modulu namapuji na jednotlivé funkční požadavky. Všechny tyto funkcionality bude nutno začlenit do stávajícího systému, toto bude obsahem další sekce. Tuto integraci doplním vhodnými mockupy a náležitě okomentuji. Tento popis a mockupy by měl zároveň posloužit jako návod k používaní modulu. V závěru této kapitoly vás seznámím s navrženým datovým modelem CRUD matice, modelem generovaných scénářů a způsobu jejich generování.

Další kapitolou, která následuje je již implementace. Zde zvolím IDE pro vyvíjení software. Seznámím vás se svým postupem při implementaci a s výsledným produktem. Vysvětlení strukury kódu bude zahrnovat základní balíčky, diagram komponent, nasazení a základních tříd. Dále uvedu také utility pro ukládáni dat a generování scénářů.

Před závěrem práce zbývá už jen produkt otestovat. Zde provedu jednotkové a integrační testy. Uživatelské akceptační testy (scénáře k jejich provedení) budou přiloženy v příloze. V závěrečné zprávě sdělím své poznatky z provedené práce, pozitiva i negativa.

V následujících kapitolách nejprve provedu analýzu v níž zjistím současný stav problematiky, na základě zadání a této analýzy zformuluji požadavky a uvedu úpravy oproti zadání plynoucí z provedené analýzy. Nejedná se tedy o svévolné změny, ale změny vyvolané zjištěnými omezeními či možnostmi v kladném smyslu. Následně navrhnu řešení, kde rozvedu všechny aspekty modulu, od případů užití až po návrh datového modelu a způsobu generování scénářů. Následuje implementační část, ta nás provede implementací modulu, od volby IDE po ukázky výsledného produktu. Následně nemůže chybět otestování výsledného produktu. Testování by nemělo chybět v žádném projektu, natož v projektu zabývajícím se samotným testováním. Na závěr shrnu své poznatky při řešení této práce.

Kapitola 2

Analýza

Cílem analýzy je zjistit a jasně definovat požadavky zákazníka/zadavatele a ty zohlednit dle dostupných technologií a jejich možností. Tento dokument se stane výchozím bodem následného návrhu. Při analýze a následném návrhu budu čerpat z kniny Softwarové inženýrství [9]. Diagramy budou modelovány v systému Entnerprise Architect.

2.1 Analýza současného stavu

Tuto analýzu pojmu pouze z pohledu testovacích technik. Analýza alternativního software je vzhledem k nedostupnosti, alespoň na volném trhu, takového software neproveditelná.

2.1.1 DCyT

Technika pro test datové konzistence definovaná v metodice *TMap next* [6]. Tato technika není přímo určena k testování jednotlivých funkcí, ale především jejich integrace. Proto je pro efektivitu této techniky vhodné, aby byly samotné funkce již otestovány. Hlavním zdrojem dat pro generování testovacích scénářů je *CRUD matice*, bez níž není možné generovat scénáře.

CRUD matice je matice nesoucí informaci o tom jaké operace jsou na jednotlivých entitách E prováděny jednotlivými funkcemi F. Tato technika operuje s následujícími příznaky:

- **Create (C)** - vytvoření entity
- **Read (R)** - čtení entity
- **Update (U)** - editace entity
- **Delete (D)** - smazání entity

Na základě dat z této CRUD matice budou následně generovány testovací scénáře. Tyto scénáře mají podobu uspořádaného seznamu/posloupnosti funkcí $F' \subset F$. Pro každou entitu je vytvořen jeden testovací scénář. Tento scénář je tvořen funkcí s příznakem C, všemi funkcemi s příznakem U a na závěr funkcí s příznakem D, v tomto pořadí, tato sekvence bude nadále nazývvána *sekvencí C,U,D*. Za každou z těchto funkcí následuje *sekvence(i jednoprvková) funkcí s příznakem R*. Všechny tyto příznaky jsou míněny ve vztahu k entitě, pro kterou je scénář generován.

Technika DCyT rozeznává dvě úrovně pokrytí.

2.1.1.1 Základní pokrytí (basic coverage)

Je tvořeno výše zmíněnou sekvencí C,U,D kdy každá z těchto operací je následována pouze jednou funkcí s příznakem R. Tato funkce může být volena na základě priorit či jiných znalostí, jsou-li k dispozici. V opačném případě je volba náhodná, je však dobré využít co nejvíce funkcí (neopakovat stále jednu funkci). Tímto principem se řídí druhá hloubka pokrytí.

2.1.1.2 Úplné R (complete R coverage)

Ta je opět tvořena sekvencí C, U, D, ovšem v tomto případě je každá operace následována všemi R operacemi.

2.1.2 EDCyT

Technika vyvíjená na ČVUT v Praze [5]. Rozvíjí a vylepšuje techniku DCyT2.1.1. Zavádí nové příznaky do CRUD matice. Těmito příznaky jsou:

- **Influence (I)** - ovlivnění entity e1 operací C, U, D funkce f provedené na entitě e2, zapisujeme jako I(e2) v poli náležící k e1 a funkci f
- **Best read (B)** - nejlepší čtení entity, tato operace má největší předpoklad odhalit chybu
  - Funkce f čte nejvíce attributů entity e
  - Funkce f je nejfrekventovanější R funkce na entitě e
  - Funkce f je využívána dalšími funkcemi ke čtení entity e
- **I'm not sure (ImNS)** - příznak nejistoty, doprovodný příznak k ostatním příznakům

Tato technika rozšiřuje/upravuje techniku DCyT následujícími úrovněmi pokrytí.

2.1.2.1 Simple read (0R)

Základní úroveň pokrytí z DCyT.

2.1.2.2 Simple best read (OB)

Upravené 0R, každá z operací sekvence C,U,D je následována operací B.

2.1.2.3 All reads one time (1R)

Sekvence operací C,U,D, každá z nich je následována jednou či více operacemi R. Každá z operací R je použita právě jednou. Pokud je počet všech C,U,D operací větší než počet všech operací R, potom použij pro zbývající U, D operace operaci B.

2.1.2.4 All reads one time, I operations reflected (1RI)

Jedná se o scénář generovaný pomocí 1R se zohlednením případných ovlivněných entit.

2.1.2.5 All reads after all changes (NR)

Úplné R pokrytí z DCyT.

2.1.2.6 All reads after all changes, I operations reflected (NRI)

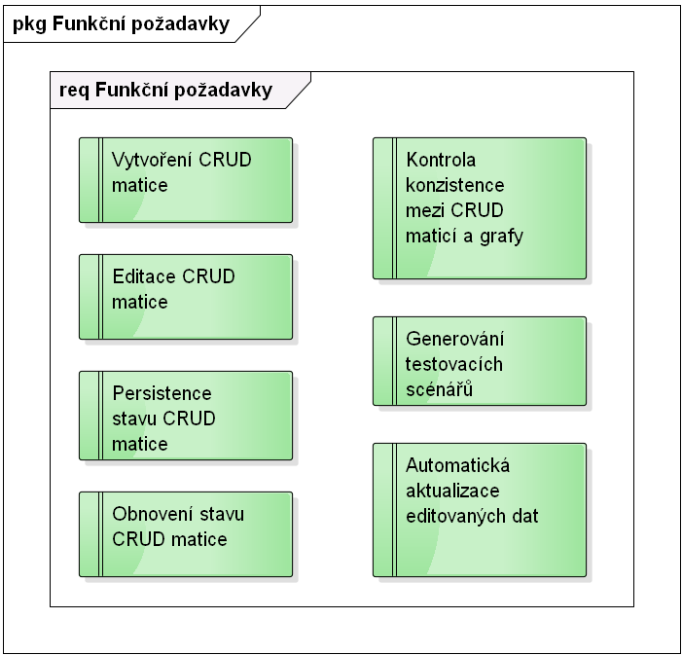
Jedná se o scénář generovaný pomocí NR se zohlednením případných ovlivněných entit.

2.2 Požadavky

Na základě zadání a analýzy shrnu požadavky a tím určím směr vývoje. Z těchto požadavků budu zjišťovat stav vývoje produktu a proto je potřeba aby tyto požadavky byly jednoznačné a šlo rozhodnout jednoduchou binární logikou zda jsou splněny.

2.2.1 Funkční požadavky

Funkční požadavky jsou funkcionality které systém uživateli umožnuje provést. Tyto požadavky jsou vystiženy následujícím diagramem 2.1.



Obrázek 2.1: Diagram funkčích požadavků

2.2.1.1 Vytvoření CRUD matice

Systém ke každému projektu založí CRUD matici.

2.2.1.2 Editace CRUD matice

U matice bude možno manipulovat s počtem funkcí/entit a nastavovat jejich atributy, dále bude možno nastavovat příznaky dle požadavků jmenovaných technik.

2.2.1.3 Uložení stavu CRUD matice

Výsledky editace uskutečněných na základě požadavku 2.2.1.2 je možné uložit.

XML Uložení stavu matice ve formátu XML v1.0. XML je značkovací jazyk sloužící k výměně dat, který byl vyvinut a standardizován konsorciem W3C [4]. Více informací o jazyce a práci s ním lze dohledat v knize [8]. Na struktutu těchto dat nejsou kladeny žádné specifické nároky, avšak bude dodán soubor DTD, oproti němuž lze generovaný dokument zvalidovat.

CSV Uložení stavu matice ve formátu CSV (viz [1]). Tento formát bude sloužit k editaci matice v tabulkových editorech. Je tedy nezbytné, aby si tento soubor zachoval vizuální vzhled matice a data v něm uložená nebyla nijak kódována (čitelné pro člověka), krom bežného escapování speciálních znaků jež je pro tento formát nezbytné. Toto escapování je součástí definice CSV formátu a bežné editory s ním umí pracovat. Případná omezení vyplývající z tohoto požadavku(čitelnost pro člověka) budou akceptována, avšak je nutno tato omezení náležitě zdokumentovat.

2.2.1.4 Obnovení stavu CRUD matice

Stav matice je možné obnovit z předešlých záloh vytvořených na základě požadavku 2.2.1.3.

2.2.1.5 Kontrola konzistence mezi CRUD maticí a grafy

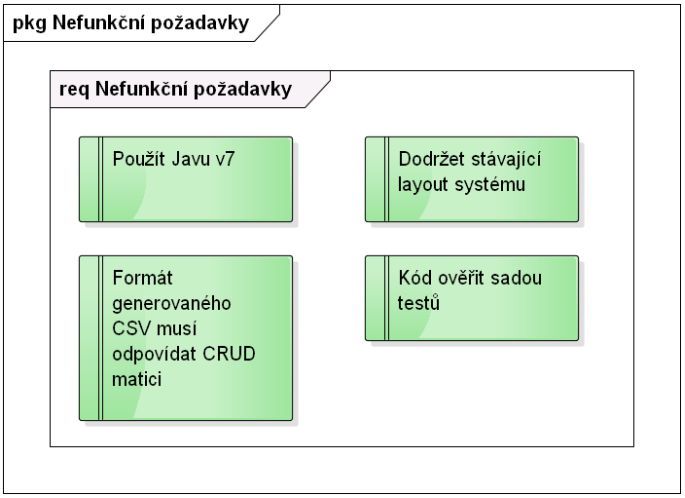
Systém podá uživateli report o nekonzistenci funkcí jmenovaných v grafech a funkcemi v CRUD matici.

2.2.1.6 Generování testovacích scénářů

Systém generuje testovací scénáře dle technik DCyT a EDCyT. Požadovány jsou všechny hloubky pokrytí.

2.2.1.7 Nefunkční požadavky

Nefunkční požadavky jsou technické náležitosti systému jako například implementační jazyk, dostupnost systému či rychlost odezvy. Tyto požadavky jsou vystiženy následujícím diagramem 2.2.



Obrázek 2.2: Diagram obecných požadavků

2.3 Změny oproti zadání

2.3.1 Testovací scénáře

Testovací scénáře s hloubkou pokryti 1R a 1RI mají dle zadání projít všechny ready a pokračovat best ready, pokud by však žádné best ready nebyly k dispozici, dle zadání by se již nečetlo, to je ovšem v rozporu s účelem testovaní a proto v případě absence best readu bude nadále čteno pomocí readů.

