Posudek diplomové práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Bc. Peter Fačko
Název práce	Package manager for C++
Rok odevzdání	2024
Studijní program	Informatika **Studijní obor** Softwarové a datové inženýrství

Autor posudku	Mgr. Jiří Klepl **Role** oponent
Pracoviště	Katedra distribuovaných a spolehlivých systémů

Text posudku:

Práce se zabývá návrhem a implementací správce balíčků pro jazyk C++. Konkrétně se jedná o nástroj, který sjednocuje funkcionalitu systémového správce balíčků a správce dependencí pro jazyk C++. Výsledný nástroj má za cíl usnadnit distribuci programů napsaných v jazyce C++ a abstrahovat specifika jednotlivých správců balíčků. K demonstraci funkcionality využívá již zavedeného správce balíčků *pacman* a správce dependencí pro C++ *Conan*. Úvod práce diskutuje relevantnost zavedení správce balíčků pro jazyk C++ a zdůrazňuje nedostatečnost existujících nástrojů (systémových správců balíčků a správců dependencí) pro distribuci programů. Konec úvodu je věnován formalizaci cílů práce do jednoduchých bodů, které přesahují rámec původního zadání.

V první kapitole práce analyzuje fungování dostupných správců balíčků a podává detailní rozbor zejména nástrojů pacman a Conan. Tato analýza je zakončena výčtem různých specifik návrhu existujících správců balíčků, kde autor předkládá relevantní problémy, řešení inspirované existujícími správci balíčků, a zhodnocení daného řešení.

Druhá kapitola popisuje návrh výsledného softwarového díla tak, aby pokrývalo relevantní funkcionalitu nástrojů pacman a Conan. Značná část této kapitoly se věnuje problematice rezoluce dependencí kombinující balíčky z různých správců balíčků a v rámci nich z různých repozitářů.

Třetí kapitola přímo navazuje na druhou a popisuje formální model pro rezoluci dependencí kombinující balíčky z různých repozitářů různých správců balíčků. Ústředním bodem této kapitoly je konstrukce formule pro SAT solver, jehož výstupem je množina balíčků splňující požadované dependence a depenční konflikty.

Poslední kapitola se věnuje implementaci návrhu. Pro oba zakomponované nástroje (pacman a Conan) autor definuje překladovou abstrakci, která sjednocuje jejich specifické vlastnosti a dovoluje kombinovat balíčky z obou nástrojů. Na sjednocené abstrakci je pak implementována rezoluce dependencí pomocí SAT solveru.

Téma práce je zajímavé a aktuální, a obojí autor v práci dobře dokládá relevantními zdroji.

Práce je dobře členěná a přehledná. Anglický text práce je dostatečně srozumitelný a vhodně stylizovaný. Autor v práci vhodně definuje použité pojmy a složitější diskuze doplňuje ukázkovými příklady. Analýza je dostatečně detailní a pokrývá relevantní části tématu. Úvod a analýza problému vhodně využívají odbornou literaturu a dokumentaci existujících nástrojů. Bohužel, v poslední sekci analýzy ("Package Manager Design Patterns") tyto zdroje nejsou řádně referencovány.

Práce dosahuje vytyčených cílů a výsledné softwarové dílo demonstruje funkčnost navrženého řešení. Návrh využívá kontejnerizace a tedy je snadno přenositelný na jiné platformy a umožňuje snadné nasazení. Softwarové dílo je řádně doplněno uživatelskou dokumentací, ukázkami použití a testy, které automaticky ověřují správnost implementace za pomoci GitHub Actions.

Implementace softwarového díla je kvalitní, ale postrádá technickou dokumentaci, což významně komplikuje jeho rozšiřování pro podporu jiných správců balíčků.

Své připomínky shrnuji do následujících bodů, které by měly být adresovány u obhajoby práce:

- Práce nediskutuje možné problémy s využitím navrhovaného řešení pro jiné běžně užívané systémové správce balíčků (např. apt, dnf, zypper). Tím je částečně narušena univerzálnost navrženého řešení. Díky využití kontejnerizace toto nepovažuji za kritické.
- Z práce není zřejmá volba právě nástrojů pacman a Conan pro demonstraci funkcionality. V návaznosti na předchozí bod, pokrývají tyto nástroje očekávanou funkcionalitu zmíněných správců balíčků? Je možné reprezentovat jejich repozitáře jako u nástrojů pacman a Conan? Je možné pokrýt funkcionality jako je například *apt pinning*?
- Uživatel u vyvíjeného projektu využívajícího autorovo řešení musí specifikovat zdroj každé žádané dependence. Je možné specifikovat, že u dané dependence je možné využít dva či více zdroje? Motivací za takovou specifikací by mohlo být to, že uživatel nemá preferenci zdroje pro danou dependenci, ale nějaká jiná dependence projektu ano.

Práci doporučuji k obhajobě.

Práci nenavrhuji na zvláštní ocenění.

V Praze dne 2. 9. 2024

Podpis: