Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Vizualizace rozsáhlých diagramů

Plzeň, 2013 Daniel Bureš

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 20. 6. 2013 Daniel Bureš

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Lukáši Holému za odborné vedení a konzultace během mé práce.

Abstract

The aim of this master thesis was studying issues with visualization large component diagrams and find out options of work with shown diagrams. Application which was modified, is web based application with supporting HTML 5 technology. Into current application for visualization of large component diagrams was designed and implemented extension, which provides grouping of shown components in graph. Next extension which was designed and implemented was adding support for user accounts. Users can share their uploaded component diagrams or save diagrams only for their use. Application after modified was tested on large component diagram *Nuxeo.*

Obsah

Seznam obrázků

1 Úvod

Software se stává stále složitější a rozsáhlejší a proto je těžší porozumět i jeho struktuře. To platí hlavně v případě, kdy se programátor snaží porozumět cizímu kódu. Jedním z možných zjednodušení této složitosti je použití přístupu komponent v tvorbě softwaru. Každá komponenta typicky obsahuje množství tříd a tím je systém částečně dekomponován. I přesto se dnes systémy mohou skládat ze stovek komponent. Díky této komplexnosti je složité porozumět i diagramům, které zachycují propojení komponent v daném systému.

Cílem této diplomové práce bylo nastudovat problematiku vizualizace rozsáhlých diagramů komponent, navrhnout a implementovat rozšíření do existujícího nástroje CoCA-Ex, který umožňuje vizualizovat diagram komponent na základě vstupní zadané aplikace.

Do stávající aplikace pro vizualizaci komponent CoCA-Ex bylo navrženo několik rozšíření. Největší rozšíření aplikace se týkají nových funkcí v aplikaci seskupení komponent a přidání podpory uživatelských účtů. V této práci jsou výše uvedené funkce blíže rozepsány včetně názorných ukázek a demonstrace použití. Uživatelské účty usnadní práci uživatelům, kteří si mohou uložit do svého účtu více diagramů a ke každému diagramu komponent uložit pozici jednotlivých komponent. S uživatelskými účty byla také přidána možnost sdílení diagramů s ostatními uživateli.

2 Vývoj aplikací

Programování aplikací procházelo postupným vývojem. Začínalo se strukturovaným programováním (jazyky Pascal, C) až se vývoj dostal k objektovému programování (jazyky Java, C#, C++). Nadstavbou objektového programování je tzv. Komponentové programování. Komponentové programování je založeno na komponentách, které jsou vzájemně nezávislé.

Využívání již vytvořených komponent při vývoji aplikací umožňuje vývojářům využívat již hotových a otestovaných funkcí aplikace. Využitím již vytvořených komponent se ušetří čas i peníze, které by se museli investovat na vývoj funkcí, které jsou již hotové. Dále se také zjednoduší údržba aplikace. Pokud se bude upravovat zdrojový kód v komponentě, stačí tuto požadovanou úpravu zdrojového kódu provést v jedné komponentě a nemusí se zasahovat do jiných částí kódu, které tuto komponentu využívají.

2.1 Komponenta

Komponenta je obecný pojem označující část programu, který sdružuje funkce (třídy) s podobnou funkcionalitou. Dle programovacího jazyka může být komponenta označena jako package[1], modul nebo namespace[2].

Následující text je převzat z [1]:

Implementace komponenty (používané objekty, funkce, data) by měla zůstat skrytá, viditelný by měl být pouze interface, přes který se bude komponenta používat. Tomuto principu se říká black-boxový model a je v reálném světě téměř nedosažitelný (i ze zkompilovaného kódu se dá vždy něco získat).

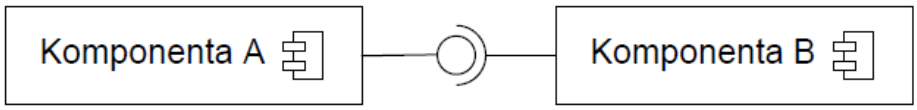
Třetí strana (kdokoliv, kdo komponentu používá) má k dispozici pouze – viditelný interface a specifikaci komponenty.

Konkrétní definice softwarové komponenty se liší v závislosti na modelu, ale obecně pro komponentu platí následující vlastnosti:

- Skrytá/neveřejná implementace funkcionality.
- Je používána třetí stranou (third party).
- Odpovídá komponentovému modelu.

[1] Package – (balík) je používán v jazyku Java pro organizaci tříd do jmenných prostorů
[2] Namespace – Jmenný prostor je používán v jazyce C#, obdoba javovských balíčků

Komponenty mohou být navzájem propojeny pomocí rozhraní. Komponenta A poskytuje vlastnosti komponentě B, analogicky (Komponenta B využívá vlastností komponenty A). Komponenta je v každém programovacím jazyku jiná část kódu.



Obr. 2.1 Spojení komponent [2]

2.2 Komponentové rozhraní

Implementace jednotlivých komponent je neveřejná a je skryta za rozhraním komponent. K funkcím komponenty lze přistupovat pouze přes definované rozhraní.

Následující text je převzat z [1]:

Rozhraní je část komponenty, která je viditelná ostatním. Popisuje, co komponenta umí a co pro to potřebuje. Rozhraní může být popsáno buď programovacím jazykem, nebo pomocí IDL – *Interface Definition Language*. Díky rozhraní může být mezi dvěma komponentami navázána komunikace (contract – pokud jedna komponenta používá druhou pomocí rozhraní). Verze rozhraní je třeba rozlišovat a kontrolovat, jelikož každá verze komponenty může mít jiné rozhraní.

Při používání komponentových rozhraní jsou používány pojmy *Dependency Injection* a *Inversion of Control*, tyto pojmy jsou vysvětleny níže.

Dependency injection je proces, kdy si objekty definují své závislosti (*dependency* – odkaz na jiný objekt, s kterým pracují) pouze pomocí:

- argumentů, se kterými se volá kontruktor,
- argumentů, které se předají nějaké statické tovární metodě (*factory* – vytváří různé objekty s různou konfigurací)
- vlastnosti *(properties)*, které jsou nastaveny objektu potom, co je vytvořený

Dependency objekt, který je vytvořen frameworkem, je po vytvoření přímo předán (injected) objektu, který ho použije. Ve skutečnosti je tedy proces definování závislosti v základě opačný běžnému postupu, protože běžně si objekt/komponenta hledá sama objekt, na kterém závisí. Odtud vznikl název Inversion of Control (IoC).

2.3 Komponentové modely

Níže uvedený text je přeložen ze zdroje [3] :

Doposud neexistuje žádná dohoda o tom, co přesně by mělo být obsaženo v komponentovém modelu. Komponentové modely by měly dodržovat následující vlastnosti:

- Jednotná struktura komponent
- Kvalita poskytovaných služeb komponentou
- Vývoj aplikace a komponent

Jednotná struktura komponent

Dvě komponenty mohou komunikovat pouze v případě, že sdílejí konzistentní předpoklady o tom, co jedna komponenta poskytuje a druhá vyžaduje.

Kvalita poskytovaných služeb komponentou

Komponenty by měly být správně navrženy, aby jejich spolupráce byla bezproblémová a rychlá. Typy komponent v daném systému by měly být standardizovány včetně jejich vzorů pro komunikaci.

Vývoj aplikace a komponent

Úspěšnost komponentově založeného softwarového inženýrství závisí na využívání komponent třetích stran. Předpokladem pro komponentový vývoj aplikace je, že komponenty mohou být nasazeny z prostředí vývojáře do sdíleného pracovního prostoru. Poté může aplikace využívat komponent, které jsou obsažené v tomto sdíleném pracovním prostoru. Z tohoto sdíleného prostoru mohou být komponenty nahrány do uživatelského prostředí.

2.4 Komponentové frameworky

Komponentový framework je implementací komponentového modelu. Jeden framework může implementovat více modelů, než jen jeden. Komponentový framework poskytuje množství služeb *(runtime services)*, které obvykle zařizují běh a komunikaci komponent, případně další podpůrné služby. V mnoha ohledech jsou komponentové frameworky podobné operačním systémům, ačkoliv pracují na mnohem vyšší úrovni abstrakce. [1]

V aplikaci je možné vizualizovat diagramy aplikací, které jsou napsány pomocí frameworku OSGi. Jádro aplikace (nástroj COMAV, který je používán v aplikaci CoCa-Ex) umí také zpracovat frameworky EJB3 a SOFA2.

2.4.1 OSGi

Open Services Gateway initiative je specifikace dynamického modulárního systému pro jazyk Java. Framework OSGi umožňuje instalace a odebírání modulů za běhu a je založen na architektuře modulů, které poskytují služby. Zdroj [4]

Nejznámějším použitím frameworku OSGi je implementace vývojového prostředí Eclipse - Equinox

2.4.2 OSGi komponenty

Komponenty jsou v OSGi uloženy v tzv. bundle. Jedná se o běžný JAR archiv s upravenou hlavičkou v souboru *manifest.mf*. V každém manifest souboru existují parametry, některé z nich jsou uvedené v Tab. 2.1.

Tab. 2.1 Přehled parametrů OSGi v souboru manifest

Bundle-Name	Krátký popis komponenty
Bundle-SymbolicName	Unikátní identifikátor komponenty
Bundle-Version	Verze komponenty
Bundle-Activator	Aktivátor třídy bude spuštěn, pokud je komponenta spuštěna nebo zastavena.
Bundle-RequiredExecutionEnvironment	Specifikace verze Javy
Bundle-ActivationPolicy	Nastavení použití komponenty. Pokud není uvedeno, je spuštěna při nahrání.
Import-Package	Vyžadované komponenty
Export-Package	Poskytované komponenty

2.4.3 EJB

Enterprise Java Beans je standardní komponentní architektura, sloužící pro realizaci aplikační vrstvy informačního systému. EJB komponenty jsou objekty implementované vývojářem, které zajišťují vlastní aplikační logiku systému. Komponenty EJB mají své uplatnění zejména ve 3 a více vrstvách distribuovaných aplikacích, jedná se o součást platformy Java EE. Aktuální verze je EJB 3.1. Specifikace EJB3 je součástí API definující Java Enterprise Edition. Zdroj [5]

Cílem EJB je:

- Poskytnout robustní infrastrukturu pro vývoj rozsáhlých informačních systémů.
- Umožnit vývojáři se soustředit výhradně na problémovou doménu.
- Umožnit tvorbu znovupoužitelných komponent s využitím různých nástrojů od různých dodavatelů a budování aplikací kombinováním takto vytvořených komponent.
- Umožnit distribuci a nasazování aplikací.

EJB kontejner

Komponenty EJB ke své činnosti vyžadují kontejner, který má následující funkce:

- Řídí životní cyklus komponent
- Zajišťuje autentizaci a autorizaci (JAAS[3])
- Umožňuje distribuci komponent (RMI-IIOP[4])
- Řídí transakce
- Může zajišťovat perzistenci
- Poskytuje přístup ke zdrojům (JDBC[5])
- Poskytuje další služby (JavaMail)

Mezi nejrozšířenější volně dostupné aplikační servery s EJB kontejnerem patří JBoss a dále např. Sun Java System Application Server.

EJB komponenty

EJB komponenty jsou distribuovány v archivu JAR. Archiv JAR dané komponenty obsahuje navíc soubor *ejb-jar.xml.* Tento soubor se nazývá deployment deskriptor a obsahuje informace o komponentě. V *ejb-jar.xml* je možné upravovat některé vlastnosti komponenty, jako jsou například práva pro přístup ke komponentě, jejím metodám nebo částečně měnit její funkčnost. Pro spuštění komponent je nutné je

[3] JAAS – Java Authentication and Authorization Service
[4] RMI-IIOP – RMI over IIOP – Java Remote Method Invocation over Internet Inter-Orb Protocol
[5] JDBC – Java Database Connectivity

umístit to EJB kontejneru, kde jsou vzájemně spárovány a řízeny. Zdroj textu o komponentách v EJB: [2]

Enterprise Java Bean architektura dělí komponenty na 3 druhy

- Entity Beans – objektový pohled na data, propojení s databází
- Message-Driven Beans – reagují na události
- Session Beans – obsahují logiku aplikace, dělí se na
    - Stateless Session Beans
    - Statefull Session Beans

2.4.4 SOFA 2

Sofa 2 je komponentový systém využívající hierarchicky složené komponenty. Komponenta je v Sofa 2 zapouzdřený objekt, který s ostatními komponentami komunikuje pouze prostřednictvím poskytovaných a vyžadovaných rozhraní. Od svého předchůdce (SOFA) převzal jádro komponentového modelu, které je vylepšené a rozšířené o následující vlastnosti: Zdroj [2]

- Popis komponentového modelu pomocí meta-modelu.
- Dynamické změny konfigurace architektury komponentové aplikace.
- Skládání komponent a jejich ověřování.
- Podporu vícenásobné komunikace.
- Zavádění aspektů do komponent.
- Oddělení business logiky komponenty od její řídící části a poskytnutí možnosti jednoduché rozšiřitelnosti pomocí aspektů.
- Podpora vývoje komponenty a jejího verzování.

2.5 Formy kompozice

Následující text je přeložen z [3].

Formu kompozice můžeme rozdělit na dva druhy. Na komponenty a na frameworky. Tímto vzniknou 3 hlavní třídy interakce, které se mohou objevit v komponentově založené aplikaci:

- **Komponenta – Komponenta** - Kompozice, která dovoluje interakci mezi komponentami. Tyto interakce poskytují funkčnost aplikace, a tak vazby, které určují tyto interakce, mohou být klasifikovány na úrovni aplikačních vazeb.

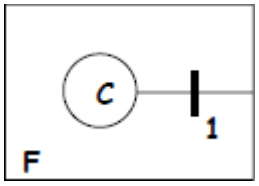
- **Framework – Komponenta** – Kompozice, která dovoluje interakci mezi komponentovým frameworkem a jeho komponenty. Tato interakce poskytuje frameworkům ovládat komponentové zdroje, a tak vazby, které specifikují tyto interakce, mohou být klasifikovány na úrovni systémových vazeb.