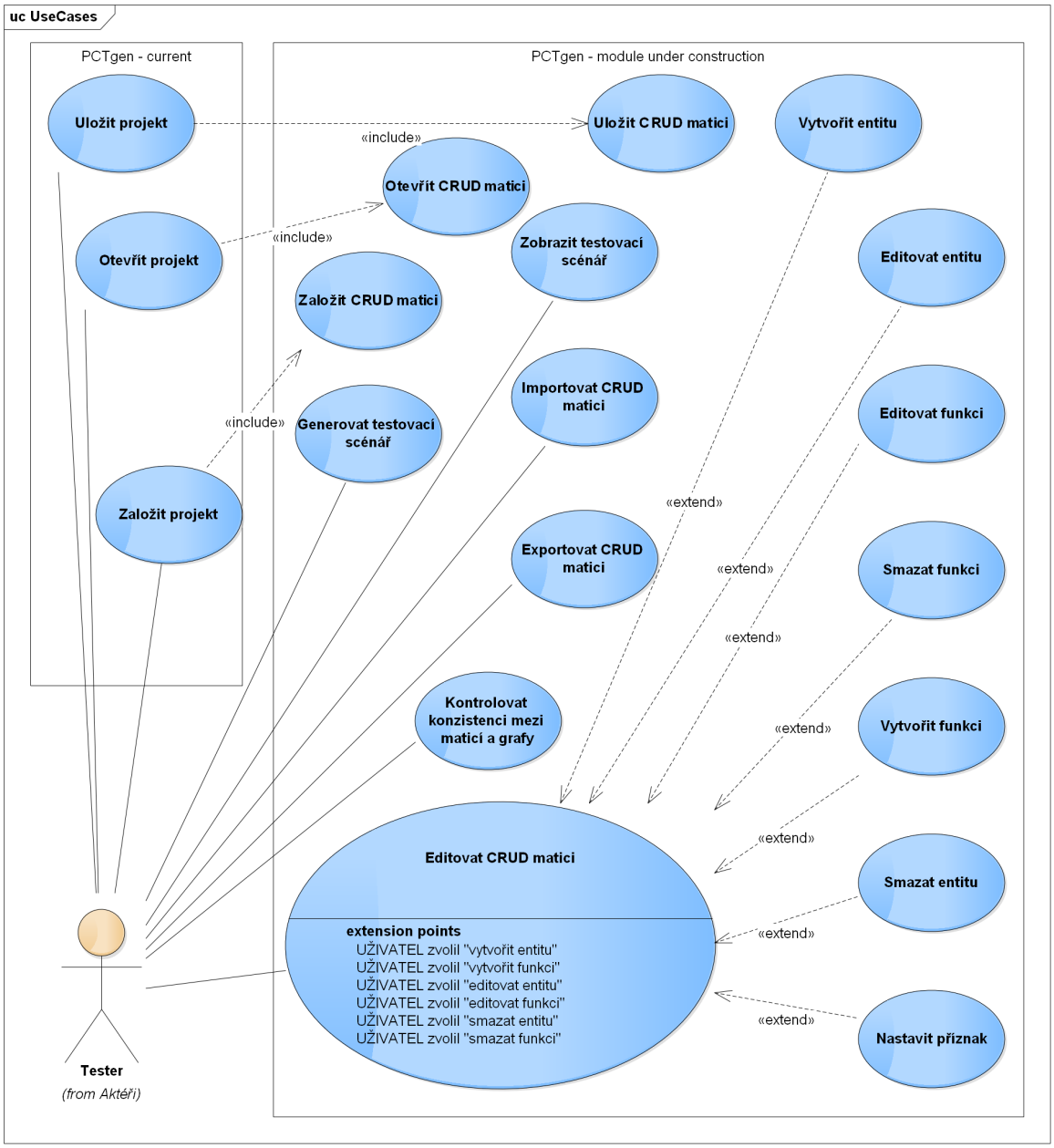
Kapitola 3

Návrh

Návrh aplikace je stěžejní část dokumentu, z níž budu čerpat při implementaci. Ke grafickým návrhům systému/modulu budu používat free nástrol Pencil Prorotyping.

3.1 Případy užití

Systém rozeznává jen jednoho uživatle/aktéra - **Testera** - uživatel od něhož jsou očekávny znalosti testování software.



Obrázek 3.1: Diagram případů užití

3.1.1 Uživatelské scénáře

3.1.1.1 Založit CRUD matici

Hlavní scénář:

1. SYSTÉM vytvoří novou CRUD matici.

2. SYSTÉM přiřadí nově vytvořenou CRUD matici ke konkrétnímu projektu.

3.1.1.2 Založit projekt

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „nový projekt“.
2. SYSTÉM vytvoří nový projekt.
3. «INCLUDE» Založit CRUD matici 3.1.1.1.
4. SYSTÉM vloží projekt do seznamu projektů.

3.1.1.3 Uložit CRUD matici

Hlavní scénář:

1. SYSTÉM vygeneruje XML dokument nesoucí informaci o matici.

3.1.1.4 Uložit projekt

Vstupní podmínky:

- Je otevřen alespoň jeden projekt a tento projekt je vybrán.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „uložit projekt“.
2. SYSTÉM vyžádá zadání názvu souboru a místa jeho uložení.
3. UŽIVATEL zadá požadované údaje.
4. IF soubor již existuje THEN:
   4.1. SYSTÉM se dotáže, zda má být soubor přepsán.
   4.2. UŽIVATEL odpoví na dotaz
5. SYSTÉM vytvoří dokument nesoucí informaci o projektu(již implementovaná část).
6. «INCLUDE» Uložit CRUD matici 3.1.1.3.
7. SYSTÉM sloučí dokument s daty projektu s daty matice.
8. SYSTÉM uloží dokument do zvoleného souboru.

Alternativní scénář 1:

1. UŽIVATEL zrušil výběr souboru tlačítmkem „STORNO“.

Alternativní scénář 2:

1. UŽIVATEL byl dotázán k přepsání souboru a ten toto zamítl.

Výjimečný scénář:

1. Nepodařilo se zapsat soubor.

3.1.1.5 Otevřít CRUD matici

Hlavní scénář:

1. SYSTÉM vytvoří novou CRUD matici.
2. SYSTÉM na základě dat obnoví stav matice.
3. SYSTÉM přiřadí nově vytvořenou CRUD matici ke konkrétnímu projektu.

Výjimečný scénář:

1. Nepodařilo se obnovit stav matice. SYSTÉM tuto informaci oznámí UŽIVATELI.

3.1.1.6 Otevřít projekt

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „otevřít projekt“.
2. SYSTÉM vyžádá zadání cesty k souboru s projektem.
3. UŽIVATEL zadá cestu k souboru a potvrdí ji.
4. SYSTÉM načte soubor.
5. SYSTÉM vytvoří nový projekt a obnoví stav projektu.
6. SYSTÉM se pokusí lokalizovat v souboru data CRUD matice.
7. IF data matice lokalizována THEN:
   7.1. «INCLUDE» Otevřít CRUD matici 3.1.1.5
8. SYSTÉM vloží projekt do seznamu projektů.

Alternativní scénář:

1. UŽIVATEL zrušil výběr souboru tlačítmkem „STORNO“.

Výjimečný scénář:

1. Problém se čtením souboru.

3.1.1.7 Exportovat CRUD matici

Vstupní podmínky:

- Je vybrána CRUD matice

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „exportovat matici“.
2. SYSTÉM vyžádá zadání názvu souboru a místa jeho uložení.
3. UŽIVATEL zadá požadované údaje.
4. IF soubor již existuje THEN:
   4.1. SYSTÉM se dotáže zda má být soubor přepsán.
   4.2. UŽIVATEL odpoví na dotaz
5. SYSTÉM vygeneruje dokument nesoucí informaci o matici.
6. SYSTÉM uloží dokument do zvoleného souboru.

Alternativní scénář 1:

1. UŽIVATEL zrušil výběr souboru tlačítkem „STORNO“.

Alternativní scénář 2:

1. UŽIVATEL byl dotázán k přepsání souboru a ten toto zamítl.

Výjimečný scénář:

1. Nepodařilo se zapsat soubor.

3.1.1.8 Importovat CRUD matici

Vstupní podmínky:

- Existuje a je zvolen projekt, do kterého bude matice importována.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „importovat matici“.
2. SYSTÉM vyzve k zadání cesty k souboru s maticí.
3. UŽIVATEL zadá požadované údaje.
4. SYSTÉM obnoví stav matice ze zvoleného souboru.

Alternativní scénář:

1. UŽIVATEL zrušil výběr souboru tlačítmkem „STORNO“.

Výjimečný scénář:

1. Nepodařilo se otevřít/číst soubor.

3.1.1.9 Zobrazit testovací scénář

Vstupní podmínky:

- Z matice byl již vygenerován scénář.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolil „zobrazit testovací scénář“ ve stromu projektů.
2. SYSTÉM zobrazí scénář.
3. WHILE UŽIVATEL prohlíží scénář.
   3.1. SYSTÉM zobrazuje požadovaná data
4. UŽIVATEL ukončil prohlížení.

3.1.1.10 Generovat testovací scénář

Vstupní podmínky:

- Je otevřen alespoň jeden projekt, z jehož CRUD matice se bude scénář generovat.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „generovat scénář“ (Každá hloubka pokrytí má své tlačítko).
2. SYSTÉM vygeneruje scénář.
3. SYSTÉM tento scénář zařadí k příslušné matici, z které byl generován.

Výjimečný scénář:

1. Data v matici nebyly adekvátní ke generování korektního scénáře. SYSTÉM nijak nereaguje ale generovaný scénář není korektní. Od UŽIVATELE je předpokládána znalost techniky a jejich požadavků, viz 3.1.

3.1.1.11 Kontrolovat konzistenci mezi maticí a grafy

Vstupní podmínky:

- Je otevřen alespoň jeden projekt a tento projekt je vybrán.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „zkontrolovat konzistenci“.
2. SYSTÉM porovná funkce v matici s funkcemi v grafech.
3. SYSTÉM zobrazí report o zjištěných rozdílech.
4. UŽIVATEL prohlídne report.

3.1.1.12 Vytvořit entitu

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „vytvořit entitu“.
2. SYSTÉM vytvoří v matici další entitu.
3. SYSTÉM obnoví view matice.

3.1.1.13 Editovat entitu

Vstupní podmínky:

- V CRUD matici je vytvořena alespoň jedna entita.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „editovat entitu“.
2. SYSTÉM zobrazí editační formulář.
3. WHILE UŽIVATEL nepřeruší editaci jinou akcí.
   3.1. UŽIVATEL zadá nové údaje.
   3.2. SYSTÉM automaticky uloží data.
   3.3. SYSTÉM obnoví view matice.

Alternativní scénář:

1. UŽIVATEL zadal data která by poručila konzistenci matice(např. název entity by byl duplicitní).
2. SYSTÉM data neuloží.
3. návrat do bodu 3 hlavního scénáře

3.1.1.14 Smazat entitu

Vstupní podmínky:

- V CRUD matici je vytvořena alespoň jedna entita.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „smazat entitu“.
2. SYSTÉM smaže entitu.
3. SYSTÉM obnoví view matice.

3.1.1.15 Vytvořit funkci

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „vytvořit funkci“.
2. SYSTÉM vytvoří v matici další funkci.
3. SYSTÉM obnoví view matice.

3.1.1.16 Editovat funkci

Vstupní podmínky:

- V CRUD matici je vytvořena alespoň jedna funkce.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „editovat funkci“.
2. SYSTÉM zobrazí editační formulář.
3. WHILE UŽIVATEL nepřeruší editaci jinou akcí.
   3.1. UŽIVATEL zadá nové údaje.
   3.2. SYSTÉM automaticky uloží data.
   3.3. SYSTÉM obnoví view matice.

Alternativní scénář:

1. UŽIVATEL zadal data která by poručila konzistenci matice(např. název funkce by byl duplicitní).
2. SYSTÉM data neuloží.
3. návrat do bodu 4 hlavního scénáře

3.1.1.17 Smazat funkci

Vstupní podmínky:

- V CRUD matici je vytvořena alespoň jedna funkce.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „smazat funkci“.
2. SYSTÉM smaže funkci.
3. SYSTÉM obnoví view matice.

3.1.1.18 Editovat CRUD matici

Vstupní podmínky:

- Je otevřen alespoň jeden projekt a tento projekt je vybrán.

Hlavní scénář:

1. Scénář začíná, když UŽIVATEL zvolí „editovat matici“.
2. SYSTÉM zobrazí matici.
3. SYSTÉM zobrazí editační formulář.
4. WHILE UŽIVATEL nepřeruší editaci jinou akcí.
   4.1. UŽIVATEL zadá nové údaje.
   4.2. SYSTÉM automaticky uloží data.
   4.3. SYSTÉM obnoví view matice.

