Posudek diplomové práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Harun Ćerim		
Název práce	Extending C# with a Library of Functional Programming Concepts		
Rok odevzdání	2020		
Studijní program	Informatika	**Studijní obor**	Softwarové a datové inženýrství

Autor posudku	Mgr. Pavel Ježek, Ph.D.	**Role**	Oponent
Pracoviště	UK MFF KDSS		

Text posudku:

Author implemented a very interesting library that allows to take advantage of the C# programming language concepts and "bend" them to extend the language with functional programming concepts. The created Funk library has a very good design, and provides useful features. The code has extensive comments, and together with the description of the library in the thesis text provide a very good overview of the implementation. All the major parts of the library are extensivelly covered by unit test. So from software engineering point of view (without context - see below) is the implementation part of the thesis a solid piece of software.
The analytical part of the thesis evaluates different approaches on how to extend C# with functional features, and shows directions where author tries to provide better features than existing libraries. What is however missing, is some reasoning about why to implement the framework from scratch and not extend some existing one - author is heavilly influenced by the FuncSharp library, that his employer is using, and he has a lot of experience with it. So why not extend FuncSharp if it is already used in production software and is open source on github?
And while author presents that his Funk library was used by his team in MFF software project STOCK, he does not provide any complex evaluation of the API designed. I would expect some complex comparision with the FuncSharp library.
In the text author briefly mentions functional concepts of the C# 8 (from September 2019) and states "we can only guess what interesting and powerful new features are coming in the future versions of C#" - however the C# 9 with more functional features will be released in November 2020, and almost final draft of these features is publically available at least since May 2020, and initial drafts and blog post in community are available since end of 2019 - so why author does not evaluate them in context of his thesis?
As author does not introduce in his thesis any strictly new ideas, but rather applies approaches used elsewhere, I would categorize the thesis as an implementation one. And here lays the major problem of the thesis - the size of the C# code is 150 kB plus additional 70 kB in unit tests including documentation comments. However a lot of the code is very repetitive - as C# lacks support for variadic templates, many generic methods and types author introduces have to come in many copies for 1, 2, 3, etc. type parameters - this is not a problem by itself, as it is a correct way how to cope with this C# language inefficiency. However it further diminishes the "real" size of the code of the thesis. And while this would be definitely enough for a bachelor thesis, it seems insufficient for an implementation master thesis.

Práci doporučuji k obhajobě.

Práci nenavrhuji na zvláštní ocenění.

Pokud práci navrhujete na zvláštní ocenění (cena děkana apod.), prosím uveďte zde stručné zdůvodnění (vzniklé publikace, významnost tématu, inovativnost práce apod.).

Datum 8.9.2020 **Podpis**