- **Framework – Framework** – Kompozice, která dovoluje interakci mezi frameworky. Tyto interakce poskytují složení komponent, které jsou vyvíjeny v různých frameworcích, tím tyto vazby mohou být klasifikovány jako mezioperační vazby.

V následujících podkapitolách jsou zobrazeny a popsány možné vazby mezi komponenty a frameworky. Zdroj [3]

2.5.1 Nasazení komponenty

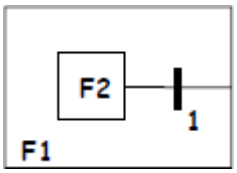
Komponenty musí být nasazeny do frameworku předtím než mohou být složeny nebo spuštěny. Vazby při nasazení (Symbol 1 v Obr. 2.2) popisují rozhraní, které komponenty musí implementovat, poté framework může využívat implementace těchto komponent.



Obr. 2.2 Nasazení komponenty do frameworku

2.5.2 Nasazení frameworku

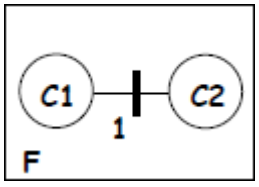
Frameworky mohou být nasazeny do jiných frameworků. Specifikace EJB částečně představuje tento vzor s EJB kontejnery, které se nasazují do EJB serverů. Vazba (Symbol 1 v Obr. 2.3) je odpovídající svými vlastnostmi vazbám, které se použijí při vazbách mezi komponentami.



Obr. 2.3 Nasazení frameworku

2.5.3 Jednoduchá kompozice

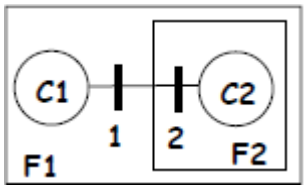
Komponenty nasazené ve stejném frameworku mohou být složené. Vazba kompozice (Symbol 1 v Obr. 2.4) vyjadřuje specifickou funkčnost aplikace a komponenty. Interakční mechanismus podporující tuto vazbu je poskytován frameworkem.



Obr. 2.4 Jednoduchá kompozice

2.5.4 Kompozice různých frameworků

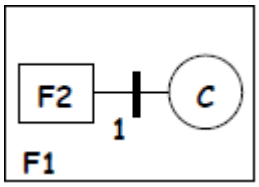
Podpora pro vnořené frameworky vyplývá ze složení komponent napříč frameworky. Frameworky mohou být vnořené nebo na stejné úrovni zanoření. V případě využití složených vazeb (Symbol 1 a 2 na Obr. 2.5) je potřebné, aby interakce dodržovala pravidla pro obecné modely komponent.



Obr. 2.5 Spojení komponent mezi různými frameworky

2.5.5 Rozšíření vnořeného frameworku komponentou

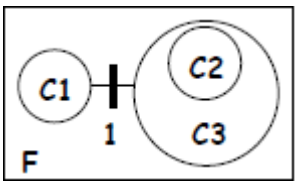
Frameworky mohou být znázorněny jako komponenty a mohou být složeny s jinými komponentami. Tato forma složení nejčastěji dovoluje parametrizaci frameworků chováním jako plugi-iny (rozšíření). Standardní vazby rozšíření pro poskytovatele služeb jsou rostoucí v komerčních frameworcích.



Obr. 2.6 Rozšíření vnořeného frameworku komponentou

2.5.6 Zanoření komponent

Komponentně založený systém je složen z komponent. Schopnost předvídat vlastnosti vztahů mezi komponentami naznačuje podobnou schopnost pro zanořené komponenty. Vazba (Symbol 1 Obr. 2.7) je používána ke složení komponenty C1 a subkomponenty C3, která obsahuje jednu nebo více komponent. Otázka, která se nyní objevuje je následující je komponenta C2 viditelná mimo komponentu C3 a pokud ano je odděleně nasazena.



Obr. 2.7 Zanoření komponent

2.5.7 Kompozice a vztahy v aplikaci CoCA-Ex

Aplikace CoCA-Ex je zaměřena na vizualizaci softwarových komponent obsahující jednoduchou formu kompozice (kap. 2.5.3), která modeluje vztahy komponenta – komponenta (tento vztah je popsán v kap. 2.5).

3 Aplikace CoCA-Ex

CoCA-Ex je aplikace pro vizualizaci softwarových komponent. Tato aplikace je vyvíjena na Katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni. Do aplikace byly navrhnuta a implementována rozšíření, která jsou popsána v kapitole 4 této diplomové práce. V této kapitole je popsána stávající aplikace pro vizualizaci komponent, technologie a jazyky použité v aplikaci *Java Servlet Pages, HTML, CSS, SVG, JavaScript a jQuery*.

3.1 Použité technologie, jazyky, knihovny

V této podkapitole jsou popsány jazyky, které jsou použité v aplikaci - JSP, HTML 5, SVG, CSS, JavaScript a javascriptové knihovny jQuery a MooTools.

3.1.1 Java Servlet Pages - JSP

JSP je technologie, která umožňuje tvořit dynamické webové stránky. Výhoda použití této technologie oproti desktopové aplikaci je snadná dostupnost uživatelům. Uživatelům stačí webový prohlížeč pro práci s aplikací. Výhodou technologie JSP oproti jiným serverovým technologiím např. PHP je využití kompilovaného jazyka Java. PHP je jazykem interpretovaným, proto se předpokládá větší rychlost u JSP při obsluze jednotlivých požadavků serverem.

3.1.2 Jazyky HTML 5, SVG

HTML je zkratka pro HyperText Markup Language, tento jazyk se používá ve webových aplikacích k zobrazení informací na webových stránkách. Jazyk HTML má za sebou dlouhý vývoj a jeho aktuální verze je HTML 5. Existuje také jazyk XHTML aktuálně verze 1.1, který je kombinací HTML a XML. V XHTML je oproti HTML nutné uzavírat html elementy.

Standard HTML 5 přináší nové možnosti využití webových stránek. Do HTML 5 byla přidána podpora vektorové grafiky, podpora HTML elementů audio, video, podpora dalších typů prvků ve formulářích. [1]

SVG je zkratka pro Scalable Vector Graphics. Tato technologie umožňuje kreslit vektorovou grafiku pomocí XML. V aplikaci je technologie SVG použita pro vykreslení diagramu komponent. Při použití kombinace HTML 5, SVG a JavaScript je možné v aplikaci přesouvat jednotlivé komponenty v diagramu, přidávat je do skupin a provádět s nimi další operace.

3.1.3 CSS

CSS je zkratka pro Cascading Style Sheets. CSS je jazyk, ve kterém je definováno jak má webový prohlížeč zobrazit prvky html. Výhodou použití kombinace HTML a CSS je oddělení textových informací a stylu prvků na webové stránce. [2]

3.1.4 JavaScript

JavaScript je skriptovací jazyk určený pro běh ve webové aplikaci na straně klienta (tj. přímo v prohlížeči uživatele). Pomocí JavaScriptu mohou být na webových stránkách naprogramovány dynamické funkce, které se vykonají po určité akci uživatele např. po kliku na určité tlačítko, přejetí myši nad obrázkem, vyjetí menu.

3.1.5 jQuery

JQuery je javascriptový framework určený pro rychlý vývoj aplikací v jazyku JavaScript. Pomocí tohoto frameworku lze velmi rychle přistupovat k prvkům na webové stránce a nastavovat jim potřebné vlastnosti. Mezi další velkou výhodu jQuery patří kompatibilnost se všemi prohlížeči a nezávislost na verzi prohlížeče.

Aktuálně je vydán framework jQuery ve verzi 2.0, které ovšem nepodporuje verze Internet Explorer 6, 7, 8, ale stále je vývojáři udržovaná verze 1.9.1, která podporuje starší verze prohlížeče Internet Explorer. [3]

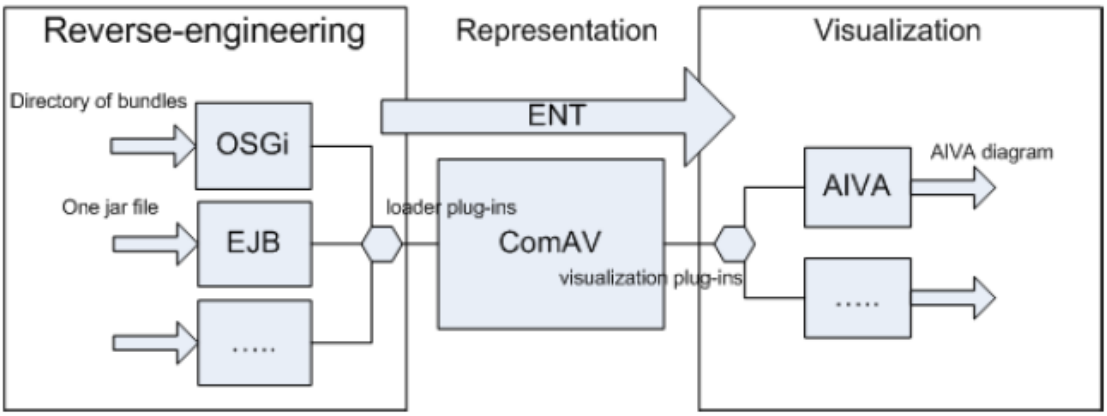
3.1.6 MooTools

Mootools je JavaScriptový framework určený pro podporu objektového programování v prototypovém jazyku JavaScript. Framework Mootools plně podporuje nejrozšířenější verze internetových prohlížečů Internet Explorer 6 a novější, Safari, Firefox, Opera a Google Chrome.

3.2 ComAV

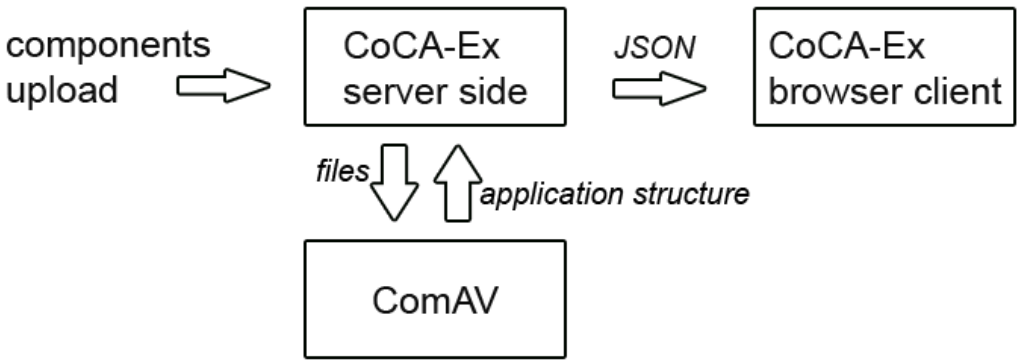
ComAV je platforma pro vizualizaci softwarových komponent vyvíjená na Katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni. ComAv umí vytvořit z archivů JAR závislosti mezi komponentami. Tato funkčnost nástroje ComAV je využita v aplikaci CoCA-Ex.

ComAV umožňuje vytvořit diagramy z následujících komponentových frameworků OSGi, EJB a SOFA 2. Architektura nástroje ComAV je zobrazena na Obr. 3.1. O načtení potřebných dat k vytvoření diagramu je v programu ComAV využíván tzv. *loader*. K vytvoření diagramu se poté využije tzv. *visualization plugin*.



Obr. 3.1 Architektura ComAV Zdroj: [9]

Schéma využití nástroje ComAV v aplikaci CoCA-Ex je zobrazeno na Obr. 3.2. Uživatel provede nahrání komponent do aplikace CoCA-Ex. CoCA-Ex tyto komponenty předá na zpracování do ComAV. ComAV vrátí do aplikace strukturu zadaných komponent a vztahy mezi komponentami. Jakmile CoCA-Ex obdrží strukturu aplikace, předá ji ve zprávě typu JSON do prohlížeče uživatele.



Obr. 3.2 Schéma využití aplikace ComAV v CoCA-Ex

3.3 Popis aplikace CoCA-Ex

Aplikace CoCA-Ex je webová aplikace složená ze dvou webových stránek, na jedné stránce má uživatel možnost nahrát komponenty (*archivy JAR*). Po výběru požadovaných komponent a odeslání formuláře je uživateli zobrazena webová stránka s diagramem komponent (Snímek obrazovky je zobrazen v příloze B a C).

3.4 Techniky pro vizualizaci diagramů

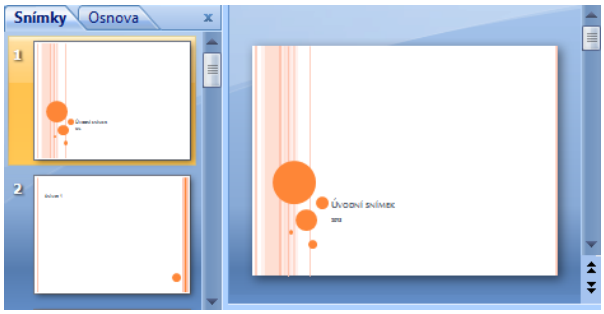
V této podkapitole jsou rozepsány techniky pro vizualizaci rozsáhlých diagramů *Overview+Detail, Zoom, Focus+Context*. Tyto techniky jsou sepsány ze zdroje [10].

3.4.1 Overview + Detail

Technika vizualizace *Overview+Detail* lze přeložit jako náhled a detail. Použití této techniky umožňuje souběžné zobrazení náhledu a detailu v jednom okně. Náhled i detail jsou od sebe vizuálně oddělené např. ohraničením, které je vidět na Obr. 3.3. Pokud uživatel provádí akci např. posun mapy v detailu nebo v náhledu, okamžitě je tento posun proveden i v druhé části okna.

Obr. 3.3 Použití techniky Overview + Detail v aplikaci Google Maps, zdroj [10]

Do techniky *Overview+Detail* můžeme také zařadit posuvníky (*ScrollBars*), které umožňují posun v jednom směru. Techniku s posuvníky můžeme vidět např. v aplikaci Microsoft PowerPoint, která obsahuje v levé části okna jednotlivé snímky prezentace *thumbnail* – náhledy a v pravé části okna je zobrazen *detail* - aktuální snímek Obr. 3.4.