3.1.2 Mapování na požadavky

Source \ Target	Funkční požadavky::Automatická aktualizace editovaných dat	Funkční požadavky::Editace CRUD matice	Funkční požadavky::Generování testovacích scénářů	Funkční požadavky::Kontrola konzistence mezi CRUD maticí a grafy	Funkční požadavky::Obnovení stavu CRUD matice	Funkční požadavky::Uložení stavu CRUD matice	Funkční požadavky::Vytvoření CRUD matice
Případy užití::Editovat CRUD matici	↑	↑					
Případy užití::Editovat entitu	↑	↑					
Případy užití::Editovat funkci	↑	↑					
Případy užití::Exportovat CRUD matici						↑	
Případy užití::Generovat testovací scénář			↑				
Případy užití::Importovat CRUD matici					↑		
Případy užití::Kontrolovat konzistenci mezi maticí a grafy				↑			
Případy užití::Nastavit příznak	↑	↑					
Případy užití::Otevřít CRUD matici					↑		
Případy užití::Otevřít projekt					↑		
Případy užití::Smazat entitu	↑	↑					
Případy užití::Smazat funkci	↑	↑					
Případy užití::Uložit CRUD matici						↑	
Případy užití::Uložit projekt						↑	
Případy užití::Vytvořit entitu	↑	↑					
Případy užití::Vytvořit funkci	↑	↑					
Případy užití::Založit CRUD matici							↑
Případy užití::Založit projekt							↑
Případy užití::Zobrazit testovací scénář			↑				

Obrázek 3.2: Mapování scénářů užití na funkční požadavky

Mapování případů užití na požadavky je důležité k ověření, že veškeré funkční požadavky budou implementovány.

3.2 Integrace na stávající systém PCTgen

3.2.1 Nástrojové lišty

Nyní má uživatel k dispozici pouze lištu pro práci s grafy. K této přibude nástrojová lišta vztahující se ke CRUD matici. Mezi těmito dvouma lištami bude automaticky přepínáno na základě jejich potřeby.

3.2.2 Projektový strom

Projektový strom je nyní tvořen seznamem projektů, kdy každý projekt pod sebou nese seznam grafů. Pod každým grafem se zas agregují vygenerované scénáře. Seznam pod projektem bude rozšířen o následující výčet položek: CRUD matice, Funkce, Entity. Pod CRUD maticí se budou agregovat vygenerované scénáře stejně jako pod grafy.

3.2.3 Hlavní pracovní plocha

V aktuální verzi je pracovní plocha tvořena pouze plochou pro práci s grafy. K této ploše přibydou plochy: CRUD matice, seznam Funkcí a seznam Entit. Mezi těmito plochami (i plochou pro práci s grafy), bude automaticky přepínáno dle potřeby na základě aktivit uživatele nad projektovým stromem. Plochy seznamu Funkcí a Entit budou neaktivní, nebudou tedy nijak interagovat s uživatelem, jejich obsah se bude měnit pouze na základě uživatelových aktivit na zbylých plochách. Seznam funkcí bude tvořen funkcemi z matice i grafů. Komu by mapování *položka projektového stromu* $\mapsto$ *pracovní plocha* nebylo již jasné, může tyto informace dohledat v následujícím schématu:

- *graf* $\mapsto$ *pracovní plocha pro práci s grafy*
- *CRUD matice* $\mapsto$ *pracovní plocha pro práci s CRUD maticí*
- *Funkce* $\mapsto$ *pracovní plocha zobrazujicí seznam funkcí*
- *Entity* $\mapsto$ *pracovní plocha zobrazujicí seznam entit*

3.2.4 Pravé panely

Pravé panely tvoří sekundární pracovní plochu částí projektu. V aktuální verzi tvoří tyto panely formuláře pro editaci různých údajů, potažmo zobrazování jejich obsahu. Tuto konveci dodržím, lépe řečeno, pravé panely použiji stejným způsobem.

3.2.4.1 Pravý panel CRUD matice

Panel zobrazuje sekundární údaje matice, její popis, linkování a verzi. Umožňuje taktéž editaci těchto údajů.

3.2.4.2 Pravý panel funkce z CRUD matice

Panel zobrazuje název a popis funkce, tyto údaje jsou editovatelné skrze toto rozhraní.

3.2.4.3 Pravý panel entity z CRUD matice

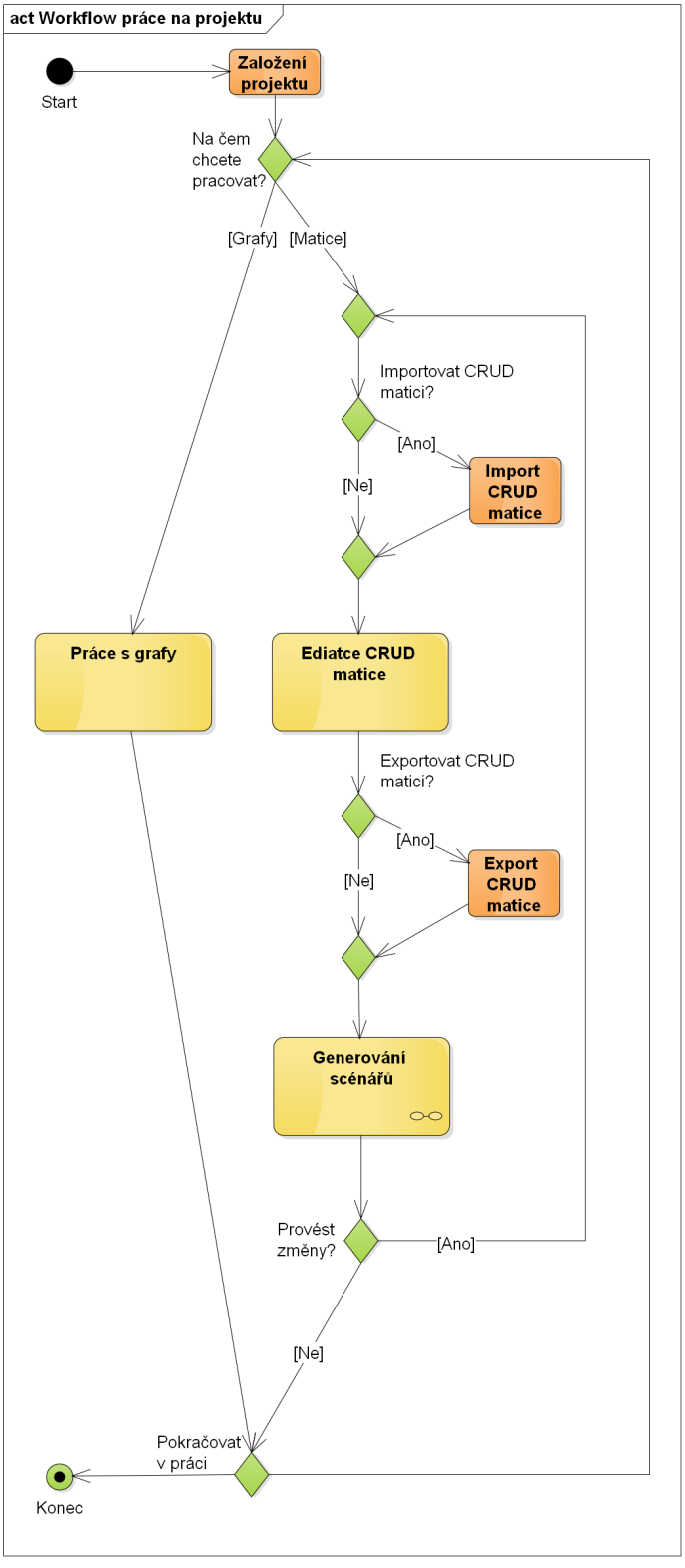
Panel zobrazuje název a popis entity, tyto údaje jsou editovatelné skrze toto rozhraní.

3.2.4.4 Pravý panel akce z CRUD matice

Tento panel slouží k nastavování příznaků akcí matice. Príznaky „create“, „reade“, „update“, „delete“, „best read“, „influence“ a u všech těchto příznaku ještě příznak „I’m not sure“.

3.3 Popis základních procesů a activity diagram

Toto flow je vnímáno z pohledu práce uživatele. Odezva systému by vždy znamenala pouze analogickou aktivitu na straně systému (např. když uživatel založí projekt, systém založí projekt).



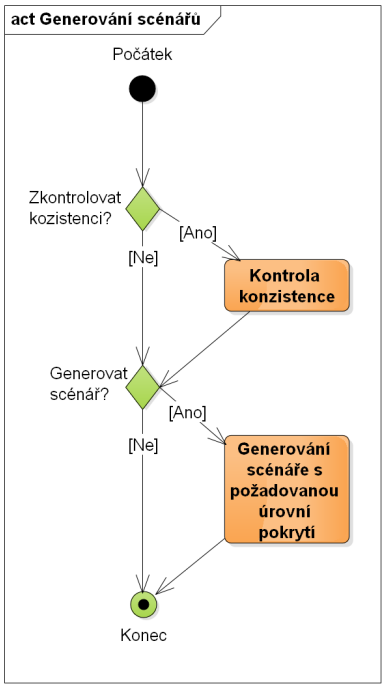
Obrázek 3.3: Workflow práce na projektu

3.3.1 Editace CRUD matice

Aktivita nastavení příznaků v matici potřebných ke generování scénářů. Dále je zde možno nastavit dokumentační náležitosti matice, či také název a popis funkcí a entit.

3.3.2 Generování scénářů

. Generování scénáře je cílovou akticitou uživatele. Uživatel má na výběr z několika úrovní pokrytí a také má možnost před generováním scénřů províést kontrolu konzistence mezi CRUD maticí a grafy workflow. Tato konzistence je myšlena z pohledu synchronicity funkcí.



Obrázek 3.4: Workflow generování scénářů

3.3.2.1 Kontrola konzistence

Zde uživatel provede kontrolu konzistence funkcí matice oproti grafům.

3.3.2.2 Generování scénáře s požadovanou úrovní pokrytí

Zde uživatel nechá vygenerovat scénář na základě zvolené úrovně pokrytí.

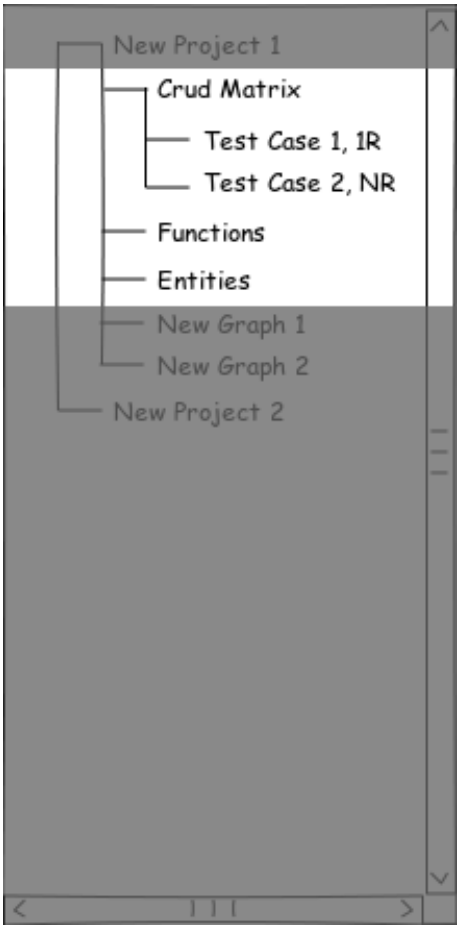
3.4 Návrh GUI a ovládání

Návrh GUI sestává z návrhů obrazovek s popisy co se na obrazovce odehrává. Návrhy budou mapovány na odpovídající aktivity z workflow. Jelikož tato kapitola seznámí čtenáře i s používaním a navigací v systému, zmíním zde i způsob založení, pořípadě uložení projektu či jeho otevření, byť toto je již implementováno. Více v uživatelské příručce na přiloženém CD.

Založení projektu je přístupné skrze hlavní lištu systému „File -> New Project“, popřípadě klávesovou skratkou „Ctrl + N“. Uložení projektu je přístupné skrze hlavní lištu systému „File -> Save Project“, popřípadě klávesovou skratkou „Ctrl + S“. Otebření projektu je přístupné skrze hlavní lištu systému „File -> Open Project“, popřípadě klávesovou skratkou „Ctrl + O“.

Ze jmenovaných položek vyplývá, že workflow je možné kdykoli přerušit, uložit a pokračovat v něm jindy. Popřípadě lze využít k zálohování práce. Toto je možné provést v kterékoli části flow, ovšem je třeba zmínit, že generované scénáře se neukládají. Systém také podporuje současnou práci na více projektech.

