Posudek oponenta diplomové práce Clustering of sofware modules using Jaya algorithm

Student:	Bc. Jakub Pavlát
Oponent dip. práce:	doc. Ing. Zdeněk Míkovec, Ph.D., FEL, ČVUT

Téma

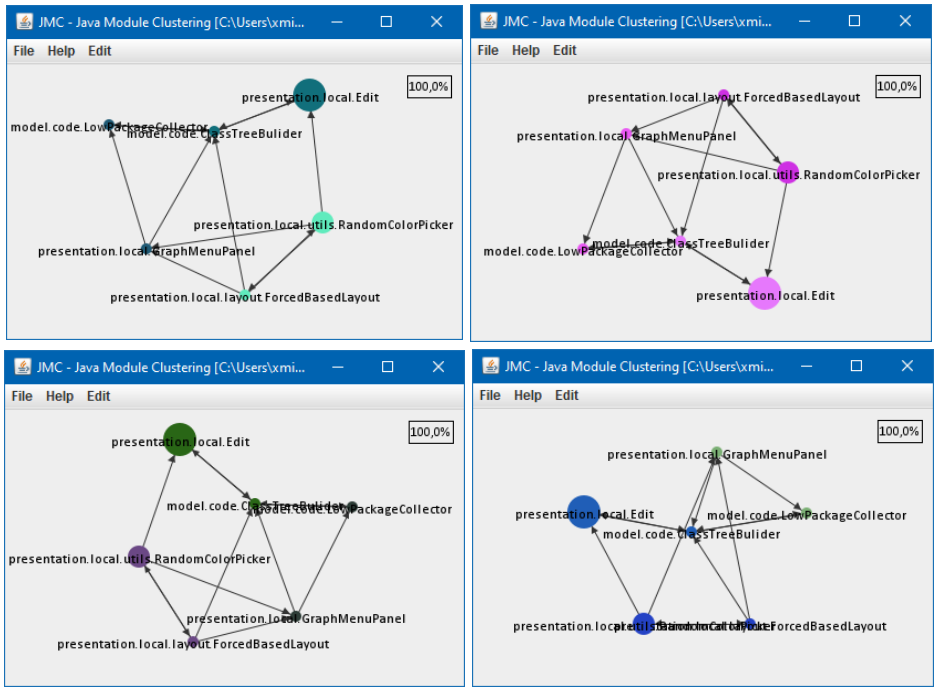
Cílem diplomové práce bylo navrhnout a implementovat softwarový nástroj pro klastrování softwarových modulů. Vedle samotné analýzy zdrojového kódu a klastrování bylo cílem vhodně vizualizovat vzniklý graf závislosti s klastry.

Řešení

V úvodní kapitole (kap. 1) student srozumitelným a přehledným způsobem představuje fungování vytvářeného nástroje (*pipeline of tasks*). Z pro mě nejasných důvodů se zde však nachází analýza počtu developerů a matice odpovědnosti (kap. 1.1). Dále zbytečně podrobně popisuje technologie jako nástroj Maven, jazyk XML, nebo datový formát JSON. Místo čtyř stran obecného popisu by stačilo několik odstavců, kde by se zejména popisovalo, jak budou tyto technologie použity v práci a proč. Toto zdůvodnění však v textu chybí. Student zde také představuje vlastní datovou strukturu pro uložení grafu závislostí a klastrů. Jedná se však pouze o odkaz do přílohy bez bližšího vysvětlení, co a proč tato struktura obsahuje. Na navržené struktuře je problematické používání popisků jako identifikátorů. Vhodnější řešení je zavedení unikátního identifikátoru pomocí *xsd:ID* pro elementový typ *node* a jeho referencování v ostatních elementových typech (*arc*, *nodelabel*) pomocí *xsd:IDREF*. Tím by byla zaručena unikátnost identifikátorů, validní referencování a také lepší srozumitelnost schématu pro stroj i člověka.

V kapitole Existující nástroje (kap. 2) student analyzuje řadu nástrojů pro analýzu Java kódu a popisuje kritéria pro finální výběr. Na tomto místě by byla vhodná srovnávací tabulka pro získání lepšího přehledu o naplňování zvolených kritérií ze strany analyzovaných nástrojů. Při výběru řešení pro vizualizaci grafu si student pro analýzu bez podrobnější argumentace vybírá JavaScriptovou knihovnu D3.js, přestože je jazyk JavaScript v daném kontextu evidentně nevhodnou volbou. Druhý analyzovaný nástroj je Java knihovna JGraphT. To ale není nástroj pro vizualizaci ale pro analýzu grafů a tak je zřejmé, že pro účely vizualizace není vhodným nástrojem. Problematice klastrování modulů je věnován pouze jeden odstavec, kde student konstatuje, že se hlouběji nebude zabývat pracemi, které jsou na seznamu doporučené literatury ([24, 25]).

V kapitole Uživatelské rozhraní (kap. 3) se student zaměřuje na vizualizaci vytvořeného MDG grafu. Definuje si vlastnosti vizualizace grafu tak, aby byl pro člověka co nejpřehlednější. Analyzuje řadu přístupů, a nakonec volí *force-directed* rozložení. Opět zde postrádám podrobnější popis vlastností různých přístupů a jejich porovnání vzhledem ke zvoleným kritériím. S ohledem na očekávané velké množství uzlů a hran zde postrádám vizuální řešení pro slučování uzlů a hran (například operace kondenzace silných komponent v grafu). Ukázka vizualizace grafu závislosti na obrázku 3.3 demonstruje řadu problematických momentů. Popisky se navzájem překrývají, zasahují do uzlů a hran a tím se výrazně snižuje jejich čitelnost, ale i přehlednost celé vizualizace. Popisky také nemají definovanou žádnou „bezpečnou oblast“ tak, aby při překryvu s jinými objekty byly stále čitelné. Vizualizace klastrů je velmi nedostatečná. Sestává se pouze z obarvení uzlů. Náhodný výběr barev je nešťastný, protože jsou často zvolené barvy velice blízké a lze je od sebe jen těžko odlišit. Při testování aplikace jsem se velice rychle do výše zmíněných situací dostal (viz ukázky vizualizace stejného grafu se třemi klastry na Obr. 1 níže). Postrádám například snahu o prostorové uspořádání, které by uzly z jednoho klastru umístilo blíže k sobě ve srovnání s uzly z jiných klastrů.



Obr. 1. Různé vizualizace jednoho grafu závislosti s třemi klastry.

V kapitole Analýza kódu (kap. 4) student detailně a srozumitelně popisuje způsob tvorby stromu tříd (ClassTree) a abstraktního syntaktického stromu. Je zřejmé, že této části své práce věnoval student největší úsilí a dosáhl velmi solidních výsledků. V navazující kapitole Problém klastrování (kap. 5) pak student popisuje úspěšnou a netriviální implementaci algoritmu Jaya.

V kapitole Testování (kap. 6) student popisuje unit testy a testy použitelnosti. Unit testy jsou popsány jen povrchně a obecně. O úspěšnosti unit testů neexistuje v práci žádná zmínka. Popis testu použitelnosti neobsahuje informaci o velikosti grafu závislosti, se kterým účastníci testu pracovali. V seznamu úkolů chybí takový, který by vybízel k vizuální analýze grafu s klastry, což mohlo vést k nedostatečnému testování této části vytvořeného nástroje. Výsledky testů použitelnosti jsou prezentovány v extrémně redukované podobě (úspěch-neúspěch) bez jakýchkoliv dalších komentářů a nálezů z pozorování provádění úkolů. Student se pouze krátce zmiňuje o poznatcích získaných z rozhovorů po skončení testu.

Text práce je napsán anglicky, což oceňuji. Text je však místy napsán poněkud kostrbatě, což snižuje čitelnost některých pasáží. Práce je příliš stručná a neobsahuje detailní popis některých důležitých částí práce (viz připomínky výše). Z formálního hlediska nedosahuje předepsaného rozsahu (min. 40 stran). Řada obrázků je nečitelná (např. obrázky 1.1., 1.2, 2.1, 5.1) a to i v elektronické podobě. Některé odkazy na webové zdroje nefungují (např. odkazy 2, 3).

Závěr

K diplomové práci mám výhrady k textovému zpracování a ke způsobu splnění úkolu vizualizace grafů. Tuto část zadání považuji za pouze částečně splněnou.

Práci hodnotím známkou **D (uspokojivě)**.

V Praze dne 27. 8. 2020

doc. Ing. Zdeněk Míkovec, Ph.D.