Obr. 3.4 Ukázka Overview+Detail, použití posuvníků v MS PowerPoint

3.4.2 Zoom

Pomocí techniky *zoom* neboli přibližování či oddalování je uživateli poskytnuta možnost rychle přiblížit požadovanou oblast nejčastěji pomocí kombinace stisku kláves *Ctrl + + (plus)* či *Ctrl + – (mínus).* Technologie je založena na postupném přibližování, v aplikaci je definováno několik úrovní přiblížení a při každém posunu se aplikace přiblíží o jednu úroveň. To umožní uživateli plynulé zvětšení nebo zmenšení požadované oblasti. Úroveň přiblížení je také zobrazena v okně jako posuvník. Nejvíce rozšířeným použitím této technologie jsou např. mapy na internetu. Na Obr. 3.5 je vidět ukázka techniky zoom s vyznačeným posuvníkem zobrazujícím úroveň přiblížení mapy.

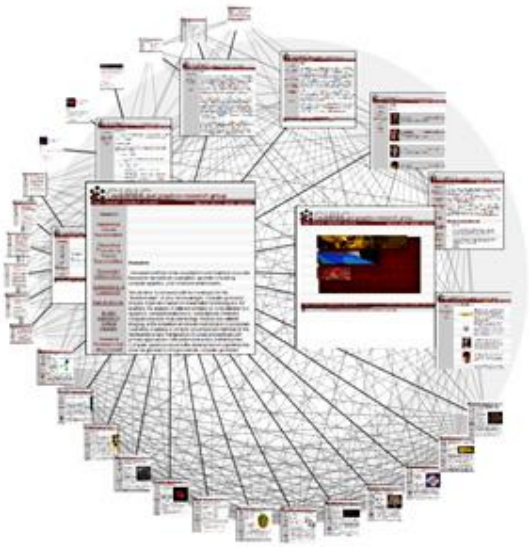
Technika *zoom* je často kombinována s technikou *overview+detail.* Je tím dosaženo vyššího uživatelského komfortu při obsluze aplikace.

Obr. 3.5 Ukázka techniky zoom s vyznačeným posuvníkem zobrazujícím úroveň přiblížení (Zdroj: Google Maps)

3.4.3 Focus + Context

Technika *overview + detail* rozdělovala okno do dvou částí. Technika *Focus + Context* zobrazuje náhled *(Focus)* přímo ve svém okolí *(Context)* v jednom okně. Při použití této techniky dochází ke zvětšení náhledu a nedochází zde k vizuálnímu oddělené přiblížené oblasti a okolí této oblasti.

V této technice může být nastaven různý počet úrovní přiblížení při přechodu z oblasti náhledu do oblasti okolí. Pro přiblížení dané oblasti mohou být použité různé algoritmy přiblížení. Ukázka použití techniky *Focus + Context* je zobrazen na Obr. 3.6.



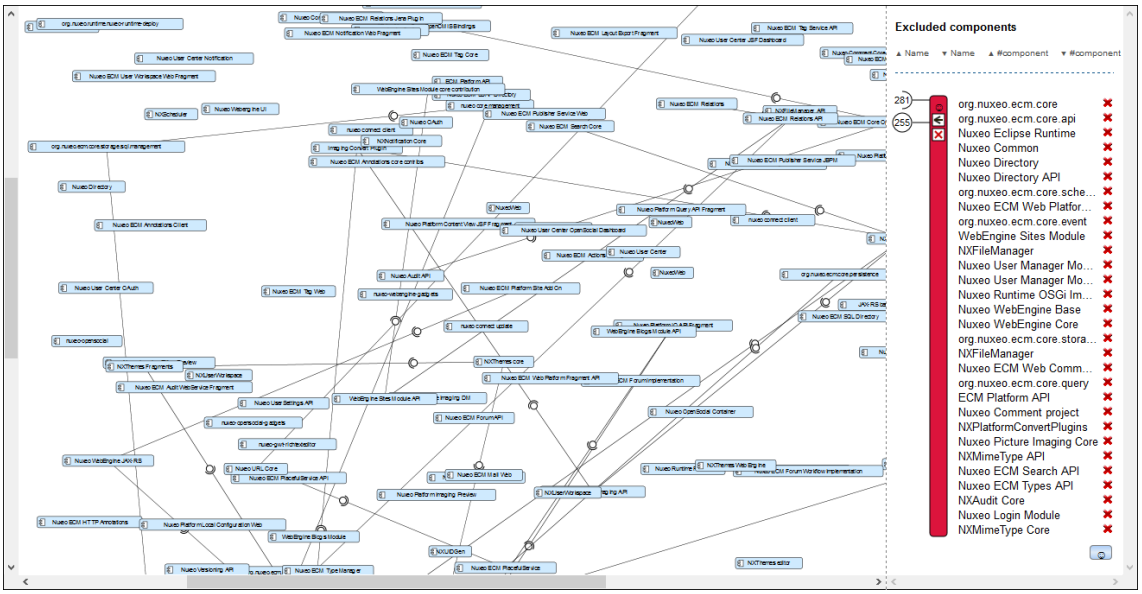
Obr. 3.6 Použití techniky Focus + Context, zdroj [11]

3.5 Technika pro vizualizaci CoCA-Ex

Vizualizační část aplikace CoCA-Ex je rozdělena na dvě oblasti viz (Obr. 3.7):

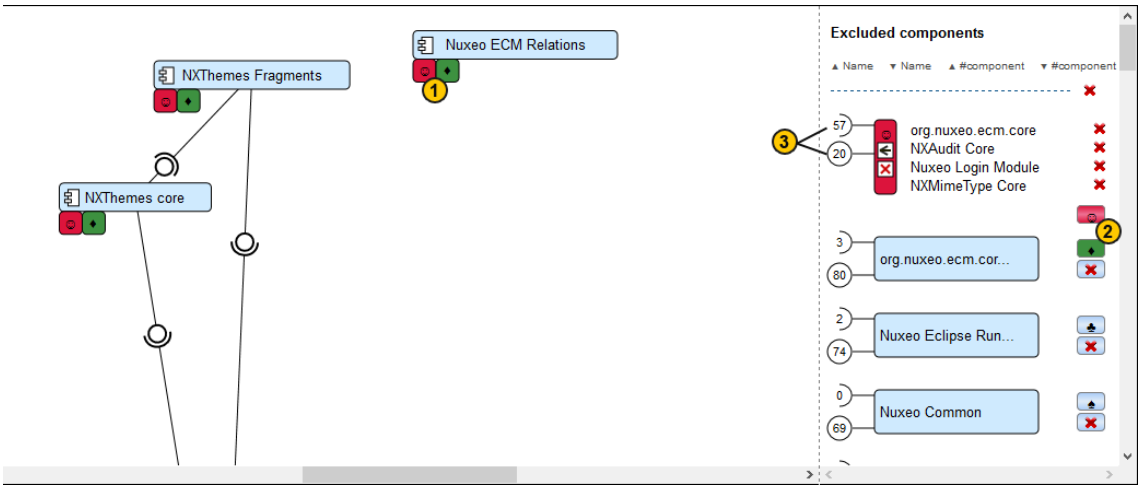
- oblast diagramu
- pravý panel

Oblast diagramu obsahuje komponenty a vazby mezi nimi (levá část Obr. 3.7). Pravý panel tzv. SeCo (Separated Components) je určen k zobrazení vyjmutých komponent z diagramu, a dále také k zobrazení rozhraní, která jsou spojena s diagramem komponent (pravá část Obr. 3.7).



Obr. 3.7 Zobrazení diagramů komponent v aplikaci CoCA-Ex

SeCo může obsahovat vyjmuté komponenty z grafu nebo vyjmuté skupiny komponent z grafu. Každá položka v pravém panelu (komponenta či skupina komponent) obsahuje atribut s počtem poskytovaných a vyžadovaných komponent a zástupný symbol, který je jednoznačný pro každou položku v SeCo. Pokud uživatel klikne na symbol v SeCo, jsou v oblasti diagramu zobrazeni tzv. Delegáti u těch komponent, které jsou spojeny s odpovídající položkou v SeCo panelu. Delegát je graficky reprezentován stejnou ikonou jako symbol.



Obr. 3.8 CoCA-Ex Zobrazení delegátů

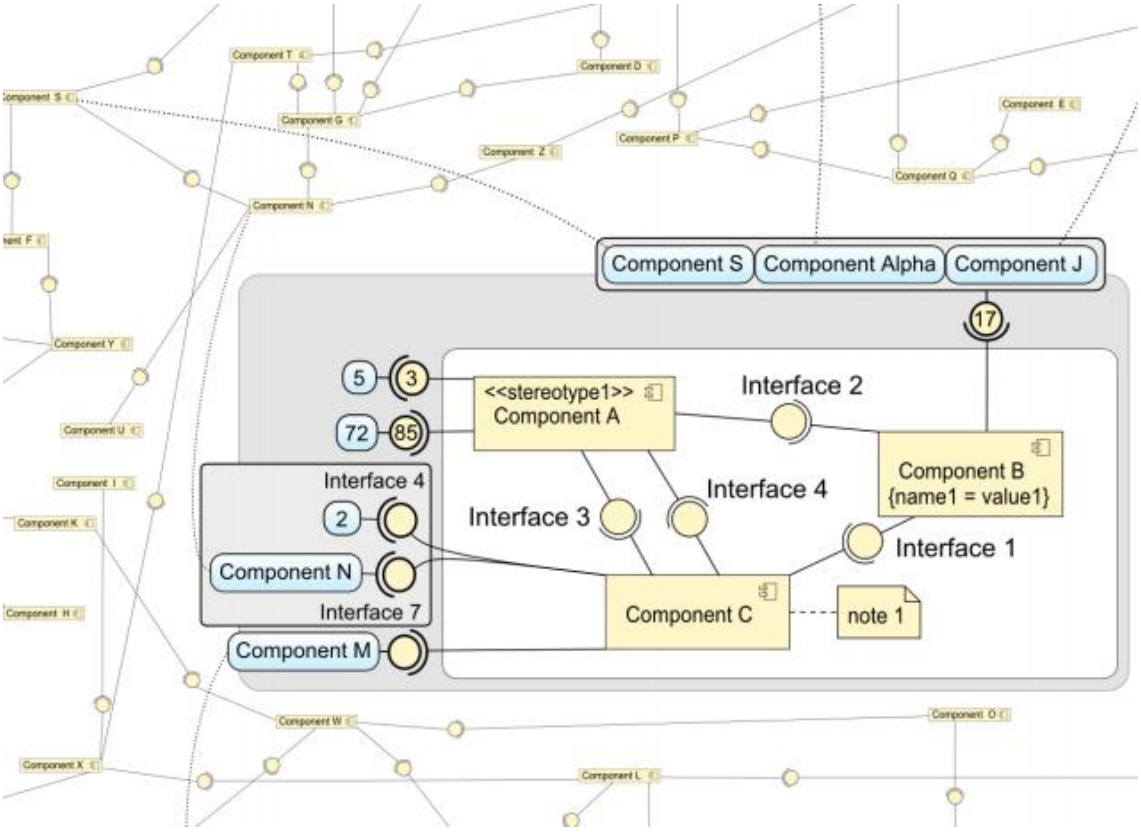
Na Obr. 3.8 jsou znázorněny delegáti (značka 1). Tyto delegáti se zobrazí po kliknutí na symbol u sousedících komponent, které jsou vyjmuté v pravém panelu (značka 2). Jedná se o zvýraznění sousedících komponent u vyjmuté položky.

Dále je na Obr. 3.8 značkou 3 vyznačen počet komponent, které sousedí s danou položkou. V tomto případě skupina komponent vyžaduje k běhu padesát sedm komponent a poskytuje rozhraní dvaceti komponentám.

Z této skupiny komponent má uživatel možnost odstranit postupně jednotlivé komponenty, které se poté zobrazí v diagramu komponent, včetně zobrazení a zachování vazeb s ostatními komponentami. Uživatel má také možnost přesunout celou skupinu do diagramu komponent a to kliknutím na šipku ukazující doleva u skupiny komponent.

3.6 Viewport pro komponentové diagramy

Následující text je přeložen ze zdroje [12]. Při zobrazení rozsáhlých diagramů komponent, které obsahují stovky až tisíce komponent, se stalo zobrazení diagramů velmi nepřehledné. Tento problém řeší metoda zobrazení viewportu přímo v diagramu mezi ostatními komponentami. Tím dochází k zachování původního diagramu, ale zároveň je umožněno uživateli zobrazit požadované komponenty v přehledném okně se zachováním vazeb na okolní prostředí (viz Obr. 3.9). Komponenty mimo oblast viewportu jsou zmenšené, ale vedou k nim vazby od komponent, které jsou zobrazené v části viewport.



Obr. 3.9 Viewport pro komponentové diagramy, zdroj [12]

Vlastnosti viewportu:

- *Clustering* (seskupení) – Vazby každé komponenty ve viewportu jsou seskupeny do dvou množin – požadovaná a poskytovaná rozhraní.
- *Proxy components* (zástupné komponenty) – Zastupují všechny komponenty, které jsou požadovány nebo poskytovány komponentou ve viewportu. V těchto zástupných komponentách je vždy zobrazen počet požadovaných či poskytovaných komponent.
- *Interactivity of the border area* – Vlastnost zahrnuje možnost uživatele manipulovat se seskupenými rozhraními či komponentami, úpravu rozvržení komponent nebo vybírání komponent zobrazených ve viewportu.

3.7 Možná budoucí rozšíření

Z výše uvedených technik pro vizualizaci diagramů komponent popsané v kap. 3.3 - 3.6 by mohla být aplikace do budoucna vylepšena o rozsáhlejší podporu metody zavedení viewportu. V této diplomové práci byla implementována metoda viewportu pomocí funkce seskupení komponent (blíže je tato funkce popsána v kap. 4.2).

Jako další možné rozšíření aplikace, by bylo zavedení prozkoumání rozhraní v SeCo. V současné době se po kliknutí na počet požadovaných a poskytovaných komponent, které jsou zobrazené v SeCo, komponenty pouze zvýrazní v diagramu.

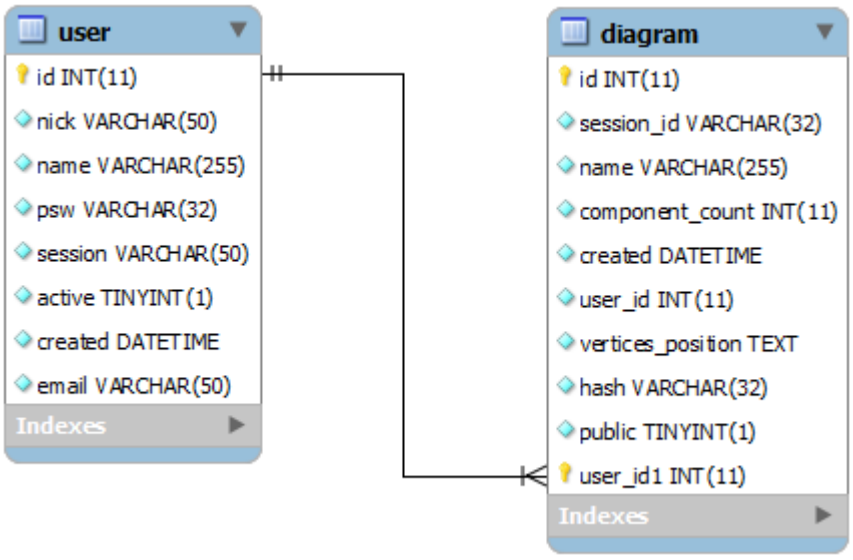
4 Návrh a implementace funkcí

V této části práce jsou rozepsány implementované techniky a metody pro lepší práci a manipulaci s rozsáhlými diagramy komponent.

4.1 Podpora databáze

Doposud nebylo v aplikaci možné přihlašování uživatelů, ukládání diagramů pro pozdější zobrazení, sdílení nahraných diagramů nebo zobrazení diagramů jinými uživateli. Pro podporu výše uvedených vlastností aplikace bylo nutné napojit stávající aplikaci na databázi.

Byla zvolena databáze MySQL, z důvodu její rozšířenosti, snadné použitelnosti, rychlé instalaci a licenci zdarma. Pro použití aplikace s databází byl navržen ERA model (Obr. 4.1).



Obr. 4.1 ERA model databáze

4.1.1 Popis databázových tabulek

Popis databázových tabulek je uveden v Tab. 4.1 a v Tab. 4.2.

Tab. 4.1 Popis databázové tabulky: user

Tabulka: User		
Název	**Datový typ**	**Popis**
Id	Integer	Id uživatele
Nick	Varchar(50)	Přihlašovací jméno uživatele
Name	Varchar(255)	Jméno uživatele
Psw	Varchar(32)	Heslo zakódované pomocí funkce MD5
Session	Varchar(50)	Session id
Active	TinyInt(1)	Aktivní uživatel (1 = aktivní, lze se přihlásit, 0 = nelze se přihlásit)
Created	Datetime	Datum registrace uživatele
Email	Varchar(50)	E-mail uživatele

Tab. 4.2 Popis databázové tabulky: Diagram

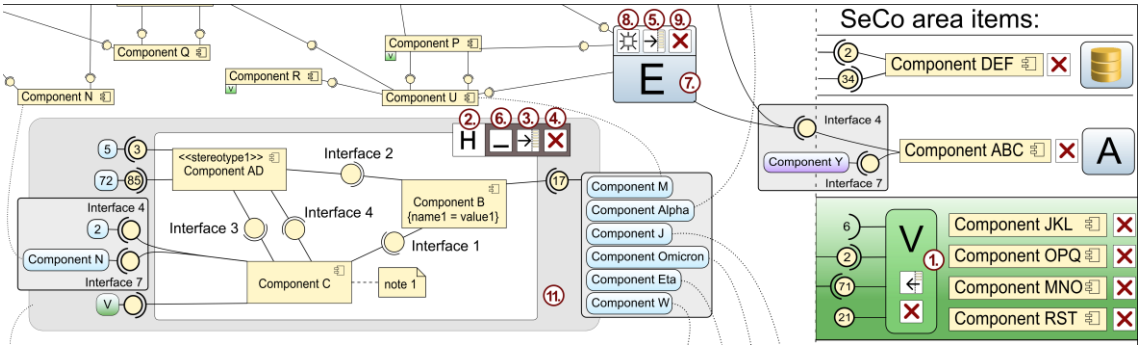
Tabulka: Diagram		
Název	**Datový typ**	**Popis**
Id	Integer	Id diagramu
Session_id	Varchar(32)	Session id
Name	Varchar(255)	Název diagramu
Component_count	Integer	Počet komponent v diagramu
Created	Datetime	Datum vytvoření diagramu
User_id	Integer	Id uživatele
Vertices_position	TEXT	Pozice komponent v zobrazeném diagramu (JSON)
Hash	Varchar(32)	Kontrolní kód používaný při načítání komponent
Public	TinyInt(1)	Veřejný diagram: 1 = zveřejněn, 0 = nezveřejněn

4.2 Skupiny komponent

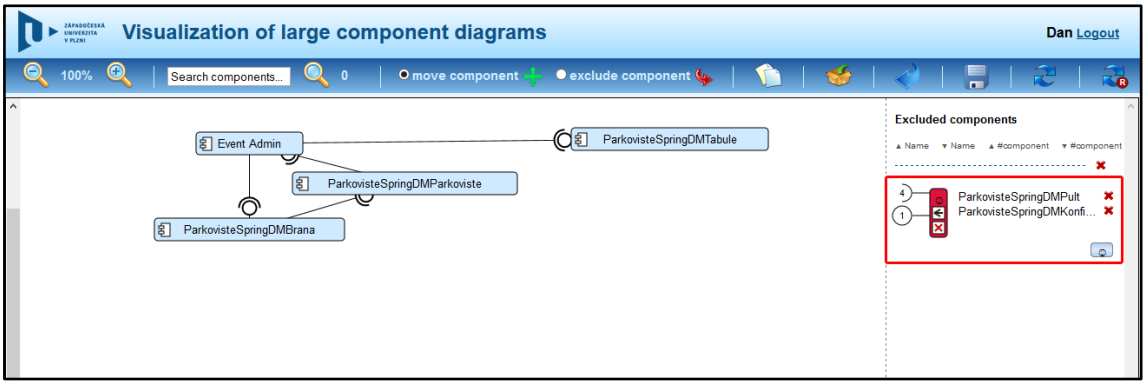
Pro lepší práci se zobrazeným diagramem komponent bylo navrženo rozšířit zobrazení diagramu komponent o funkčnost zobrazit skupiny komponent přímo v diagramu. Implementace této funkčnosti byla navržena dle metody viewportu pro komponentové diagramy (kap. 3.6). Funkčnost zachovává zobrazení hran vedoucích do skupiny komponent i hran vedoucích od skupiny komponent k jiným komponentám. Doposud existovala tato funkčnost skupin komponent, ale zobrazovala se pouze v pravém panelu v tzv. *SeCo (Separated Component area* Obr. 4.3).

Detailní pohled na návrh této funkčnosti lze vidět na Obr. 4.2. Na tomto obrázku jsou označené jednotlivé prvky značkami: značka 1 – skupina komponent v SeCo, značka 11 – rozšířená skupina komponent, značka 7 – minimalizovaná skupina komponent, značka 2 – označení skupiny komponent symbolem, značky 3, 4, 5, 6, 8, 9 slouží pro přepínání mezi stavy skupin komponent. Skupina komponent může nabývat následujících stavů:

- Skupina komponent v Seco (Obr. 4.2 značka 1)
- Rozšířená skupina komponent (Obr. 4.2 značka 11)
- Minimalizovaná skupina komponent (Obr. 4.2 značka 7)

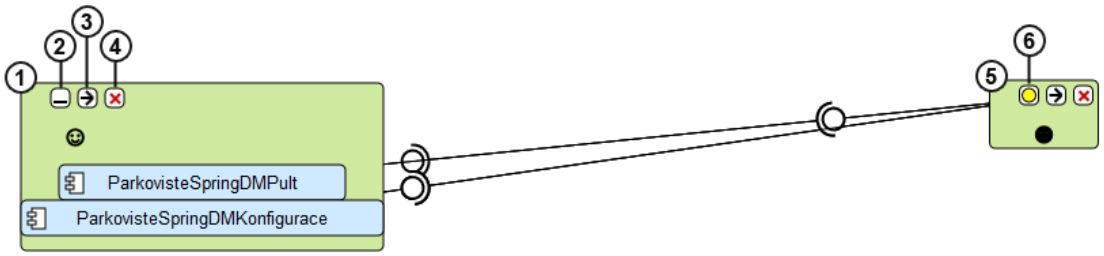


Obr. 4.2 Návrh funkčnosti viewportu



Obr. 4.3 Zvýrazněný pravý panel tzv. SeCo

Seskupení komponent v diagramu je zobrazeno na Obr. 4.4. Dále jsou na tomto obrázku vyznačené důležité prvky a funkce, které jsou popsány v Tab. 4.3. Na obrázku je také vidět důležitá vlastnost seskupených komponent - při seskupení zůstala zachována vazba jednotlivých komponent s ostatními komponentami v diagramu.



Obr. 4.4 Seskupení komponent v diagramu

Tab. 4.3 Popis obrázku seskupených komponent

Číslo	Popis
1	Zobrazené komponenty ve skupině.
2	Tlačítko ke skrytí komponent ve skupině. Po kliku na toto tlačítko nejsou zobrazeny komponenty v dané skupině. Celá skupina je zobrazena v malém boxu (viz č. 5).
3	Tlačítko k převedení skupiny komponent z diagramu do *SeCo*.
4	Tlačítko k odstranění všech komponent ze skupiny. Komponenty zůstanou zobrazeny na stejném místě, akorát bude zrušena jejich vazba na danou skupinu.
5	Zobrazená skupina komponent.
6	Tlačítko k zobrazení komponent ve skupině. Po kliku na toto tlačítko jsou komponenty zobrazeny stejně jako na prvku označeném č. 1.

4.2.1 Implementace skupin komponent

Vytvoření skupiny komponent v diagramu je řešeno pomocí přidání nového elementu *g* v SVG tento element má atribut *class* nastaven na hodnotu *group_vertices*, to je nastaveno z důvodu snadného *JavaScriptového* prohledávání, zda se jedná o komponentu či o skupinu komponent. Níže je uveden výpis skupiny komponent. Element *g* obaluje celou skupinu obsahující zelený obdélník (který představuje skupinu komponent) a všechna viditelná tlačítka uvnitř tohoto obdélníku.

```
<g  id="gv_DC143C☺"
    class="group_vertices"
    transform="translate(1757,1460)">

    <rect rx="4" ry="4" height="50" width="80"
        stroke="black" fill="#CFEA9E"
        style="cursor:pointer">

    <text x="40" y="38" font-size="20" fill="black"
        text-anchor="middle" dominant-baseline="central">☺</text>

    <a xlink:href="#"
        onclick="OffScreenKiv.removeAllComponentFromGroup('DC143C☺');
                 return false;"> …</a>

    <a xlink:href="#"
        onclick="  OffScreenKiv.showComposeComponents('DC143C☺');
                   OffScreenKiv.showGroupInRightPanel('DC143C☺');
                   return false;">…</a>

    <a xlink:href="#"
        onclick="OffScreenKiv.showDecomposeComponents('DC143C☺');
        return false;">…</a>
</g>
```

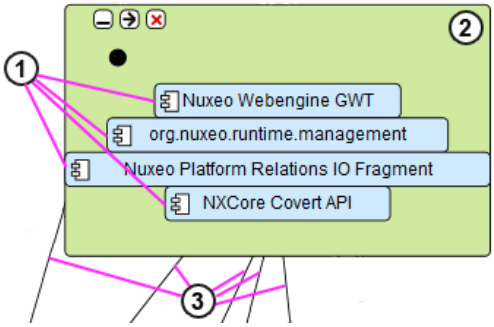
Při implementaci v SVG bylo nutné vyřešit problém s překrýváním komponent. SVG nepodporuje u elementů vlastnost *z-index*[6]. V SVG platí pro zobrazení prvků jednoduché pravidlo: Prvek, který je ve zdrojovém kódu blíže konci souboru, je zobrazen „blíže“ k uživateli. (Překrývá prvky, které mají stejnou pozici *x* a *y.*) Této vlastnosti bylo využito při implementaci funkce seskupení komponent. Pokud je skupina komponent minimalizovaná (viz Obr. 4.4 značka 5) jsou seskupené komponenty zobrazeny ve zdrojovém kódu před skupinou komponent (zelený obdélník). Komponenty jsou skryté pomocí vlastnosti *hidden,* tím je dosaženo zachování viditelnosti vazeb vedoucích k ostatním komponentám a zároveň jsou skryté seskupené komponenty (modré obdélníky).

[6] Z-index – Css vlastnost, která umožní nastavit výšku prvku na ose Z. Hodnota představuje prioritu při překrývání prvků, čím je tato hodnota vyšší, tím je prvek zobrazen blíže uživateli.

Na Obr. 4.5 značka 2 lze vidět rozšířenou skupinu komponent. Implementací tohoto rozšíření byly provedeny následující operace:

- *Nastavení šířky obalujícího elementu skupiny (zelený obdélník)* – šířka je nastavena dle nejširší komponenty ve skupině, výška je nastavena dle počtu elementů
- *Posun komponent na konec SVG elementu* – Tím se komponenty přesunou „blíže“ k uživateli a zůstanou viditelné (viz Obr. 4.5 značka 1). Vazby mezi komponentami mají ponechanou pozici „na pozadí“ umístěné za skupinou komponent (viz Obr. 4.5 značka 3).

S komponentami, které jsou obsažené v rozšířené skupině komponent lze posunovat a přesouvat je na jiné pozice. Ovšem při posunu celé skupiny komponent, je tato skupina minimalizována (viz Obr. 4.4 značka 5). Minimalizace skupiny komponent se provádí kvůli rychlosti při přesunu ve webovém prohlížeči.