3.4.1 Navigace ve stromu projektů



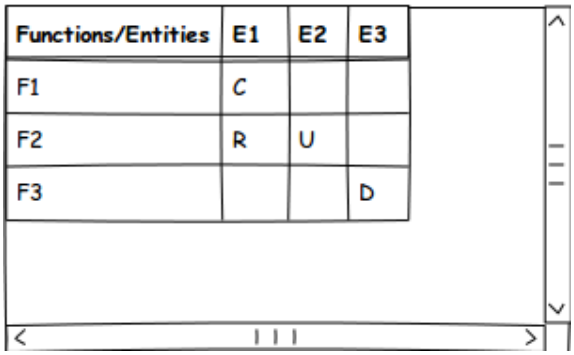
Obrázek 3.5: Návrh projektového stromu, zobrazuje jak modul zasáhne do stávajícího projektového stromu

Do stávajícího seznamu přibydou položky CRUD matice(plus vygenerované scénáře pod ní, slouží k zobrazení generovaných scénářů), funkce a entity. Každá položka je aktivní a po kliku se spustí požadovaná akce. Položka „Crud matrix“ zahájí aktivitu „Editace CRUD matice“ 3.3.1. Položka „Functions“ zobrazí seznam funkcí z matic i grafů, tato položka není ve flow znázorněna, ale je chápána jako součást práce na projektu, kdy v aktuální verzi slouží jako přehled jmenovaných funkcí v celém projektu. Položka „Entities“ je obdobou předešlé položky, avšak zobrazuje seznam entit(grafy s enitami/objekty nepracují). Položka „Editace CRUD matice“ je navíc doprovázena kontextovým menu(přístupné skrze pravé tlačítko myši) pro import a export matice.

3.4.2 Zobrazení dat

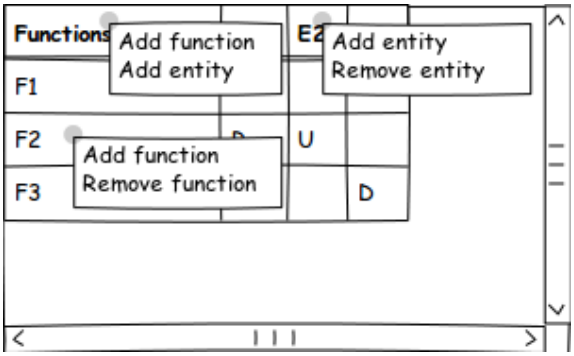
Modul pracuje se dvěma typy dat, s CRUD matící a s testovacími scénáři.

3.4.2.1 CRUD matice



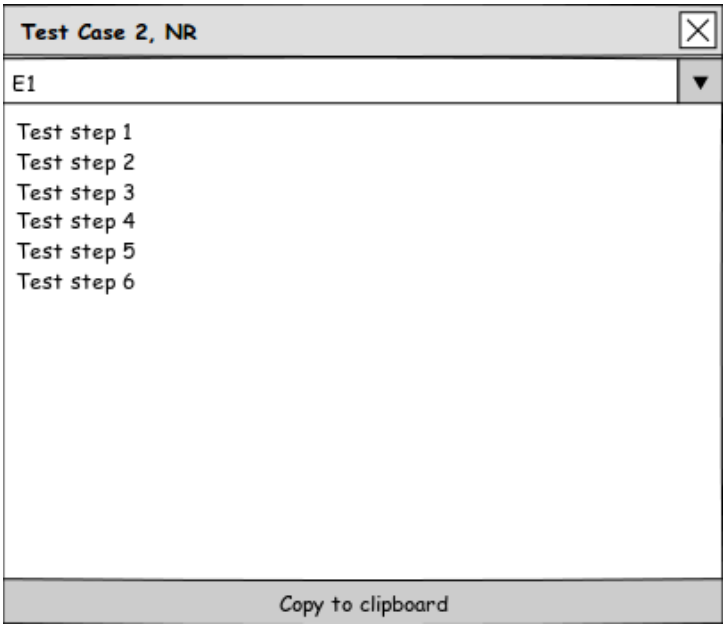
Obrázek 3.6: Návrh projektového stromu, zobrazuje jak modul zasáhne do stávajícího projektového stromu

Zobrazení matice probíhá v hlavní pracovní ploše projektu, údaje nejsou skrze tuto tabulku editovatelné, avšak po kliknutí na požadovanou položku je editace prováděna skrze pravý panel(viz Editace dat). Skrze kontextové menu (vyvoláme pravým tlačítkem myši) této tabulky je možné přídávat/odebírat funkce/entity. Příznak *I'm not sure* je v této tabulce rozlišen červenou barvou danného příznaku. Tato obrazovka je součástí aktivity „Editace CRUD matice“ 3.3.1.



Obrázek 3.7: Kontextová menu matice

3.4.2.2 Testovací scénáře

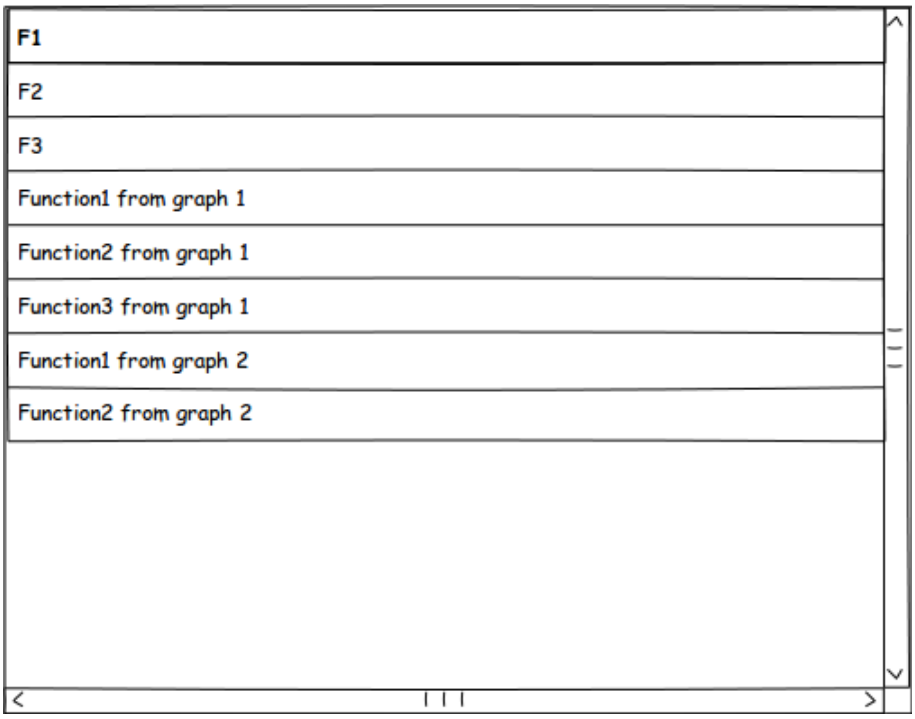


Obrázek 3.8: Návrh dialogového okna zobrazující vygenerované scénáře.

Přepínaní mezi scénáři vztahujím se k jednotlivým entitám je přepínáno pomocí ComboBoxu, v textovém poli pod ním je posléze zobrazen příslušný scénář. Zobrazené scénáře je možné nahrát do clipboardu pomocí tlačítka na spodu okna. Tato obrazovka je součástí aktivity „Generování scénářů“ 3.3.2.

3.4.2.3 Seznam funkcí

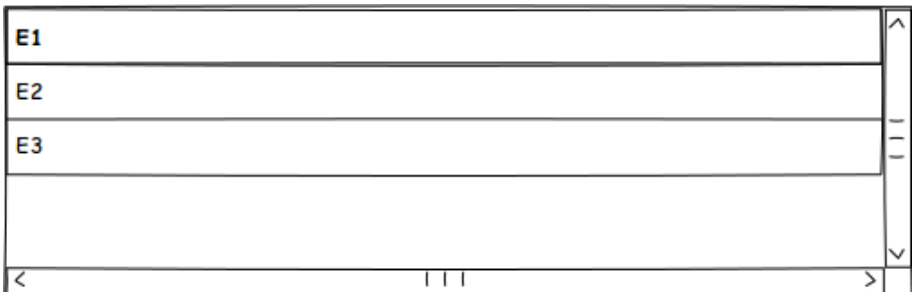
Zobrazuje seznam funkcí z CRUD matice a grafů. Tento list je needitovatelný a ani nevyvolává žádne akce.



Obrázek 3.9: Návrh listu entit

3.4.2.4 Seznam entit

Zobrazuje seznam entit z CRUD matice. Tento list je needitovatelný a ani nevyvolává žádne akce.

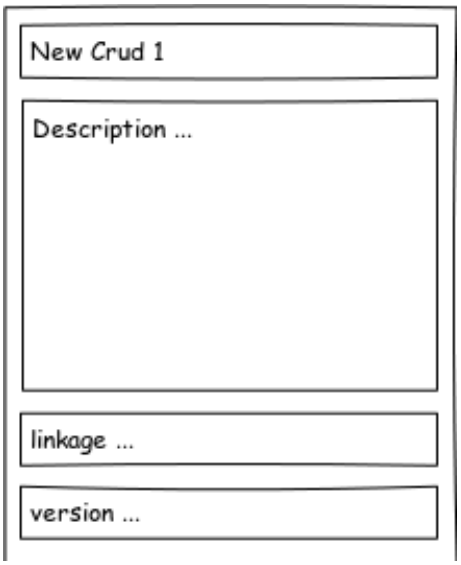


Obrázek 3.10: Návrh seznamu entit

3.4.3 Editace dat

Níže uvedené návrhy by bylo možné zařadit i do sekce zobrazení dat, ovšem zařazení do této sekce je vhodnější.

3.4.3.1 Pravý panel CRUD matice



Obrázek 3.11: Návrh pravého panelu CRUD matice, slouží k editaci informací o matici.

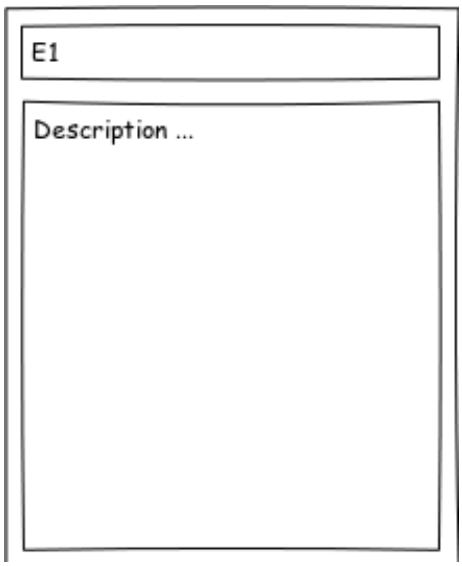
Pravý panel slouží k ediatci informací o matici. Tato obrazovka je součástí aktivity „Editace CRUD matice“ 3.3.1.

3.4.3.2 Pravý panel CRUD funkce

Obrázek 3.12: Návrh pravého panelu CRUD funkce, slouží k editaci informací o funkci a jejího názvu.

Pravý panel slouží k ediatci informací o funkci. Tato obrazovka je součástí aktivity „Editace CRUD matice“ 3.3.1.

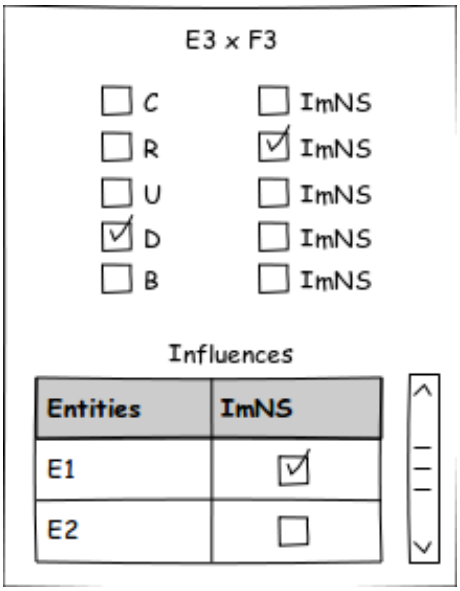
3.4.3.3 Pravý panel CRUD entity



Obrázek 3.13: Návrh pravého panelu CRUD entity, slouží k editaci informací o entitě a jejího názvu.

Pravý panel slouží k ediatci informací o funkci. Tato obrazovka je součástí aktivity „Editace CRUD matice“ 3.3.1.

3.4.3.4 Pravý panel CRUD akce



Obrázek 3.14: Návrh pravého panelu CRUD akce, slouží k editaci informací o entitě a jejího názvu.

