

# Posudek na diplomovou práci Štefana Šimona

## *Generování sekvenčních diagramů UML z kódu*

Cílem práce bylo navrhnout a implementovat nástroj pro reverzní inženýrství sekvenčních diagramů ve verzi UML 2.1 z programových kódů, včetně editoru vygenerovaných diagramů.

V první části práce p. Štefan představuje motivaci a cíle, které posunou generátor sekvenčních diagramů od standardních postupů směrem k detekci rámců, zejména alternativ, cyklů (včetně vnořených) a mechanismu výjimek. Střední část stručně popisuje použité řešení a diskutuje některé alternativní přístupy. V závěrečné části je popsán vlastní editor sekvenčních diagramů a jsou zmíněny některé příklady analyzovaných vstupů.

### Pozitiva:

Práce je napsaná srozumitelně, je dobře strukturována a v místech, která to vyžadují, je popis i dostatečně formálně rozepsán. Z textu je patrné, že studovanou část problematiky autor dobře chápe a že se vyzná i v technologiích, které použil při vlastní implementaci analyzátoru programového kódu.

Implementace analyzátoru i editoru je vhodně integrována do programového prostředí studia .NET, ze kterého lze provádět jednoduše všechny potřebné akce. Prakticky využitelné je i rozšíření standardu XMI, která umožňuje převést vygenerované diagramy do prostředí Enterprise Architectu.

### Negativa:

Zvolené řešení není příliš zevrubně popsáno, celý popis včetně rozepsání převzatých algoritmů a několika celostraných obrázků ve stěžejní části práce zabírá zhruba 15 stran. Popis existujících přístupů se omezuje dokonce na jeden a půl strany, to rozhodně není dostatečné.

Postup v práci vychází zejména z publikací z oblasti konstrukce překladačů [8] a [9] s tím, že je přejata i většina značení. Přesnost při popisu ovšem za rozšiřovanými pracemi zaostává, zejména v popisu vlastností GRT (*control flow graph*). V některých částech (zejména algoritmických) není zřejmé, které jsou dílem autora a které jsou pouze adaptací postupů z výše uvedených publikací.

Autor vybírá pouze dvě základní metody analýzy programového kódu – běhovou (z informací získaných z profileru) a statickou (s využitím API CCI), které však nekombinuje přestože některé známé techniky to již využívají. Dochází tak např. k situacím, že běhová analýza kódu generuje obrovský graf vzájemných volání, zatímco statická analýza rovnou ukazuje, že se v kódu vyskytuje cyklus. Naopak statická analýza provádí výpočet vždy pro všechny větve zkoumaného kódu bez ohledu na to, zda nás tyto případy zajímají či ne.

Pro praktickou použitelnost by bylo potřeba implementaci upravit tak, aby bylo možné zúžit zkoumanou část kódu. Aktuální běhový analyzátor vždy zkoumá a také generuje graf přes celou strukturu programu v nejjemnější granularitě, což je potřeba velmi zřídka. Bylo by

možné program upravit i tak, jak popisuje autor v závěru v diskuzi možných rozšíření, tj. o hierarchické strukturování diagramů – tam by ale mohl vyvstat problém celkové doby potřebné k analýze.

Závěr:

Přes výše uvedené připomínky, práce splňuje schválené zadání, a proto práci **doporučuji k obhajobě.**

oponent diplomové práce  
Mgr. Kamil Toman  
KSI MFF UK