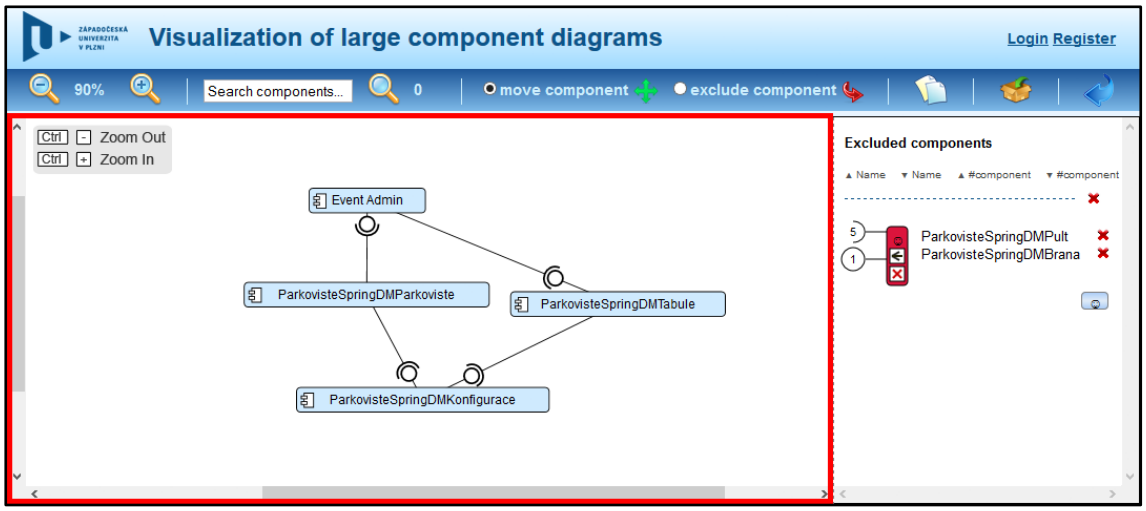


Obr. 4.5 Rozšířená skupina komponent

4.3 Přiblížení a oddálení diagramu

Před úpravou aplikace nebylo možné snadno přibližovat či oddalovat zobrazený diagram komponent. Funkce přiblížení a oddálení byla přístupná přes tlačítka s lupou a symbolem „+“ („-“), ale při každém kliku na tato tlačítka se diagram komponent vycentroval.

Další možností bylo použití klávesových zkratek dostupných v prohlížeči [ctrl] a [+] ([ctrl] a [-]), nebo klávesy [ctrl] s využitím kolečka myši. Nevýhoda tohoto způsobu přiblížení diagramu se objevila ihned – přibližovala se celá stránka včetně horního menu i pravého sloupce, který obsahuje komponenty odebrané z diagramu.



Obr. 4.6 Vyznačená oblast diagramu

Funkce přiblížení a oddálení diagramu byla upravena a při jejím použití mohou nastat dvě možnosti. Pokud je kursor myši v oblasti nad diagramem, poté je přiblížen pouze obsah diagramu (Obr. 4.6). Pokud je kursor myši mimo oblast diagramu, je přiblížena celá webová stránka, stejně jako při použití funkce přiblížení nebo oddálení v menu prohlížeče. Při stisku klávesy *Ctrl* je automaticky zobrazena nápověda na přiblížení nebo oddálení webové stránky (Obr. 4.6).

Při přiblížení nebo oddálení je pomocí *JavaScriptu* zjištěna pozice kurzoru myši nad oblastí diagramu a následně zvětšeno zobrazení s přiblížením na kurzor myši. Po úpravě se neprovede vycentrování obrazovky, které probíhalo před touto úpravou.

4.4 Uživatelé

Aplikace byla původně navržena bez podpory databáze a uživatelských účtů. Pokud uživatelé chtěli použít vizualizační nástroj, museli vždy znovu vybrat všechny komponenty a nahrát je znovu na server, po nahrání komponent na server bylo možné tyto komponenty zobrazit v diagramu.

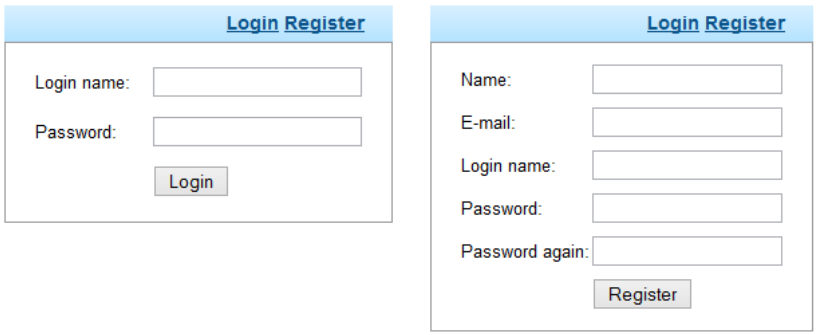
Pro efektivnější využití bylo navrženo propojit aplikaci s databází, ve které budou uložené nahrané diagramy a uživatelské účty. Uživatel má možnost se zaregistrovat a přihlásit. Jakmile je uživatel přihlášen může si vytvořit diagram, nahrát k tomuto diagramu komponenty a uložit diagram pro pozdější zobrazení.

Nahrané diagramy může uživatel kdykoliv znovu zobrazit. Uživatel může také poslat pouze odkaz na stránku se zobrazeným diagramem komponent. Existuje možnost nastavit diagramu atribut veřejný – poté se tento diagram zobrazí všem uživatelům. Uživatelé si tento diagram mohou zkopírovat do svého účtu a dále přidávat a odebírat komponenty z tohoto diagramu.

4.4.1 Implementace uživatelských účtů

Registrace a přihlašování uživatelů je řešena pomocí dvou HTML formulářů (Obr. 4.7), které se při kliku na daný odkaz zobrazí pomocí JavaScriptu. Registrační formulář je ošetřen na unikátnost zadaného e-mailu a přihlašovacího jména.

Po odeslání registračního formuláře je pomocí JavaScriptu zaslán požadavek na server, který vrátí výsledek registrace. Pokud proces registrace vrátí chybové zprávy, jsou poté zobrazeny ve formuláři. Pokud registrace proběhne v pořádku, je následně zobrazen pouze přihlašovací formulář, ve kterém je již předvyplněné přihlašovací jméno.



Obr. 4.7 Přihlašovací formulář a registrační formulář

4.5 Správa diagramů

Pro efektivnější využití aplikace bylo navrženo a implementováno rozšíření stávající aplikace o možnost správy diagramů. Pod pojmem správa diagramů si můžeme představit možnost nastavení názvu diagramu, možnost nastavení veřejného diagramu, možnost uložení a načtení pozic komponent v zobrazeném diagramu.

Správa diagramů je zobrazena pouze přihlášenému uživateli (Obr. 4.8). Přihlášený uživatel má možnost nahrát nový diagram, upravit nahraný diagram nebo zobrazit některý zveřejněný diagram od jiného uživatele. Na stránce se zobrazeným diagramem je možnost vrátit se na stránku s výběrem komponent (Obr. 4.8 – ikonka č. 1).

Obr. 4.8 Menu přihlášeného uživatele

Přihlášený uživatel může ukládat pozice komponent v diagramu (Obr. 4.8 - ikonka č. 2). Na Obr. 4.8 jsou dále vyznačené položky v menu č. 3 a č. 4. Položka v menu označená č. 3 umožňuje načtení uložených pozic komponent v diagramu. Položka č. 4 je určena pro náhodné načtení komponent do diagramu (Pozice komponent, které uživatel dříve uložil, zůstanou zachovány).

4.5.1 Implementace uložení pozic komponent

K implementaci funkce uložení komponent bylo využito skriptovacího jazyka JavaScript, pomocí kterého se získají data ze zobrazené stránky a odešlou se ke zpracování na server. Po kliku na tlačítko č. 2 z Obr. 4.8 je zavolána funkce, která postupně projde veškeré komponenty v diagramu. U každé komponenty zjistí jednoznačný identifikátor tzv. *id* a pozici této komponenty. Pozice komponenty je uložena v svg elementu jako atribut *transform*.

Data jsou odesílána na server pomocí metody post. Data obsahují parametry *id_diagram* a *vertices_position*. Id_diagram je jednoznačný identifikátor diagramu uložený v databázi. Parametr vertices_position obsahuje data ve formátu JSON. Ukázka dat ve formátu JSON

```
{"vertices_position":[
  {"id":"vertex1" , "transform":"translate(527,730)" },
  {"id":"vertex2" , "transform":"translate(724,771)" },
  {"id":"vertex3" , "transform":"translate(882,734)" },
  {"id":"vertex4" , "transform":"translate(483,980)" },
  {"id":"vertex5" , "transform":"translate(905,20)"  },
  {"id":"vertex6" , "transform":"translate(577,26)"  } ]}
```

Požadavek je zpracován servletem *SaveDiagram,* který zároveň provede kontrolu, zda má uživatel právo nastavit pozice u daného diagramu. Po úspěšném uložení pozic je zobrazena hláška informující o uložení pozic komponent.

4.5.2 Implementace načtení pozic komponent

Funkce pro načítání pozic komponent byla implementována pomocí HTML odkazu na stránku. Jako parametry byly zvolené id_diagramu, tj. jednoznačný identifikátor diagramu a hash kód diagramu, tj. kontrolní kód pro ověření, zda má uživatel právo načítat daný diagram.

```
<SERVER_URL>VisualizationTool/ShowGraph?diagram_id=<id_diagramu>
&diagram_hash=<hash_kód>
```

Diagram je nejdříve načten bez pozic jednotlivých komponent. Je to stejný způsob jako při běžném načtení stránky s diagramem. Po načtení diagramu je pomocí JavaScriptu odeslán metodou http GET požadavek na získání pozic jednotlivých komponent.

```
<SERVER_URL>VisualizationTool/LoadDiagram?diagram_id=<id_diagramu>
&diagram_hash=<hash_kód>
```

Server vrátí JSON zprávu, která obsahuje všechny uložené pozice komponent.

```
{"vertices_position":[
  {"id":"vertex1" , "transform":"translate(527,730)" },
  {"id":"vertex2" , "transform":"translate(724,771)" },
  {"id":"vertex3" , "transform":"translate(882,734)" },
  {"id":"vertex4" , "transform":"translate(483,980)" },
  {"id":"vertex5" , "transform":"translate(905,20)"  },
  {"id":"vertex6" , "transform":"translate(577,26)"  } ]}
```

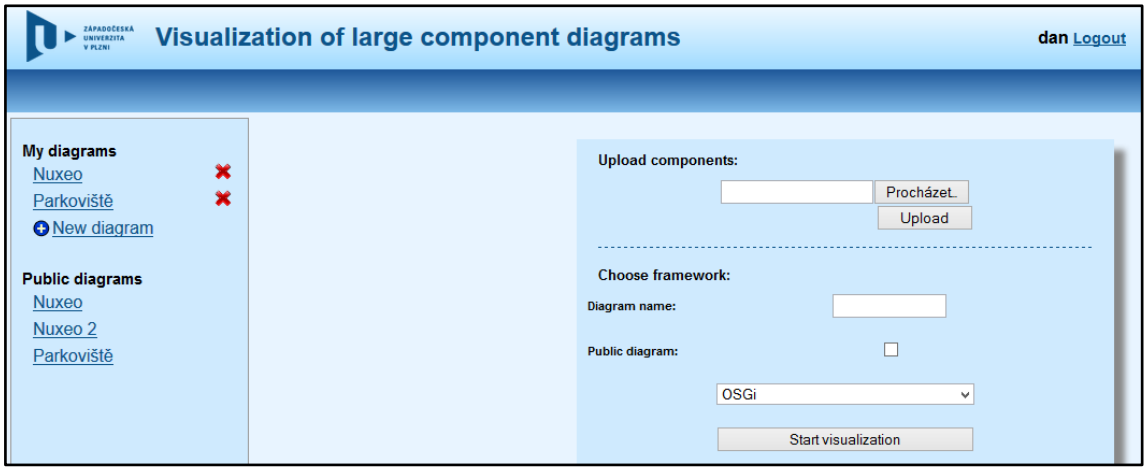
Poté je pomocí JavaScriptu postupně vykreslován diagram a z objektu typu JSON jsou získávány pozice pro jednotlivé komponenty. Tyto pozice se při tvorbě SVG rovnou nastaví danému elementu.

Pokud diagram nemá nastavené žádné pozice komponent, poté jsou pozice komponent náhodně vygenerovány. Tento postup náhodného generování pozic komponent se využívá také při každém zobrazení diagramu, který není uložen v databázi.

4.6 Sdílení diagramů

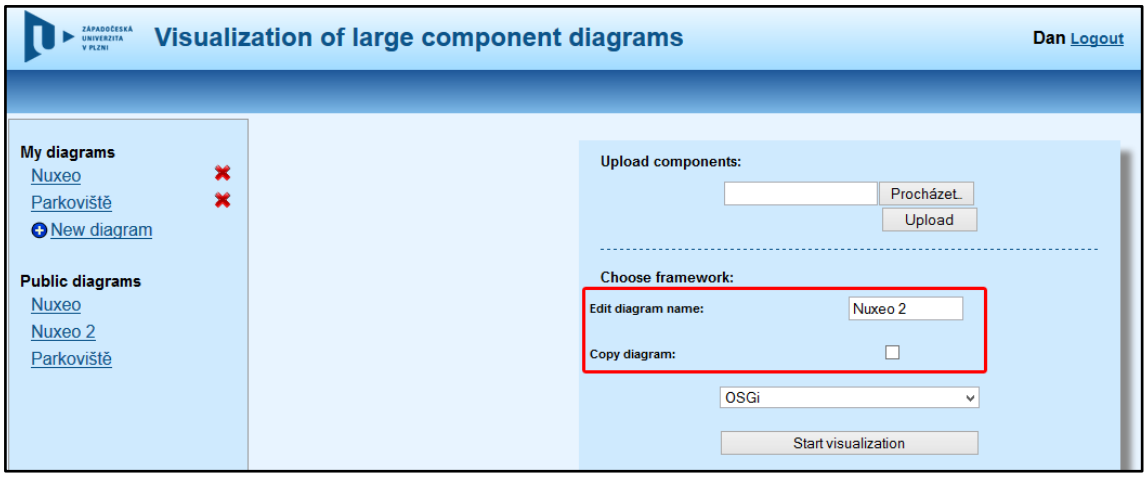
Pro snazší zobrazení diagramů známých open source programů, jako je např. Eclipse byla navržena a implementována funkce sdílení diagramů. Sdílení diagramů lze nastavit pomocí volby „*Public diagram*“. Tato volba se zobrazí přihlášenému uživateli při nahrávání nového diagramu nebo při zobrazení již nahraného diagramu (Obr. 4.9).