Pravý panel slouží k editaci příznaků. Tato obrazovka je součástí aktivity „Editace CRUD matice“ 3.3.1.

3.4.4 Generování scénářů

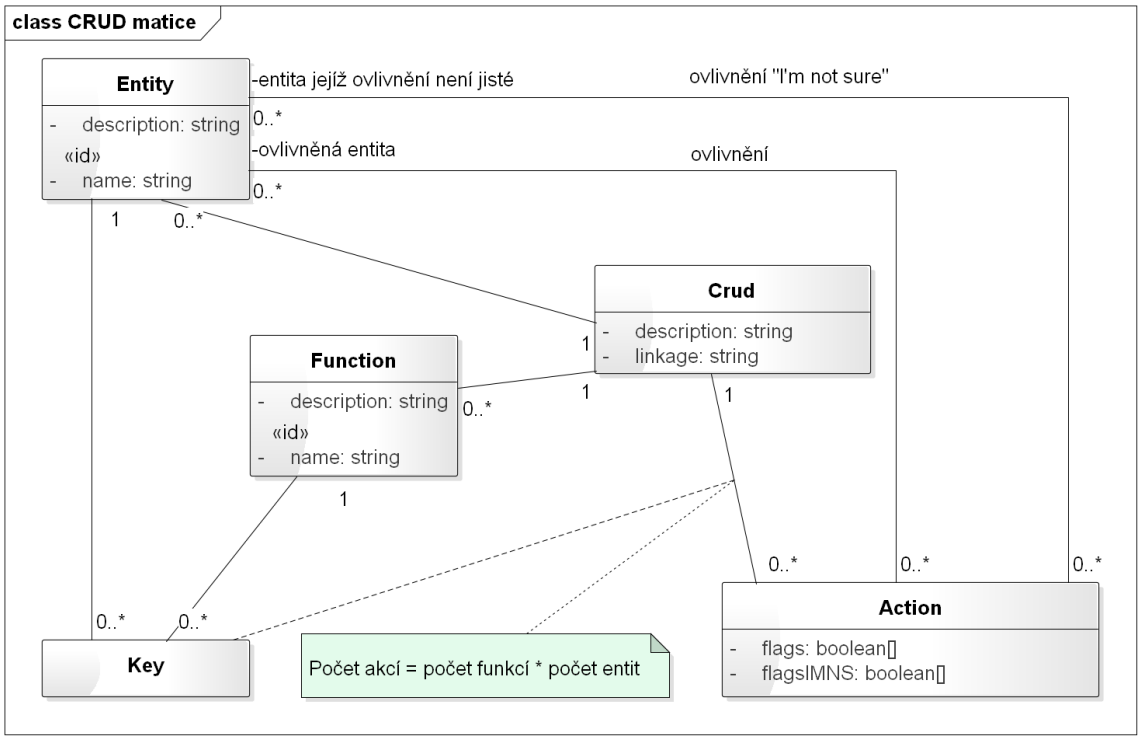
Obrázek 3.15: Návrh toolbaru matice sloužící ke generování scénářů a kontrole konzistence.

Toolbar slouží ke generování scénářů a kontrole konzistence. Tato obrazovka je součástí aktivity „Generování scénářů“ 3.3.2. Funkci tlačítek není třeba komentovat.

3.5 Datový model

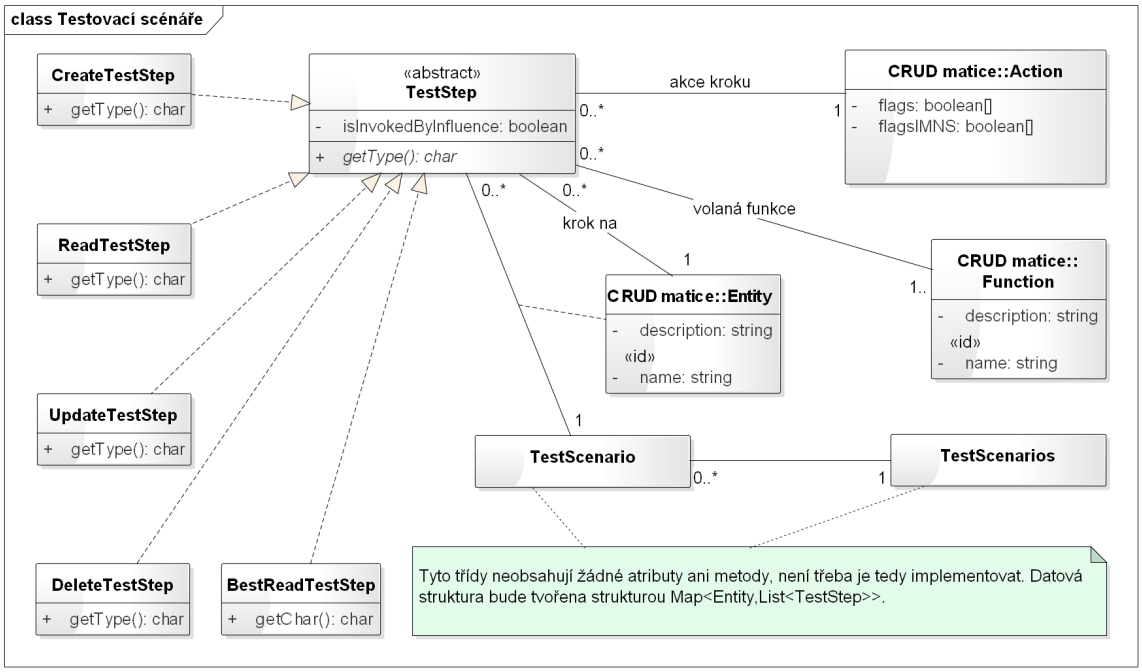
Datový model bude sloužit jako podklad při implementaci modelových tříd modulu.

3.5.1 CRUD matice



Obrázek 3.16: Diagram datové struktury CRUD matice

3.5.2 Testovací scénáře



Obrázek 3.17: Diagram datové struktury testovacích scénářů

3.6 Formát exportovaných dat

Jelikož exportovaná data budou importována zpět do systému(kromě HTML), musejí mít jasně definovanou struktutu. Tuto strukturu objasním následujícími schématy.

3.6.1 XML

crud:node (při exportu root)

- linkage:attribute
- version:attribute
- longenum:textNode
- entities:node
  - entity:node (multiple)
    - entity_id:id
    - name:textNode
    - longenum:textNode
- functions:node
  - function:node (multiple)

        - function_id:id
        - name:textNode
        - longenum:textNode

- actions:node
    - action:node (multiple)
        - function_id_ref:idRef
        - entity_id_ref:idRef
        - create:emptyNode
            - is_set:attribute
            - imns:attribute
        - read:emptyNode
            - is_set:attribute
            - imns:attribute
        - update:emptyNode
            - is_set:attribute
            - imns:attribute
        - delete:emptyNode
            - is_set:attribute
            - imns:attribute
        - best_read:emptyNode
            - is_set:attribute
            - imns:attribute
        - influences:node
            - influence:emptyNode (multiple)
                - entity_id_ref:idRef
                - is_set:attribute
                - imns:attribute

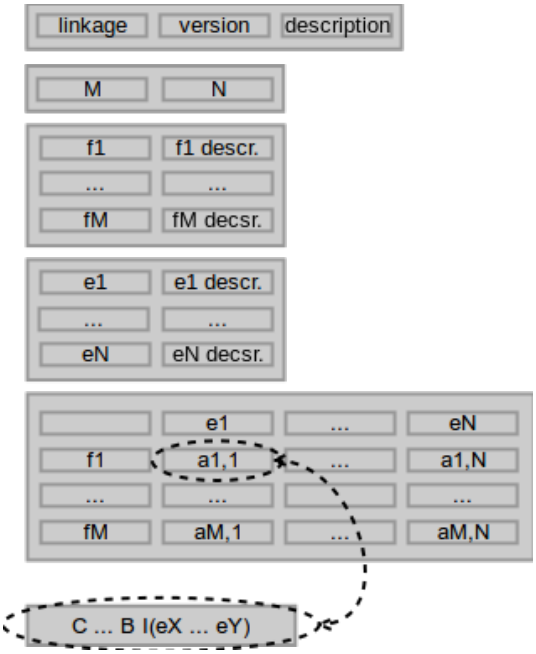
Více v souboru *crud.dtd* na přiloženém CD.

Formát určení výhradné k uložení stavu matice a jako součást dat při ukládání celého projektu.

3.6.2 CSV

Tento formát je navržen pro výměnu tabulkových dat.

3.6.2.1 Visual CSV



Obrázek 3.18: Schéma datové struktury vizuelního CSV

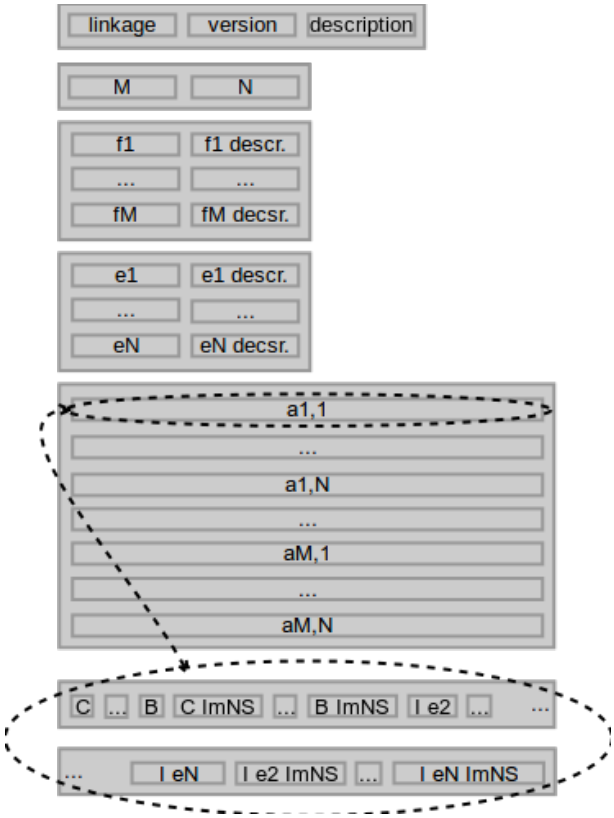
Formát je zamýšlen k editaci v běžných tabulkových editorech. Je tedy požadována snadná čitelnost pro člověka. Formát tedy kopíruje vzhled tabulky a barevné rozlišení příznaku *I'm not sure* je v tomto případě nedosažitelné. Tento formát je tedy vůči tomuto příznaku ztrátový.

Další featurou tohoto formátu je uchování informace o ovlivnění. Tato informace je sice uchovávána, ale za jistých okolností může dojít k poškození informace. Toto demonstruji následujícím příkladem:

Pokud *abc*, *def* a *abc def* jsou entity, potom z příznaku *I(abc def)* není rozlišitelné která entita/y je v příznaku myšlena.

Tuto vlastnost musí uživatel hlídat. Dále je nutno zmínit, že při přejmenování funkce/entity, musí uživatel změnit název jak v příslušném seznamu, tak v záhlaví matice. V opačném případě není zaručeno úspěšné importování matice.

3.6.2.2 Technical

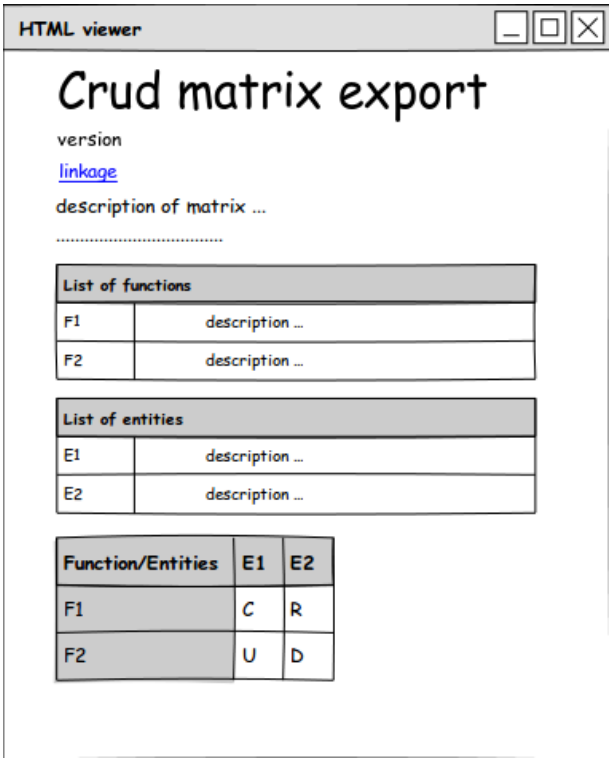


Obrázek 3.19: Schéma datové struktury technického CSV

Formát slouží pouze k uložení matice. Jsou uložena všechna data, ovšem za cenu snadné čitelnosti. Pomocí znalosti schématu je sice možno s daty pracovat, ovšem takováto práce není předpokládána.

Formát akce jak vidíte v detailu na obrázku 3.19 je tvořen řádkem boolean hodnot v pořadí: hodnoty *C,R,U,D,B*, hodnoty *I'm not sure* těchto příznaků, hodnoty *I* jednotlivých entit v pořadí odpovídajícímu jejich výpisu (entita vztahující se k aktuální akci je vynechána, je zde tedy *N - 1* hodnot), nakonec přijdě obdobný seznam příznaků *I - I'm not sure*.

3.6.3 HTML



Obrázek 3.20: Návrh výsledného html

Tento formát není sice požadován, ale bude implementován na základě konzultace se zadavatelem. Výsledný HTML kód bude produkovat stránku podobnou té na obrázku 3.20.

3.7 Postupy pro generování testovacích scénářů

Algoritmus 1 Algoritmus generování scénářů

procedure GENERATETESTSCENARIOS
Input: CRUD matice M
Output: mapa(K,V) *scenarios* kdy K je entita $\in E$ a v seznam funkcí $\subseteq F$, kde E jsou všechny entity a F všechny funkce CRUD matice

```
E ⇐ množina všech entit ∈ M
F ⇐ množina všech funkcí ∈ M
A ⇐ množina všech akcí ∈ M
allC je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
allR je mapa(K,V) kdy K je entita ∈ E a V je množina akcí ⊆ A ▷ Mapa je prozatím prázdná
allU je mapa(K,V) kdy K je entita ∈ E a V je množina akcí ⊆ A ▷ Mapa je prozatím prázdná
allD je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
allB je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
[allC, allR, allU, allD, allB] ⇐ processCrud(M)
for ∀e ∈ E do
    scenario je seznam funkcí ⊆ F                    ▷ Seznam je prozatím prázdný
    c ⇐ akce a z allC uložená pod klíčem e
    f ⇐ funkce která vyvolala akci c
    scenario ⇐ f
    R ⇐ množina akcí a z allR uložených pod klíčem e
    b ⇐ akce a z allB uložená pod klíčem e
    scenario ⇐ seznam funkcí které vyvolaly akce z O ⇐ handleReads(R, b)
    if hloubka pokrytí zohledňuje ovlivnění ∧c ovlivňuje entity then
        EI ⇐ množina entit ovlivněných entitou e
        for ∀ei ∈ EI do
            RI ⇐ množina akcí z allR uložená pod klíčem ei
            bi ⇐ akce z allB uložená pod klíčem ei
            scenario ⇐ funkce která vyvolala akci a ⇐ handleInfluences(RI, bi)
        end for
    end if
    U ⇐ množina akcí z allU uložená pod klíčem e
    for ∀u ∈ U do
        f ⇐ funkce která vyvolala akci u
        scenario ⇐ f
        scenario ⇐ seznam funkcí které vyvolaly akce z O ⇐ handleReads(R, b)
        if hloubka pokrytí zohledňuje ovlivnění ∧u ovlivňuje entity then
            EI ⇐ množina entit ovlivněných entitou e
            for ∀ei ∈ EI do
                RI ⇐ množina akcí z allR uložená pod klíčem ei
                bi ⇐ akce z allB uložená pod klíčem ei
                scenario ⇐ funkce která vyvolala akci a ⇐ handleInfluences(RI, bi)
            end for
        end if
    end for
```

```
        d ⇐ akce a z allD uložená pod klíčem e
        f ⇐ funkce která vyvolala akci d
        scenario ⇐ f
        scenario ⇐ seznam funkcí které vyvolaly akce z O ⇐ handleReads(R, b)
        if hloubka pokrytí zohledňuje ovlivnění ∧d ovlivňuje entity then
            EI ⇐ množina entit ovlivněných entitou e
            for ∀ei ∈ EI do
                RI ⇐ množina akcí z allR uložená pod klíčem ei
                bi ⇐ akce z allB uložená pod klíčem ei
                scenario ⇐ funkce která vyvolala akci a ⇐ handleInfluences(RI, bi)
            end for
        end if
    end for
end procedure
```

Algoritmus 2 Algoritmus zpracování CRUDu pro generování scénářů

procedure ProcessCrud
Input: CRUD matice M
Output: $allC$ je mapa(K,V) kdy K je entita $\in E$ a V je akce $\in A$, $allR$, $allU$ jsou mapy(K,V) kdy K je entita $\in E$ a V je množina akcí $\subseteq A$, $allD$, $allB$ jsou mapy(K,V) kdy K je entita $\in E$ a V je akce $\in A$, kde E jsou všechny entity a A všechny akce z CRUD matice

```
E ⇐ množina všech entit ∈ M
F ⇐ množina všech funkcí ∈ M
A ⇐ množina všech akcí ∈ M
allC je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
allR je mapa(K,V) kdy K je entita ∈ E a V je množina akcí ⊆ A ▷ Mapa je prozatím prázdná
allU je mapa(K,V) kdy K je entita ∈ E a V je množina akcí ⊆ A ▷ Mapa je prozatím prázdná
allD je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
allB je mapa(K,V) kdy K je entita ∈ E a V je akce ∈ A ▷ Mapa je prozatím prázdná
for ∀e ∈ E do
    c je akce ∈ A                          ▷ Akce je prozatím prázdná
    R je množina akcí ⊂ A                  ▷ Množina je prozatím prázdná
    U je množina akcí ⊂ A                  ▷ Množina je prozatím prázdná
    d je akce ∈ A                          ▷ Akce je prozatím prázdná
    b je akce ∈ A                          ▷ Akce je prozatím prázdná
    for ∀f ∈ F do
        a ⇐ akce vyvolaná funkcí f na entitě e
        if a má příznak C then
            c ⇐ a
        end if
        if a má příznak R then
            R ⇐ a
        end if
        if a má příznak U then
            U ⇐ a
        end if
        if a má příznak D then
            d ⇐ a
        end if
        if a má příznak B then
            b ⇐ a
        end if
    end for
    allC ⇐ (e, c)
    allR ⇐ (e, R)
    allU ⇐ (e, U)
    allD ⇐ (e, d)
    allB ⇐ (e, b)
end for
end procedure
```

Algoritmus 3 Algoritmus zpracování čtení

procedure HANDLEREADS
Input: množina akcí R s příznakem R, akce b s příznakem B
Output: množina akcí $M \subset R \cup \{b\}$
M $\Leftarrow$ seznam akcí s příznakem R nebo B vygenerovaných na základě požadované úrovně pokrytí.
end procedure

Algoritmus 4 Algoritmus zpracování ovlivnění

procedure HANDLEINFLUENCES
Input: list akcí R s příznakem R, akce b s příznakem B
Output: akce a s příznakem R nebo B
if $b \neq null$ **then**
$a \Leftarrow b$
else
$a \Leftarrow Random(R)$
end if
end procedure

Kapitola 4

Implementace

V této kapitole seznámím čtenáře s průběhem implementace, potížemi, které při implementaci vyvstaly a následným řešením. Dále přiblížím struktutu výsledného řešení.

Při implementaci budu využívat dokumentace [2] dostupné na stránkách společnosti Oracle.

4.1 Použité technologie

4.1.1 IDE

Jako IDE jsme zvolil InteliJ IDEU 13.0 jelikož jsem s ní byl zvyklý pracovat.

4.1.2 JDK

Bylo vyvíjeno s JDK SE 7. Bylo zvoleno na základě požadavků zadavatele.

4.1.3 IDE

Jako IDE jsme zvolil InteliJ IDEU 13.0 jelikož jsem s ní byl zvyklý pracovat.

4.1.4 Testování

K testování(unit a integration) byl vybrán framework JUnit4. Pro UAT testování, respektive generování jeho scénářu byl použit PCTgen, modul pro generovaní process cycle testů. Více se dozvíte v kapitole Testování 5.

4.2 Postup a refaktoring

Jako první jsem implementoval model matice. Po dokončení modelu následovala jeho integrace do systému. Integrováno bylo jak na úrovni GUI, kdy byly implementovány jednotlivé komponenty view, tak rozšířením komponent stávajících. Tyto úpravy probíhaly tedy paralelně. Rozšířeny byly třídy *MainGui, MainGuiPresenter, ProjectPrsenter, RightPanels* a *ProjectTreeNode*. Tyto třídy jsem upravil přidáním vlastních metod, ale i rozšířením metod stávajících. Úprava těchto tříd nebo alespoň jejich podmnožiny, byla nadále součástí každého rozšíření a úprav.

Následovalo lazení odvedené práce. Zde jsem musel předělat view matice z třídy *JTable* na třídu *JScrollPane*, kdy tato třída posloužila jako wrapper pro dvě tabulky třídy *JTable*, toto řešení jsem dohledal na webu [3]. Toto bylo vykonáné z nutnosti rozdělení matice do dvou tabulek z důvodu implementace tabulky s horizontálním i vertikálním záhlavím, kdy tato záhlaví budou držet svou pozici při scrollováni v obou směrech. Dále jsem provedl řadu dalších drobných změn spojených s laděním kódu.

Když byl editor matice hotov, pokračoval jsem dalšími požadavky. Prvním z nich bylo generování testovacích scénářů. Tyto generátory jsou potomky abstraktní třídy *DataConsistencyTestGenerator*. Vzhledem k faktu, že všechny scénáře jsou tvořeny posloupností operací *C, U, D*, lišící se pouze sekvencí operacíR, popřípadě B a zohledněním operace I, bylo vhodné

využít návrhového vzoru *Template method* viz [7]. Dále byla zhotovena utilita pro test konzistence mezi CRUD matici a grafy. Toto rozšíření si opět vyžadovalo integraci do stávajícího kódu.

V poslední fázi jsem systém rozšířil o funkcionality spojené s ukládáním/obnovením dat matice a automatickým ukládáním dat. Pro ukládání/obnovení dat jsem vytvořil knihovní třídu *CrudUtils* do které byly přesunuty i některé pomocné metody z třídy *CrudModel* a pro ukládání celého projektu byla rozšířena třída *XML*. V této fázi jsme se zadavatelem zjistili že ukládání do formátu CSV bylo slabě specifikováno a podoba výsledného formátu neodpovídala skutečným potřebám. Byla tedy provedena úprava/dolazení požadavků a následný refaktoring kódu. Druhý požadavek této fáze a celkově poslední, automatické aktualizace. Toto rozšíření vedlo k vytvoření nových tříd, potomků třídy *ListenableAbstractRightPanel*, jež je potomkem původní třídy *AbstractRightPanel*. Analogicky k tomuto bylo vytvořeno nové rozhraní *RightPanelListeningItem*, potomek rozhraní *RightPanelItem*. Dědění zde bylo potřebné pouze z důvodu komptibility těchto tříd na volání metody *changeRightPanel* třídy *MainGui*, jelikož původní metody zůstaly, až na vyjímky, prázdné. Využití stále jedné metody k téže funkci s drobnou úpravou jejího těla, se mi jeví jako vhodnější, než vytvářet druhou metodu za stejným účelem. Vše jsem zakončil závěrečným lazením drobných nedostatků a výsledků testování.

4.3 Struktura kódu

V této sekci uvedu čtenáře do struktury výsledného kódu, což poslouží k lepší orientaci při jeho prohlížení v přiloženém projektu na CD.