Ke každému diagramu lze nastavit název, pod kterým bude diagram viditelný v levém menu. Tento název se nastavuje při nahrávání komponent na server. Pokud se název diagramu nenastaví, diagram nebude trvale uložen. Pokud uživatel při nahrávání zadá název diagramu, diagram se automaticky uloží a poté bude viditelný v levém menu v části *My diagrams*. V této části je také možnost vytvořit úplně nový diagram pomocí odkazu *New diagram*.



Obr. 4.9 Obrazovka s možností nahrání diagramu komponent

V levém menu v části *Public diagrams* jsou zobrazeny diagramy, které mají nastavené veřejné zobrazení. Při kliku na některý diagram bude uživatel přesměrován na stránku, kde si může diagram zkopírovat do vlastních diagramů a nastavit mu vlastní název (Obr. 4.10). Pokud si uživatel zobrazí diagram, bude mít možnost manipulovat s komponentami v diagramu a poté si uložit vlastní rozvržení komponent na stránce.



Obr. 4.10 Obrazovka s možností kopie sdíleného diagramu

4.6.1 Implementace sdílení diagramů

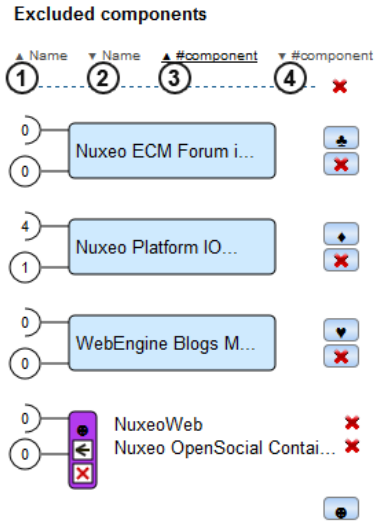
Každý uložený diagram v databázi obsahuje atribut *public*. Pokud je tato hodnota jedna, poté je veřejný a je sdílen s ostatními uživateli. Pokud hodnota public je rovna nule, diagram je nastaven jako soukromý a žádný jiný uživatel ho nemůže upravit.

Pokud si uživatel zkopíruje veřejný diagram, vytvoří se v databázi nový záznam s nastaveným názvem a pozicemi komponent z veřejného diagramu, který je kopírován. Dále jsou také zkopírovány všechny soubory, které jsou uložené v adresáři na serveru do nového adresáře vytvořeného k novému diagramu. Název nového adresáře je vytvořen dle vzoru *<session_id>_<diagram_id>*.

4.7 Řazení komponent v pravém sloupci

Do SeCo můžeme přidávat mnoho komponent či seskupených komponent. Komponenty jsou do SeCo zařazovány postupně nakonec seznamu. Pokud v pravém sloupci bylo již mnoho komponent, nastávala situace, že se velmi špatně dohledávala hledaná komponenta.

Z důvodu snadného vyhledání komponent v pravém sloupci byla navržena funkce řazení komponent (viz Obr. 4.11). Na Obr. 4.11 jsou vyznačené možnosti řazení. Možnost řazení číslo 1 seřadí prvky v pravém sloupci dle jména vzestupně, číslo 2 seřadí prvky v pravém sloupci dle jména sestupně. Možnost řazení číslo 3 seřadí prvky dle počtu komponent vzestupně. Poslední možnost (číslo 4) seřadí prvky v pravém sloupci dle počtu komponent sestupně.



Obr. 4.11 Pravý panel - funkce řazení komponent

4.7.1 Implementace řazení komponent v pravém sloupci

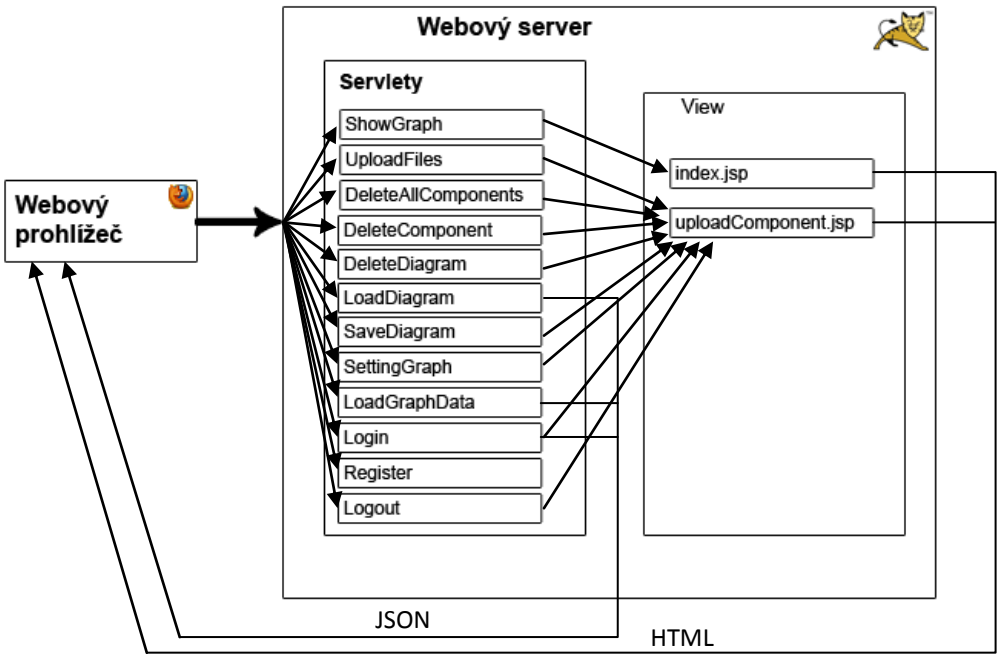
Po kliku na odkaz pro seřazení komponent v pravém sloupci je zavolána *JavaScriptová* funkce dle případného řazení (*sortByNameASC*, *sortByNameDESC*, *sortByCountComponentASC*, *sortByCountComponentDESC*). V této funkci jsou postupně zjištěny všechny prvky, které obsahuje SeCo a jsou seřazeny přímo v *JavaScriptu*. Neprobíhá zde žádný požadavek na server.

4.8 Komunikace klient – server

Veškeré zaslané požadavky klienta na webový server jsou předány obslužnému servletu uvedenému v konfiguračním souboru *web.xml*. Servlet poté může spustit příslušnou JSP stránku nebo pouze zpracuje předávaná data a vrátí odpověď typu JSON. V Tab. 4.4 jsou vypsány servlety, podporované HTTP metody a jejich typ odpovědi klientovi. Schéma komunikace klient – server je zobrazeno na Obr. 4.12.

Tab. 4.4 Přehled servletů v aplikaci

Servlety	HTTP metody	Odpověď
ShowGraph	GET	HTML - index.jsp
UploadFiles	GET	HTML - uploadComponent.jsp
DeleteAllComponents	GET	HTML - uploadComponent.jsp
DeleteComponent	GET	HTML - uploadComponent.jsp
DeleteDiagram	GET	HTML - uploadComponent.jsp
LoadDiagram	GET	JSON
SaveDiagram	POST	HTML - uploadComponent.jsp
SettingGraph	GET	HTML - uploadComponent.jsp
LoadGraphData	GET	JSON
Login	POST	HTML - uploadComponent.jsp
Register	POST	JSON
Logout	GET	HTML - uploadComponent.jsp



Obr. 4.12 Schéma komunikace klient - server

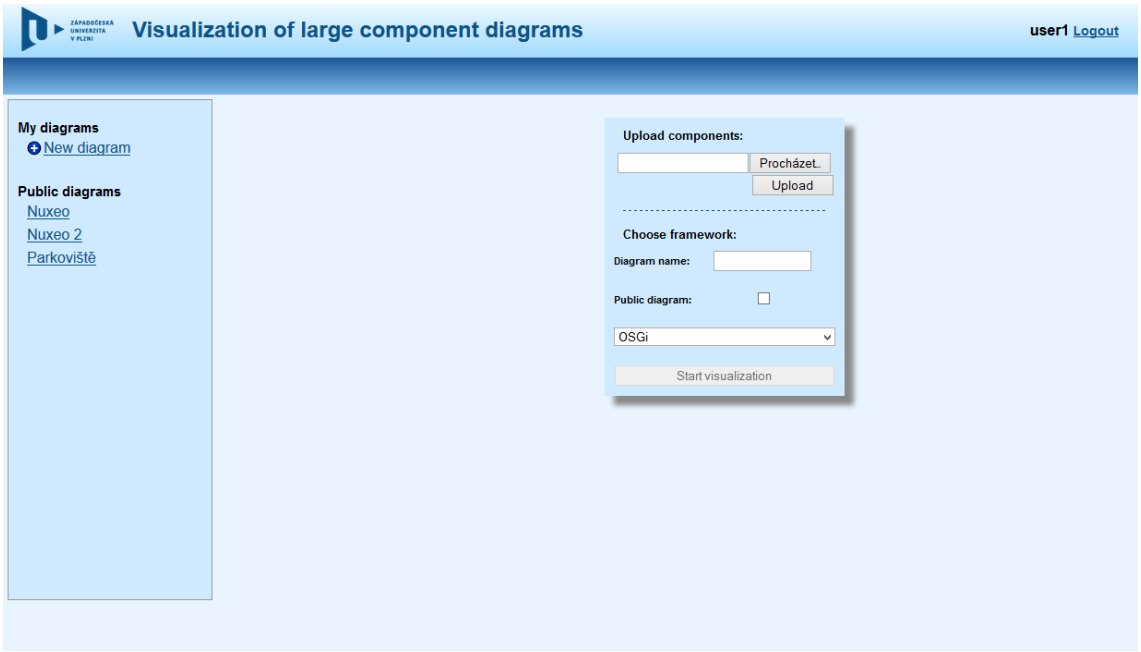
5 Demonstrace použitých technik

V této kapitole je rozepsána demonstrace implementovaných technik pro vizualizaci softwarových komponent. Jednotlivé podkapitoly jsou kroky, které musí uživatel udělat pro vytvoření diagramu a jeho následné sdílení. V ukázkách budou použity dva uživatelské účty uživatel 1 (na obrázcích je uveden „*user1*“) a uživatel 2 (na obrázcích je uveden „*user2*“).

5.1 Vytvoření nového diagramu

Aby bylo možné demonstrovat práci s diagramem komponent včetně sdílení diagramů, je nejprve nutné nahrát komponenty na server. Na Obr. 5.1 lze vidět obrazovku aplikace, kde uživatel č. 1 provede následující akce, potřebné pro nahrání a uložení diagramu:

- Nahraje komponenty na server – Vybere komponenty ze svého disku a klikne na tlačítko „*Upload*“
- Vloží název diagramu do textového pole „*Diagram name*“
- Zaškrtne volbu „*public diagram*“
- Stiskne tlačítko „*Start visualization*“



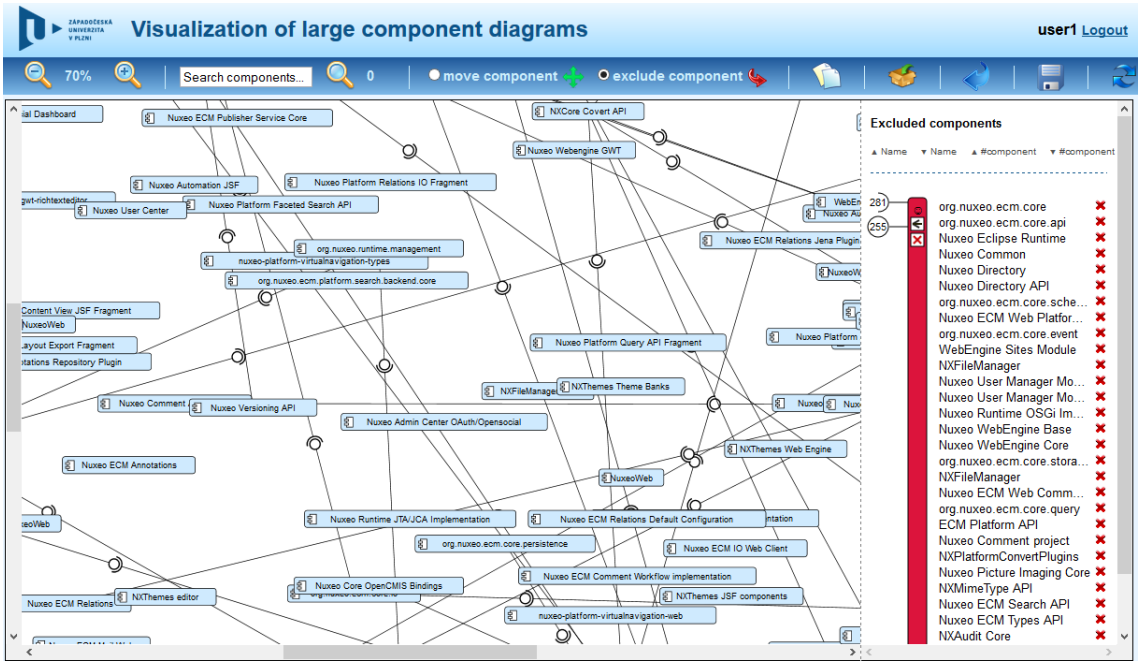
Obr. 5.1 Nahrání komponent, uložení diagramu komponent - uživatel č. 1

Na Obr. 5.1, v levém menu, je možné vidět nahrané diagramy komponent přihlášeného uživatele (Uživatele č. 1) a sdílené diagramy komponent, které si může uživatel zobrazit. V tomto případě byl uživatelem nahrán diagram aplikace *Nuxeo* a diagram byl pojmenován „*Nuxeo – test*“.

5.2 Vizualizace diagramu

Jakmile je stisknuto tlačítko *„Start visualization“* (viz Obr. 5.1), je zobrazena stránka se zobrazeným diagramem komponent (viz Obr. 5.2). V tmavě podbarveném menu může uživatel vyvolat následující činnosti:

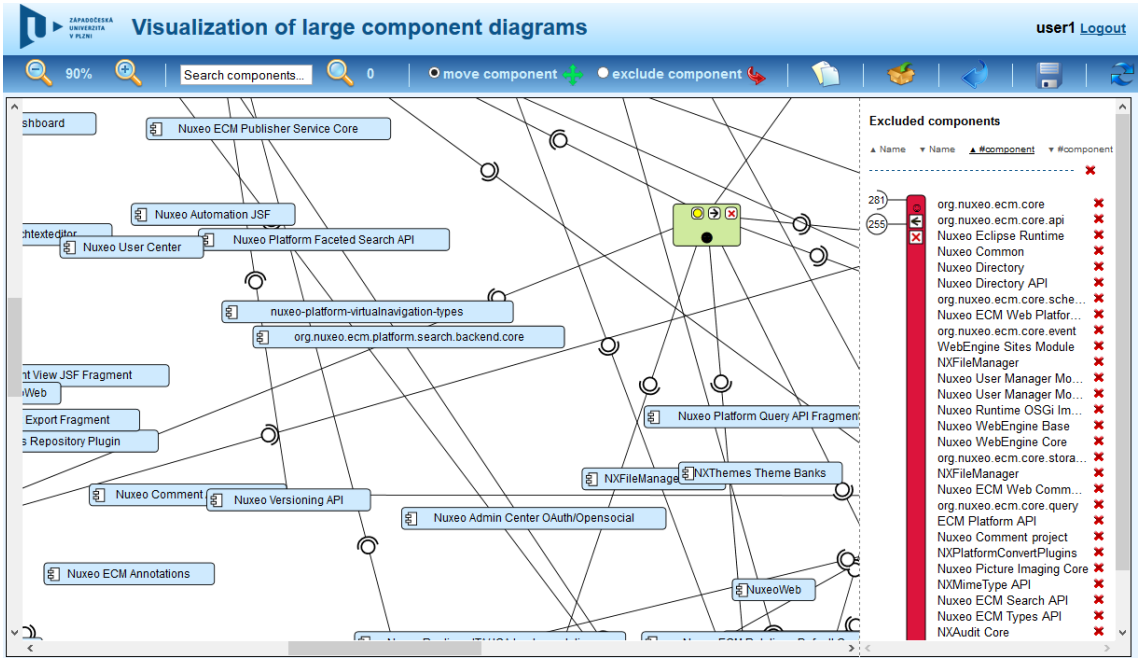
- Návrat na předchozí stránku (ikonka s modrou šipkou)
- Uložení pozic komponent (ikonka s disketou)
- Načtení pozic komponent (ikonka obsahující dvě modré šipky)



Obr. 5.2 Vizualizace diagramu komponent – uživatel č. 1

Uložení pozic komponent umožňuje uživateli zobrazit později stejné rozvržení komponent diagramu, které si dříve uložil. Po uložení komponent je uživateli zobrazena zpráva o úspěšném uložení. Při každém zobrazení diagramu jsou komponenty načteny na stejných pozicích, jaké uživatel uložil.

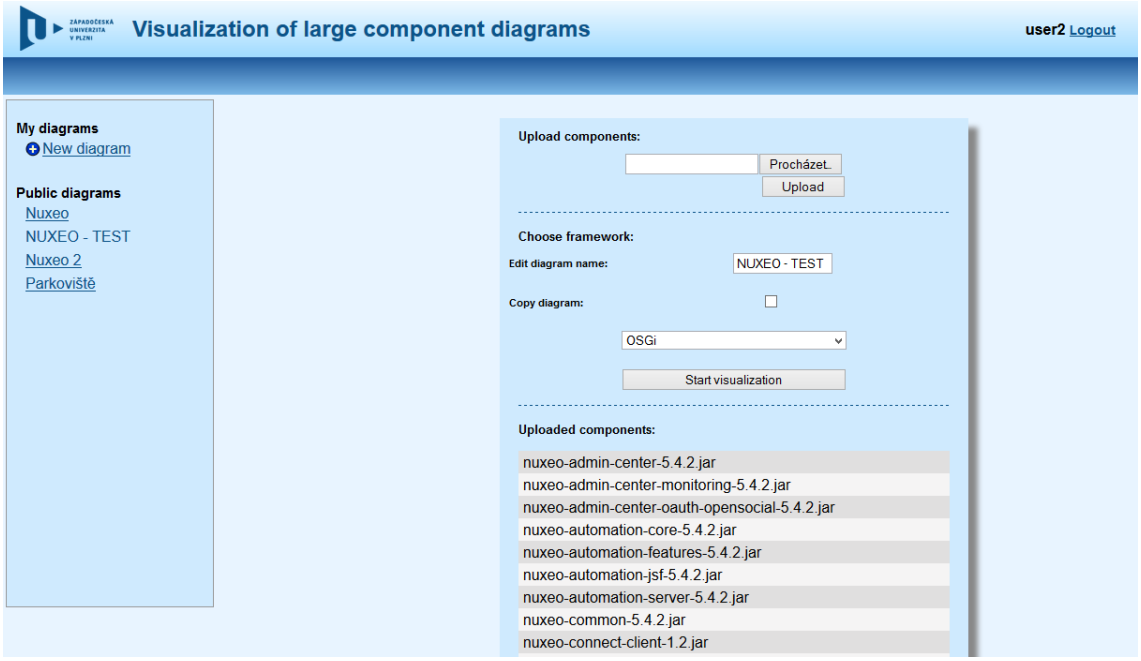
Speciálním případem při načítání pozic komponent v diagramu jsou seskupené komponenty. U těchto seskupených komponent jsou načteny pozice jednotlivých komponent ve skupinách, nikoliv pozice skupiny komponent. Při znovu načtení diagramu jsou komponenty načteny samostatně a netvoří skupinu komponent. Seskupené komponenty lze vidět na Obr. 5.3.



Obr. 5.3 Skupina komponent v diagramu je znázorněna zeleným obdélníkem

5.3 Práce se sdíleným diagramem

V předchozích kapitolách (5.1 a 5.2) bylo demonstrováno vytvoření sdíleného diagramu *uživatelem č. 1*. V této kapitole je přihlášen *uživatel č. 2*, který bude pracovat se sdíleným diagramem *„Nuxeo - test“*, tento diagram byl sdílen uživatelem č. 1.

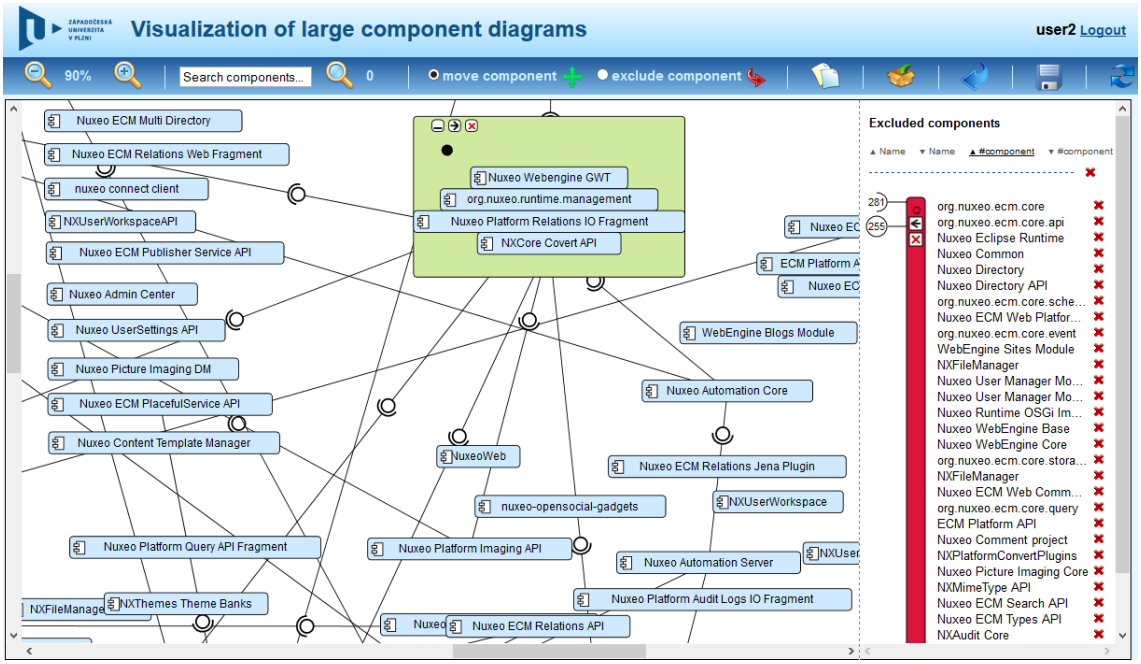


Obr. 5.4 Výběr sdíleného diagramu "Nuxeo - test" - uživatel č. 2

Na Obr. 5.4 je zobrazena obrazovka přihlášeného uživatele č. 2, který klikl v levém menu na veřejný diagram *„Nuxeo – test“*. Uživatel má následující možnosti:

- Změnit název diagramu – textové pole *„Edit diagram name“*
- Vytvořit kopii sdíleného diagramu a uložit ji pod svůj uživatelský účet – možnost *„Copy diagram“*

Při zaškrtnutí volby *„Copy diagram“* a následném stisku tlačítka *„Start visualization“* je uživateli vytvořena kopie sdíleného diagramu. V této kopii má uživatel právo trvale odstranit komponenty z diagramu, přidat své komponenty do diagramu nebo může uložit své rozvržení zobrazených komponent. Uživateli se tímto postupem zobrazí zkopírovaný diagram v levém menu v části *„My diagrams“.*



Obr. 5.5 Rozvržení komponent uživatele č. 2, zelený obdélník představuje rozšířenou skupinu komponent v diagramu

5.3.1 Skupina komponent

Skupina komponent je v diagramu znázorněna zeleným obdélníkem (Obr. 5.3; detail lze vidět Obr. 4.4). Skupina komponent se může nacházet ve třech následujících stavech:

- Rozšířená skupina komponent (v diagramu: Obr. 5.5; detail: Obr. 4.4 značka 1)
- Minimalizovaná skupina komponent (v diagramu: Obr. 5.3; detail: Obr. 4.4 značka 5)
- Skupina komponent v SeCo (Obr. 5.3 oblast SeCo)

Se skupinou komponent lze provádět následující akce:

- Rozšíření skupiny komponent (detail je na Obr. 4.4 značka 6)
- Přesun skupiny komponent do SeCo (detail je na Obr. 4.4 značka 3)
- Odstranění skupiny komponent (detail je na Obr. 4.4 značka 4)
- Minimalizace skupiny komponent (detail je na Obr. 4.4 značka 2)

6 Instalace aplikace, nastavení

V této kapitole je rozepsán postup instalace serveru Apache Tomcat, vizualizačního nástroje CoCA-Ex a MySQL databáze. Dále je zde také popsán postup nastavení konfiguračních souborů aplikace.

6.1 Instalace Apache Tomcat a nástroje CoCA-Ex

Nejdříve je nutné nainstalovat webový server, na kterém bude běžet aplikace. Aplikace je určena pro server Apache Tomcat, lze jej nalézt na přiloženém CD v adresáři Instalace nebo stáhnout z webových stránek http://tomcat.apache.org/download-70.cgi.

Jakmile bude Apache Tomcat nainstalován, provedeme instalaci aplikace zkopírováním souboru *VisualizationTool.war* z přiloženého CD do adresáře (platí pro OS Windows) *C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps*. Pokud je Apache Tomcat nainstalován v jiném adresáři, přizpůsobíme tomu začátek cesty.

Minimální požadavky pro správný běh aplikace

- JRE 1.6 nebo vyšší verze
- Windows Vista, 7, 8
- Firefox 11.0

Po dokončení instalace včetně databáze MySQL (kap. 6.2) spustíme aplikaci zadáním URL adresy *localhost:8080/VisualizationTool* do adresního řádku prohlížeče.

6.2 Instalace databáze MySQL

Databázi MySQL lze instalovat buď samostatně, nebo současně se serverem Apache a modulem PHP. Instalace současně s webovým serverem je určena pro začínající uživatele a je rychlejší než instalace samotného MySQL. Součástí tohoto balíku je často také webová aplikace PHPMyAdmin, která slouží ke správě databáze MySQL. Jako instalační balík je možné použít například EasyPHP (*http://www.easyphp.org/*), instalace tohoto programu je záležitostí několika minut. Při instalaci je nutné si zapamatovat uživatelské jméno a heslo pro přístup do MySQL databáze. Toto heslo se musí poté zadat do konfiguračního souboru aplikace (kapitola 6.3).

Před prvním spuštěním aplikace je nutné spustit SQL dotazy v databázi, které vytvoří databázové tabulky. SQL dotazy jsou uložené v souboru *create_table.sql* na přiloženém CD.

6.3 Konfigurační soubor

V aplikaci je nutné nastavit cestu k adresáři, do kterého se budou nahrávat uložené soubory. Tato cesta je uložena v souboru *WEB-INF/configuration.properties* jako parametr *storageLocation*.

Dále je nutné v aplikaci nastavit soubor *web.xml*, který je uložen v adresáři *WebContent/WEB-INF/*.

V souboru *web.xml* je nutné nastavit následující parametry:

DbUrl Url adresa, na které běží databáze MySQL.
DbName Název databáze.
DbUser Jméno uživatele databáze.
DbPsw Heslo uživatele databáze.
HOME_URL Url adresa, na které bude běžet aplikace. Adresa musí končit lomítkem.