4.3.1 Základní balíčky

Veškerá má práce, krom drobných úprav ve stávajícím kódu, se nachází v balíčku **dctmodule**.

autoupdating balíček obsahující rozhraní pro automatickou aktualizaci editovaných dat

exceptions balíček s vyjímkami zobrazitelných v GUI

gui balíček obsahující komponenty grafického rozhraní

- **dialogs** balík s dialogovými okny modulu
- **mainview** grafické komponenty hlavní pracovní plochy
- **projecttreenode** balíček obsahující komponenty projektového stromu
- **rightpanels** balíček obsahující komponenty pravého panelu

listeners listenery matice(a entity) a pravého panelu

messages zprávy ktere si systém/modul při práci vyměňuje

model balíček obsahující model CRUD matice a testovacích scénářů

utils balíček obsahující utility potřebné k fungování modulu

testing balík obsahující generátory scénářů

validators balík obsahující validátory

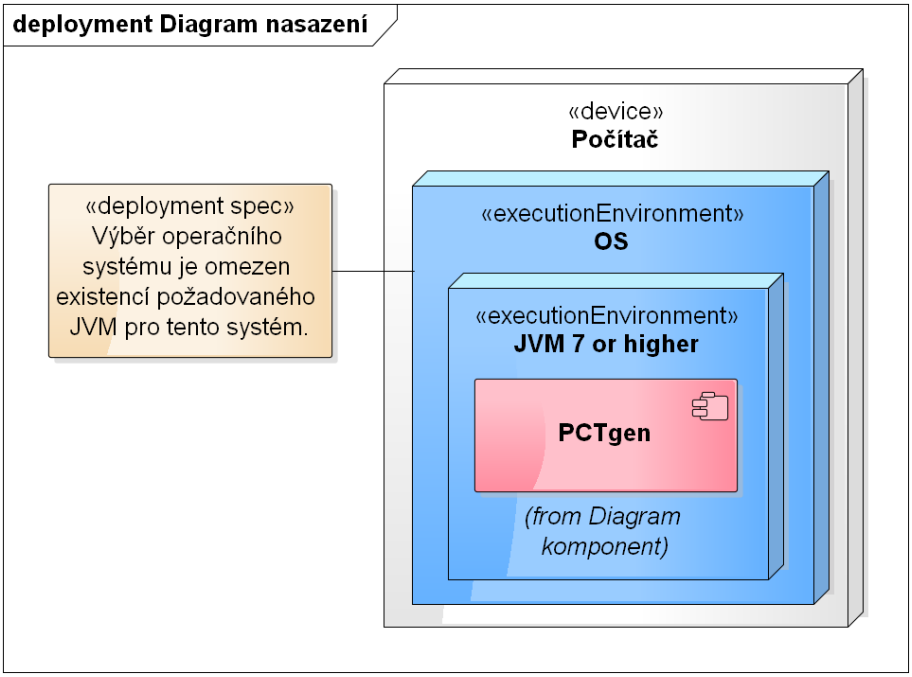
4.3.2 Diagram komponent

Tento diagram zachycuje strukturu komponent systému.

Obrázek 4.1: Diagram komponent

4.3.3 Diagram nasazení

Tento diagram zachycuje informaci o nasazení systému.



Obrázek 4.2: Diagram nasazení

4.3.4 Základní třídy

- Rozhraní

  Crud Rozhraní reprezentující matici.

  CrudEntity Rozhraní reprezentující entitu.

  CrudFunction Rozhraní reprezentující funkci.

  CrudAction Rozhraní reprezentující akci vykonanou funkcí na entitě.

  RightPanelListeningItem Rozhraní pro vložení položky do pravého panelu ListenableAbstractRightPanel. Toto rozhraní rozšiřuje původni rozhraní RightPanelItem o možnost automatické aktualizace.

- Abstraktní třídy

  ListenableAbstractRightPanel Třída reprezentující pravý panel. Rozšiřuje původní AbstractRightPanel o schopnost automatické aktualizace spravované položky.

  DataConsistencyTestGenerator Třída definující postup při genrování scénářů, specifika přenechává svým potomkům.

  TestStep Krok testovacího scénáře, specifikaci kroku přenechává svým potomkům.

- Třídy

  CrudModel modelová třída implementující Crud

EntityCell modelová třída implementující CrudEntity a RightPanelListeningItem.

FunctionCell modelová třída implementující CrudFunction a RightPanelListeningItem.

ActionCell modelová třída implementující CrudAction a RightPanelListeningItem.

CrudTable view třída zobrazující CRUD matici a umožňující její editaci(CRUD componenta, např CrudFunction, musí implementovat RightPanelListeningItem)

ListTable view třída pro zobrazení seznamů entit a funkcí

ZeroRGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí 0R

ZeroBGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí 0B

OneRGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí 1R

OneRInfluencesRGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí 1RI

NRGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí NR

NRInfluencesGenerator třída specifikující DataConsistencyTestGenerator generující scénáře pokrytí NRI

CrudUtils knihovní třída s řadou užitečných funkcí pro práci s CRUD maticí.

CreateTestStep třída specifikující TestStep reprezentující C operaci

ReadTestStep třída specifikující TestStep reprezentující R operaci

UpdateTestStep třída specifikující TestStep reprezentující U operaci

DeleteTestStep třída specifikující TestStep reprezentující D operaci

BestReadTestStep třída specifikující TestStep reprezentující B operaci

CrudTreeNode třída reprezentující CRUD matici v projektovém stromu, implementuje RightPanelListeningItem

DCTTestCaseTreeNode třída reprezentující vygenerované scénáře v projektovém stromu

FunctionsTreeNode třída reprezentující seznam funkcí v projektovém stromu

EntitiesTreeNode třída reprezentující seznam entit v projektovém stromu

4.3.5 Utility

Zde jsou uvedeny navržené postupy tvořící specifické funkcionality.

4.3.5.1 Persistence a obnovení dat

Tyto funkcionality jsou přístupné skrze knihovní třídu CrudUtils.

K uložení a načtení dat v XML(a HTML) formátu bylo použito JAVA API. Díky Document Object Model (DOM) byl dokument snadno vygenerován. V případě CSV byla práce o něco komplikovanější, k parsování CSV dat bylo potřeba naimplementovat vlastní parser. Vzhledem ke komplikovanosti čtení a nepřehlednosti kódu způsobené větvením bylo použito návrhového vzoru *State*.

Při implementaci/úpravě metody určené k načtení uloženého projektu muselo být dbáno na zpětnou kompatibilitu. Projekty vytvořené před implementací modulu logicky neobsahovaly žádnou CRUD matici a toto musí systém umět řešit. V takovém případě systém reaguje vložením prázdné matice.

4.3.5.2 Generování scénářů

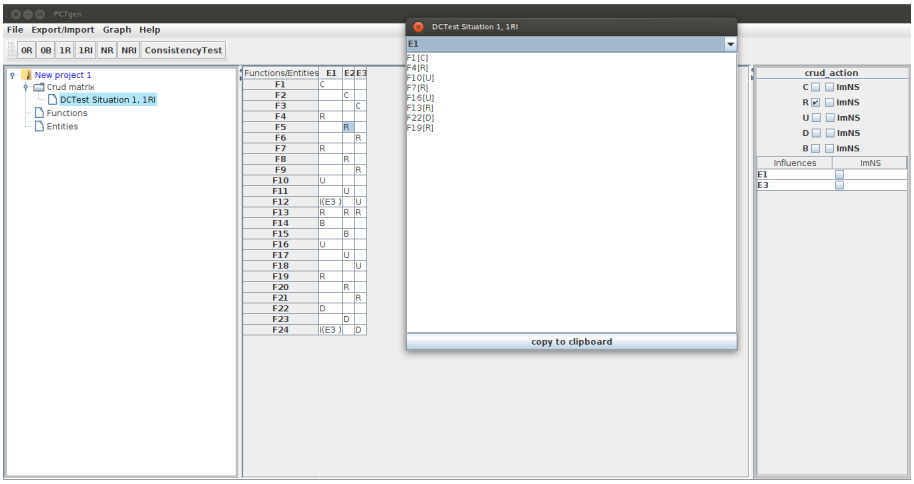
Generátory scénářů jsou tvořeny jednotlivými třídami dle úrovně pokrytí, tyto třdy se nacházejí v balíku *dctmodule.utils.testing*.

Jak již bylo zmíněno, generoané scénáře mají stejnou základní kostru, liší se především důsledností při čtení dat a jejich ovlivněním v některých případech. K řešení tohoto problému se dokonale hodil návrhový vzor *Template method*. Tyto generátory jsou v balíku dctmodule.utils

4.3.6 Integrace na existující systém PCTgen

Pro modul byl vytvořen samostatný balík *dctmodule*. Tento balík obsahuje veškeré náležitosti nutné pro fungování modulu. K začlenění tohoto modulu do systému bylu upraveny metody v třídách *MainGui, MainGuiPresenter, ProjectPrsenter, RightPanels* a *ProjectTreeNode* či byly přidány metody nové.

4.3.7 Ukázky aplikace



Obrázek 4.3: Ukázka hotového systému, zobrazuje všechny stěžejní části modulu.

Na obrázku 4.3 můžeme vidět všechny ztěžejní prvky systému/modulu. Uprostřed obrazovky se nachází CRUD matice a vedle ní, na jejím základě vygenerovaný, testovací scénář. Tento scénář byl vygenerován po kliknutí na tlačítko *1RI*, které se nachází v toolbaru vlevo od matice, těsně nad její úrovní. Pod tímto toolbarem se nachází projektový strom, ve kterém můžeme spatřit položky vytvořené v rámci této práce. Na protější straně okna se nachází pravý panel, který je nyní v módu úpravy akce, kde je možno nastavovat potřebné příznaky.

Kapitola 5

Testování

5.1 Jednotkové a integrační testy

K otestování těchto okruhů byl použit framework JUnit4. V rámci těchto testů byl testován model CRUD matice a generované scénáře. Tyto testy jsou k dispozici v projektu PCTgen na přiloženém CD.

Tyto testy byli používány při implementaci k ověření správného fungování daných komponent a jejich integrace. V rámci projektu jsou tyto testy umístěny v samostatném balíku **tests**.

5.1.1 Jednotokové testy

V rámci jednotkových testů byly otestovány všechny modelové třídy, tedy třídy CRUD matice a třídy testovacích scénářů. Tyto testy se nacházejí v balíčku **tests.unit**.

5.1.2 Integrační testy

Integrační testy se nacházejí v balíčku **tests.integration**. Pomocí integračních testů byly testovány výhradně vlasnosti sytému plynoucí z funkčních požadavků. Byly tedy testovány vybrané metody třídy CrudUtils pro import a export dat CRUD matice, dále byly testovány jednotlivé generátory testovacích scénářů a třída ConsistencyTest pro test konzistence mezi CRUD maticí a grafy. V poslední řadě byla testována vlastnost automatické aktualizace položek spravovaných pravými panely.

5.1.2.1 Výsledek jednotkových a akceptačních testů

Tyto testy v průběhu implementace objevovaly řadu implementačních nedostatků a chyb. Tyto chyby byly vždy následně opraveny.

5.2 Uživatelské akceptační testy (UAT)

Tyto testy mají za úkol demonstrovat a otestovat dosažené výsledky. Testy jsou tvořeny jednotlivými step-by-step scénáři, pomocí níchž se má ověřit konkrétní funkcionalita. Kladné vyhodnocení těchto testů je důležité k akceptování dodaného produktu.

5.2.1 Scénáře UAT

5.2.1.1 Výsledek UAT testů

Testy objevily řadu chyb, které byly následně opraveny. Nejčastějším problémem byla špatná manipulace s grafickými komponentami spojená s jejich překreslováním při změně obsahu. Další častou chybou byly problémy spojené s automatickou aktualizací. V tomto případě se pravé panely pokoušely aktualizovat španou komponentu, ale i tento defekt byl úspěšně odstraněn. Posledním okruhem chyb byly chyby estetického charakteru. Tyto chyby sice nebránily užívaní systému, ale byly nepříjemnou vadou na kráse. Tyto chyby byly odstraněny změnou nastavení grafických komponent.