Ukázka nastavení parametrů:

```
<context-param>
   <param-name>DbUrl</param-name>
    <param-value>jdbc:mysql://127.0.0.1/</param-value>
</context-param>
<context-param>
   <param-name>DbName</param-name>
    <param-value>visualization_tool</param-value>
 </context-param>
 <context-param>
   <param-name>DbUser</param-name>
    <param-value>root</param-value>
 </context-param>
 <context-param>
   <param-name>DbPsw</param-name>
   <param-value>heslo</param-value>
 </context-param>
 <context-param>
   <param-name>HOME_URL</param-name>
   <param-value>http://localhost/VisualizationTool/</param-value>
</context-param>
```

7 Závěr

Cílem této diplomové práce bylo rozšířit stávající aplikaci pro vizualizaci diagramů komponent o funkce, které by urychlily a usnadnily práci s aplikací. Nejdříve jsem v diplomové práci nastudoval techniky pro vizualizaci rozsáhlých diagramů komponent. Podrobněji jsou tyto techniky popsané v kapitole 2.5.7, kde je také popsána aplikace CoCA-Ex.

V diplomové práci bylo navrženo rozšířit stávající aplikaci o možnost seskupit komponenty a tím zpřehlednit zobrazený diagram. Skupinu komponent je možné standardně posouvat v diagramu či zobrazit komponenty ve skupině. Při seskupení komponent zůstanou zachovány všechny vazby seskupených komponent s ostatními komponentami. I po seskupení komponent zůstala zachována možnost zobrazit požadovaná a poskytovaná rozhraní daných komponent.

Dále bylo do aplikace navrženo a implementováno rozšíření o podporu uživatelských účtů. Přidáním uživatelských účtů do aplikace vznikly další možnosti pro rozšíření aplikace, které byly implementovány. Přihlášený uživatel má možnost do aplikace nahrát několik svých diagramů. V každém diagramu má poté možnost uložit název diagramu a rozmístění komponent v grafu. Při pozdějším znovunačtení diagramu uživatel nemusí znovu posunovat komponenty, aby byl diagram přehledný.

Další důležitou funkcí byla možnost nastavení sdílení diagramů s ostatními uživateli. Funkce sdílení byla navržena z důvodu velkých softwarových balíků, se kterými pracuje více uživatelů např. *Eclipse Equinox*. Přihlášený uživatel má poté možnost zkopírovat tento sdílený diagram a přidat do něj další komponenty. Diagram lze poté uložit jako svůj diagram pod svým názvem, který již nebude sdílen s ostatními uživateli.

Navržená a implementovaná rozšíření jsou rozepsána v kapitole 4, kde jsou zároveň ukázky jednotlivých funkcí a možnost uživatelské práce s těmito rozšířeními. V kapitole 5 je demonstrována práce s implementovanými technikami z pohledu uživatele.

Literatura

[1] Úvod do komponent. *ZČU KIV Wiki.* [Online] 23. 3. 2011 [cit. 15. 4. 2013] http://wiki.kiv.zcu.cz/UvodDoKomponent/HomePage.

[2] Pavlíková, Jindra. *Vizualizace rozsáhlých diagramů komponent a interakce s nimi.* 2012.

[3] Bachmann, Felix. *Volume II: Technical Concepts of Component-Based Software Engeneering,2nd edition.* 2000.

[4] *OSGi Architecture.* http://www.osgi.org/Technology/WhatIsOSGi. [Online] [cit. 18. 5. 2013]

[5] *Enterprise Java Beans.* http://kore.fi.muni.cz/wiki/index.php/Enterprise_JavaBeans. [Online] 8. 5. 2013 [cit. 20. 5. 2013].

[6] HTML 5. *W3.org.* [Online] http://www.w3.org/TR/html5/ [cit. 2. 4. 2013].

[7] CSS Introduction. *W3Schools.* [Online] http://www.w3schools.com/css/css_intro.asp [cit. 10. 5. 2013].

[8] jQuery. *jQuery javascript library.* [Online] http://jquery.com/ [cit. 10. 5. 2013].

[9] Jaroslav, Šnajberk a Přemysl, Brada. ComAV - A Component Application Visualization Tool - Use of Reverse Engineering and Interactivity in Visualization for Component Software Comprehension.

[10] Andy Cockburn, Amy Karlson, Benjamin B. Bederson. *A Review of Overview+Detail, Zooming, and Focus+Context Interfaces.* 2007.

[11] T.J. Jankun-Kelly, Kwan-Liu Ma. *MoireGraphs: Radial Focus+Context.* Mississippi State University, University of California, 2003.

[12] HOLÝ, Lukáš a BRADA, Přemysl. *Lowering Visual Clutter in Large Component diagrams.* Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2012.

[13] *Oracle EJB.* http://www.oracle.com/technetwork/java/javaee/ejb/index.html. [Online] [cit. 3. 6. 2013]

Příloha A - Zdrojové soubory aplikace

V této příloze je sepsán seznam zdrojových souborů aplikace. U souborů, které již v aplikaci byly je jejich popis převzat z [2].

Java balíky a třídy

- cz.zcu.kiv.offscreen.api
  - EdgeInterface.java – rozhraní definující, nutný obsah každé hrany v grafu
  - GraphInterface.java – rozhraní definující, nutný obsah grafu
  - VertexInterface.java – rozhraní definující, nutný obsah uzlu grafu

- cz.zcu.kiv.offscreen.graph
  - EdgeImpl.java – třída reprezentující hranu, která implementuje rozhraní EdgeInterface
  - GraphImpl.java – třída reprezentující graf, který implementuje rozhraní GraphInterface
  - VertexImpl.java – třída reprezentující uzel grafu, který implementuje rozhraní VertexInterface
  - GraphExport.java – třída, která zajistí zjednodušení struktury grafu, aby mohla být data převedena do formátu JSON

- cz.zcu.kiv.offscreen.graph.creator
  - GraphMaker.java – třída, která zajišťuje vytvoření grafu na základě informací získaných z načtení komponent

- cz.zcu.kiv.offscreen.graph.loader
  - GenericComponentLoader – třída, která je oděděna od třídy ComponentLoader a zajišťuje získání dat z načtených komponent

- cz.zcu.kiv.offscreen.loader.configuration
  - ConfigurationLoader.java – třída zajišťující získání cesty pro ukládání adresářů jednotlivých uživatelů ze souboru configuration.properties

- cz.zcu.kiv.offscreen.servlets
  - LoadGraphData.java – servlet zajišťující odeslání struktury grafu ve formátu JSON klientovi
  - Login.java – servlet, který přihlásí uživatele
  - Logout.java – servlet, který odhlásí uživatele
  - Register.java – servlet, který zaregistruje uživatele
  - SettingGraph.java – servlet, který zajistí vytvoření adresáře na základě získané session id z cookies a id diagramu

    - ShowGraph.java – servlet, starající se o zobrazení stránky aplikace, která obsahuje vytvořený graf komponent

- cz.zcu.kiv.offscreen.servlets.actions
    - DeleteAllComponents.java – servlet, který smaže veškeré nahrané komponenty na serveru, dle session id
    - DeleteComponent.java – servlet, odstraní nahranou komponentu na serveru dle zadaných parametrů
    - DeleteDiagram.java – servlet, který odstraní vytvořený diagram, dle hash kódu a id diagramu
    - LoadDiagram.java – servlet, který načte uložený diagram dle hash kódu a id diagramu
    - SaveDiagram.java – servlet, který uloží diagram
    - UploadFiles.java – servlet, který se stará o nahrání komponent na server

- cz.zcu.kiv.offscreen.session
    - SessionManager – třída obstarávající odstranění nahraných komponent po vypršení session

- cz.zcu.kiv.offscreen.storage
    - FileManager.java – třída obsahující metody, které zajišťují vytváření uživatelského adresáře, ukládání a mazání komponent a diagramů

- cz.zcu.kiv.offscreen.user
    - DB.java – třída, která komunikuje s databází, přes tuto třídu se zpracovávají veškeré sql dotazy
    - Diagram.java – třída, ve které se vytvářejí, načítají a ukládají parametry diagramu, dále se zde také načítají seznamy veřejných a uživatelských diagramů
    - User.java – třída, ve které vytvářejí, načítají a ukládají uživatelé
    - Util.java – třída, ve které jsou uložené užitečné funkce využívané napříč aplikací, například funkce pro vytvoření md5 hashe.

JavaScriptové soubory

JavaScriptové soubory jsou umístěné v adresáři *js*.

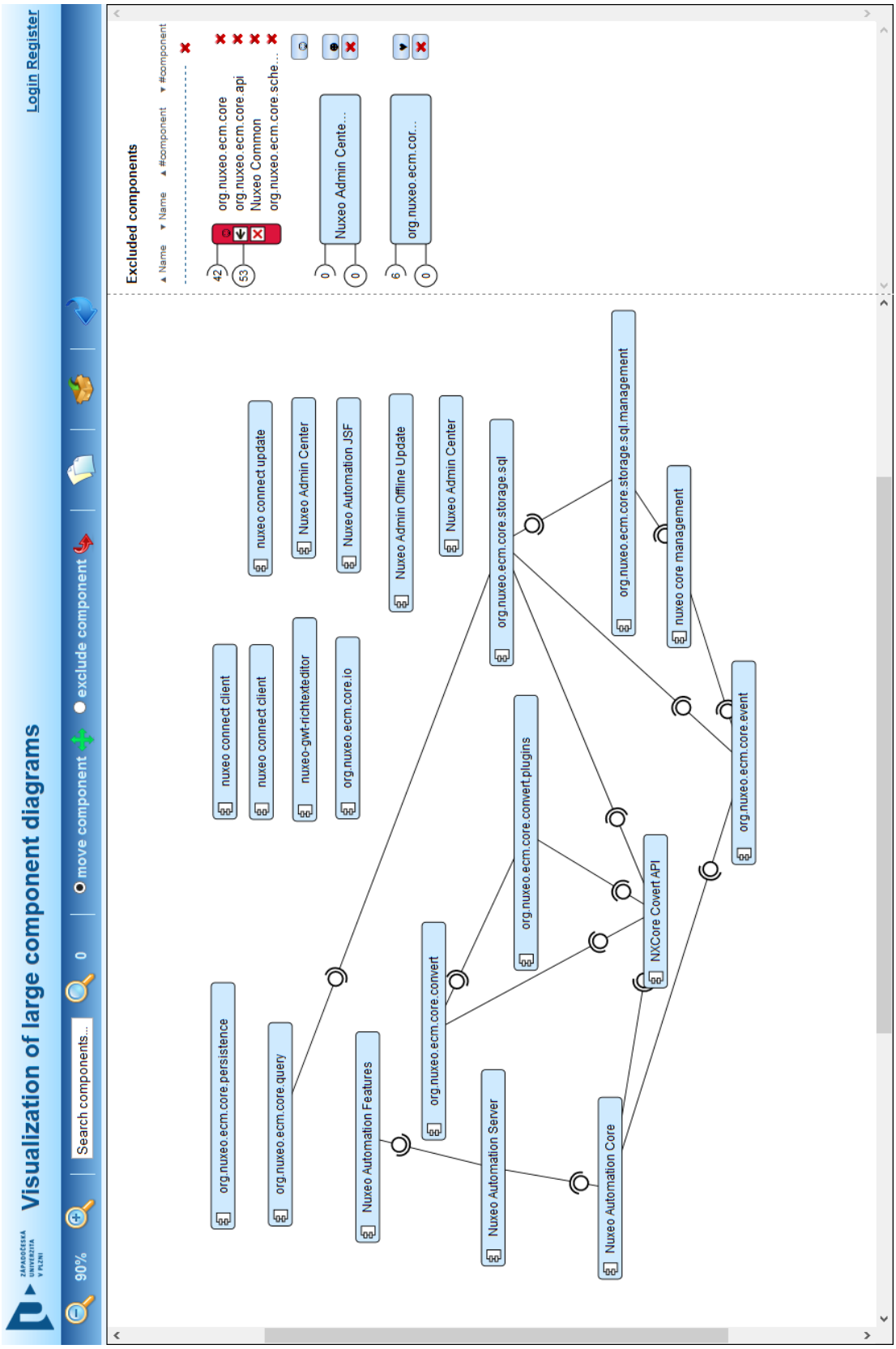
diagram.js – soubor obsahující funkce potřebné pro načítání, ukládání, odstranění diagramu
graphManager.js – reprezentuje graf, který načte data ze serveru a vytvoří graf ve formátu svg, který zobrazí na stránce
gridMarg.js - zajišťuje vytváření známek (delegátů) a jejich řazení u komponenty
group.js – zastupuje skupinu a obsahuje funkce pro přidávání a mazání prvků skupiny
groupManager.js – správce vytvořených skupin komponent
loader.js – zajišťuje zobrazení a skrytí JavaScriptového loaderu
main.js – zajišťuje pohyb s komponentami v grafu
mark.js – reprezentuje známku (symbol), který je přiřazován vyjmutým komponentám
markSymbol.js – vytváří symboly (znak + barva)
offScreenKiv.js – obsahuje implementaci off-screen techniky CoCA-Ex
tooltips.js – registrace veškerých tooltipů v aplikaci
util.js – obsahuje pomocné funkce
zoom.js – obsahuje funkce pro přiblížení a oddálení diagramu komponent
jquery-1.7.1.js – knihovna jQuery
jquery.contextMenu.js – plugin, který slouží k vytvoření kontextového menu
jquery.qtip.js – plugin k vytváření tooltipů
mootools-core-1.4.1.js – plugin, který umožňuje vytvářet třídy v jazyku JavaScript
spin.js – plugin pro vytváření loaderů

Soubory se styly

Soubory s css styly jsou umístěné v adresáři *styles*.

basic.css – definované styly pro celou aplikaci
jquery.contextMenu.css – styly používané v pluginu jquery.contextMenu.js
jquery.qtip.css – styly doplňující plugin jquery.contextMenu.js
tooltips.css – styly přepisující některé styly z jquery.qtip.css

Příloha B – Náhled aplikace



Příloha C – Náhled aplikace, přihlášený uživatel