ID	1
Název	Vytvoření CRUD matice spolu se založením projektu
Vstupní kritéria	-
Kroky	1. Založ projekt
Očekávané výsledky	• Pod novým projektem ve stromové struktuře je položka CRUD matice.
Prošel	ANO

Tabulka 5.1: UAT, Vytvoření CRUD matice spolu se založením projektu

ID	2
Název	Přidání funkce do CRUD matice
Vstupní kritéria	• Je založen projekt
Kroky	1. Vyber matici k editaci 2. Přidej do ní funkci
Očekávané výsledky	• V matici přibyla položka funkce
Prošel	ANO

Tabulka 5.2: UAT, Přidání funkce do CRUD matice

ID	3
Název	Přidání entity do CRUD matice
Vstupní kritéria	• Je založen projekt
Kroky	1. Vyber matici k editaci 2. Přidej do ní entitu
Očekávané výsledky	• V matici přibyla položka entity
Prošel	ANO

Tabulka 5.3: UAT, Přidání entity do CRUD matice

ID	4
Název	Editace funkce v CRUD matici
Vstupní kritéria	• Je založen projekt • Tento projekt obsahuje alespoň jednu funkci
Kroky	1. Vyber matici k editaci 2. Vyber funkci 3. Změň název 4. Změň popis 5. Přidej entitu 6. Opět vyber tutéž funkci
Očekávané výsledky	• Při přejmenování se data promítla ihned do matice(pokud by byl název duplicitní k přejmenování by nedošlo) • Po opětovné editaci funkce byl zobrazen již změněný popis
Prošel	ANO

Tabulka 5.4: UAT, Editace funkce v CRUD matici

ID	5
Název	Editace entity v CRUD matici
Vstupní kritéria	• Je založen projekt • Tento projekt obsahuje alespoň jednu entitu
Kroky	1. Vyber matici k editaci 2. Vyber entitu 3. Změň název 4. Změň popis 5. Přidej funkci 6. Opět vyber tutéž entitu
Očekávané výsledky	• Při přejmenování se data promítla ihned do matice(pokud by byl název duplicitní k přejmenování by nedošlo) • Po opětovné editaci entity byl zobrazen již změněný popis
Prošel	ANO

Tabulka 5.5: UAT, Editace entity v CRUD matici

ID	6
Název	Editace funkce v CRUD matici
Vstupní kritéria	• Je založen projekt • Tento projekt obsahuje alespoň jednu funkci
Kroky	1. Vyber matici k editaci 2. Smaž funkci
Očekávané výsledky	• Z matice zmizela smazaná funkce
Prošel	ANO

Tabulka 5.6: UAT, Editace funkce v CRUD matici

ID	7
Název	Smazání entity v CRUD matici
Vstupní kritéria	• Je založen projekt • Tento projekt obsahuje alespoň jednu entitu
Kroky	1. Vyber matici k editaci 2. Smaž entitu
Očekávané výsledky	• Z matice zmizela smazaná entita
Prošel	ANO

Tabulka 5.7: UAT, Smazání entity v CRUD matici

ID	8
Název	Editace akce v CRUD matici
Vstupní kritéria	• Je založen projekt • Tento projekt obsahuje alespoň jednu entitu a jednu funkci
Kroky	1. Vyber matici k editaci 2. Vyber akci 3. Nastav příznak/y
Očekávané výsledky	• V matici se promítly nastavené příznaky
Prošel	ANO

Tabulka 5.8: UAT, Editace akce v CRUD matici

ID	9
Název	Import/Export CRUD matice v XML
Vstupní kritéria	• Je založen projekt • CRUD matice má nastavena popisná pole • CRUD matice tohoto projektu má nenulový počet funkcí a entit • CRUD matice má nastaveny příznaky, každý příznak alespoň jednou
Kroky	1. Vyber matici 2. Exportuj matici v XML 3. Založ nový projekt 4. Do tohoto projektu importuj CRUD matici z kroku 2
Očekávané výsledky	• Importovaná matice je totožná s exportovanou
Prošel	ANO

Tabulka 5.9: UAT, Import/Export CRUD matice v XML

ID	10
Název	Import/Export CRUD matice v CSV
Vstupní kritéria	• Je založen projekt • CRUD matice má nastavena popisná pole • CRUD matice tohoto projektu má nenulový počet funkcí a entit • CRUD matice má nastaveny příznaky, každý příznak alespoň jednou
Kroky	1. Vyber matici 2. Exportuj matici v CSV 3. Založ nový projekt 4. Do tohoto projektu importuj CRUD matici z kroku 2
Očekávané výsledky	• Importovaná matice je totožná s exportovanou
Prošel	ANO

Tabulka 5.10: UAT, Import/Export CRUD matice v CSV

ID	11
Název	Import/Export CRUD matice v CSV
Vstupní kritéria	• Je založen projekt • Tento projekt má neprázdnou CRUD matici
Kroky	1. Ulož projekt 2. Otevři uložený projekt
Očekávané výsledky	• Nově otevřený projekt je toto
Prošel	ANO

Tabulka 5.11: UAT, Import/Export CRUD matice v CSV

ID	12
Název	Generování scénářů
Vstupní kritéria	• Je založen projekt • CRUD matice tohoto projektu je ve tvaru, kdy z ní je možno generovat scénáře.
Kroky	1. Vyber CRUD matici 2. Vygeneruj libovolný scénář
Očekávané výsledky	• Vygenerovaný scénář je korektní dle definice (záleží na zvolené úrovni pokrytí)
Prošel	ANO

Tabulka 5.12: UAT, Generování scénářů

ID	13
Název	Kontrola konzistence CRUD matice oproti grafům
Vstupní kritéria	• Je založen projekt • Tento projekt má neprázdnou CRUD matici • Tento projekt má nenulový počet neprázdných grafů
Kroky	1. Vyber CRUD matici 2. Zkontroluj konzistenci
Očekávané výsledky	• Pokud je funkce F1 součástí matice, ale nevyskytuje se v žádném grafu, je to reportováno • Pokud je funkce F2 součástí alespoň jednoho grafu, ale v CRUD matici chybí, je toto reportováno (i s výčtem grafů které ji obsahují).
Prošel	ANO

Tabulka 5.13: UAT, Kontrola konzistence CRUD matice oproti grafům

Kapitola 6

Závěr

Tato práce pro mne byla velkým přínosem. Byť jsem se během studia podílel na několika projektech, samostatná práce na celém projektu je něco zcela jiného. Velkým přínosem pro mne bylo také rozšiřování cizího kódu. Toto pro mne, alespoň v takovýchto rozměrech, bylo něco zcela nového a utvrdilo mě to v otázce potřeby dokumentace.

Dále jsem se utvrdil v tom, že analýza a návrh jsou velmi důležité části vedení projektu, neboť bez těchto procesů by samotná implementace kódu byla plná refaktoringu a tedy velmi zdlouhavá. Nějakému tomu refaktoringu se ale člověk nevyhne, a zde je znatelné jak důležité je dodržování pravidel objektového programování. Dodržování těchto pravidel tento proces činí snadějším oproti zbytečně provázanému kódu.

Při implementaci kódu proběhly celkem 4 zásadní refaktoringy. Prvním z nich byla refaktorizace view CRUD matice, zde bylo původně implementováno pomocí třídy JTable. Toto řešení bylo nevhodné, jelikož neumožňovalo implementovat druhé svislé záhlaví pro funkce. Toto bylo vyřešenou třídou JScrollPane která posloužila jako obal pro celkem dvě JTable. Tato změna si vyžádala úpravu modelu matice a vytvoření dvou dekorátorů tvořících modely dvou nových JTable uvnitř JScrollPane.

Další refaktorizací bylo zautomatizování aktualizace editovaných dat, tento refaktoring nebyl důsledkem špatné přípravy, nýbrž byl plánovaný. Špatným bych shledal tento plán refaktorizace, jelikož to byla zbytečná práce navíc. Chybou bych to ovšem nenazval, spíše dobrou zkušeností.

Jako třetí byl refaktorován výsledný formát exportovaného CSV, jelikož účel formátu nebyl správně pochopen. Nejedná se přímo o refaktoring, ale o rozšíření, jelikož původní formát byl zachován a přidán další, což vyústilo ve dva CSV formáty. Toto pochybení shledávám jako následek vynechání User Stries při analýze, kdy by otázka „proč“ tomuto zabránila.

Posledním refaktoringem byla úprava generátoru pokrytí 1R, respektive 1RI. Tato pokrytí mají, dle definice, vyčerpat všechny R operace. Původně tyto generátory generovaly obyčejné 0R pokrytí, s tím že nakonec, po operaci D, dopoužily zbylé R operace. Toto řešení nebylo sice v rozporu s algoritmem z definice, avšak s logikou využití všech R operací ano. Manipulace s R operacemi byla opravena a nyní generátory využívají R operace rovnoměrně po každé z C,U,D operací.

S výsledným produktem jsem spokojen a věřím že nalezne celou řadu uživatelů, kterým usnadní jejich práci a pomůže odhalit řadu chyb.

Literatura

[1] Common Format and MIME Type for Comma-Separated Values (CSV) Files. Dostupné z: <https://tools.ietf.org/html/rfc4180>. , stav ze 23. 5. 2016.

[2] JAVA 7 dokumentace. Dostupné z: <http://docs.oracle.com/javase/7/docs/>. , stav ze 23. 5. 2016.

[3] RowNumberTable. Dostupné z: <https://tips4java.wordpress.com/2008/11/18/row-number-table/>. , stav ze 23. 5. 2016.

[4] Extensible Markup Language (XML). Dostupné z: <https://www.w3.org/XML/>. , stav ze 23. 5. 2016.

[5] BUREš, M. Techniky pro test datové konzistence. Katedra počítačů při ČVUT v Praze, nepublikováno.

[6] KOOMEN, T. *TMap next, for result driven testing.* Utrecht: UTN, 2006. ISBN 90-72194-80-2.

[7] PECINOVSKÝ, R. *Návrhové vzory: [33 vzorových postupů pro objektové programování].* Brno: Computer Press, 2007. ISBN 978-80-251-1582-4.

[8] RAY, E. T. *Learning XML.* O'Reilly Media, 1st edition, 2001. ISBN 0-59600-046-4.

[9] SOMMERVILLE, I. *Softwarové inženýrství.* Brno: Computer Press, 1. vydání edition, 2013. ISBN 978-80-251-3826-7.

Příloha A

Zkratky

0R Simple read

0B Simple best read

1R All reads one time

1RI All reads one time, I operations reflected

API Application Programming Interface

CRUD Create, read, update, delete

CSV Comma-separated values

DCyT Data cycle test

DOM Document object model

DTD Data definition document

EDCyT Extended data cycle test

GUI Graphic User Interface

HTML HyperText Markup Language

IDE Integrated Development Environment

JDK Java Development Kit

JVM Java Virtual Machine

NR All reads after all changes

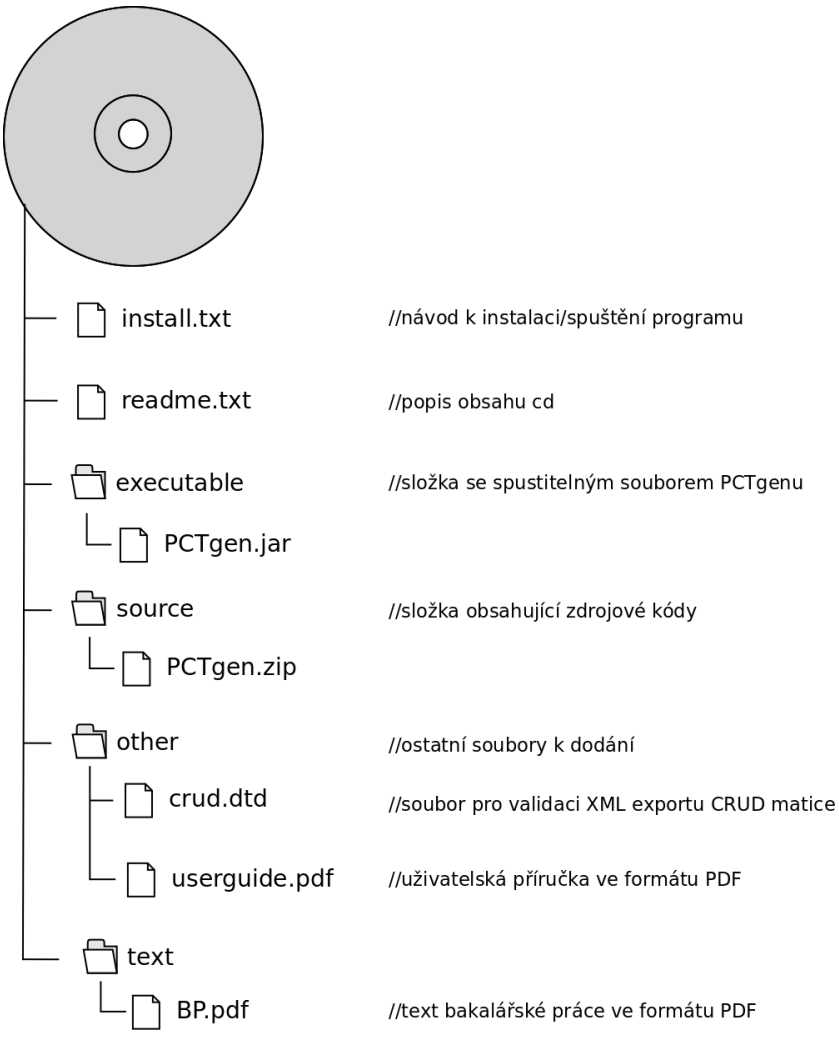
NRI All reads after all changes, I operations reflected

UAT Uživatelské akceptační testování

XML Extensible Markup Language

Příloha B

Obsah přiloženého CD



Obrázek B.1: Seznam přiloženého CD