2015 International Conference on Computational Science and Computational Intelligence

# Towards an Empirical Analysis of .NET Framework and C# language Features' Adoption

Petr Capek, Erik Kral and Roman Senkerik
Faculty of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, 760 05 Zlin, Czech Republic
{capek,ekral,senkerik}@fai.utb.cz

***Abstract*— Software companies such as Microsoft periodically release new versions of the .NET framework and C# language suites. New features are introduced in each of these new releases. Currently, there is little or no empirical analysis of the adoption of .NET framework and C# programming language features and studies on how successfully those features are accepted by the community. The aim of this work is to create a set of long term statistics about the use of the new .NET and C# language features across existing popular libraries.**

***Keywords*—.NET, C#, Roslyn, code analysis, github**

## I. Introduction

So far, Microsoft has released several versions of .NET framework. In each new version, Microsoft introduced new methods and classes. Independently from .NET framework, Microsoft has also released several C# language versions.

A detailed survey exists about Java Language features [1] as well as an empirical analysis of C# generics [2] but there are no detailed empirical analyses of the adoption of NET framework and C# language features. There is also a domain-specific programming language for analyzing large scale software repositories [3], but we would like to use the Roslyn compiler rich code analysis to achieve the best possible understanding of the code. Our proposed analysis could be used as a decision support for teaching programming language features, or for software architects and developers to show what the current state and trends in this area are.

## II. Research objectives

### *A. .NET framework and C# language version distribution*

Firstly, we want to discover the distribution of .NET framework versions in the projects that were examined. Also, because C# language version is independent from the .NET framework, we will evaluate the distribution for this separately.

The goal of this objective is to create distribution statistics for the .NET and C# versions. We also want to try and establish if there is a trend between versions of .NET and versions of C# language.

### *B. Portable Class Library profile distribution*

Then we will focus on the PCL (Portable Class Library) support for the examined projects. PCL is a subset of the .NET framework which allows one to create a .NET library (assembly) which can be used on different platforms. Initially, PCL supported the Windows, Silverlight, Windows phone and Xbox platforms. Later, Microsoft - together with Xamarin, introduced support for the iOS and Android platforms. By choosing the platforms supported, one can define the PCL profile. Currently, there are around 40 profiles available.

The goal is to measure the share of projects which support PCL. In the event that a project supports PCL to find out which profiles it supports so that we can identify the most used profiles within all examined libraries.

### *C. Adoption rate of .NET framework and C# language features*

After that, we will measure the adoption rate. What we mean by adoption rate is a time analysis of the particular project in which we want to identify how long it takes to migrate to a new version of .NET framework and to use the new C# language features. For projects in which there was no migration to a new .NET framework - or no use of any new language features, we want to identify the reason(s) why.

The goal of this objective is to find out how long it takes a developer to migrate to a new .NET framework and how long it takes to use any new C# language features. We also want to identify variables which affect this adoption rate (e.g. the size of the project, the age of the project, supported OS …)

### *D. Usage share of new .NET framework and C# language features*

The last metric which we want to study is the usage share of the .NET and C# features. What we mean by usage share is the usage of a newly-introduced class or method or newly introduced C# language features to accomplish an expected goal instead of using an older alternative approach. A typical representative is the C# Auto-Implemented property feature. Before C# 3.0, you had to define a property with an implemented body and use a backfield property. Since C# 3.0, one has been able to use this

This work was supported by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2015/062.

feature to reduce the amount of “boiler-plate” code in order to have a cleaner code.

The goal of this objective is to define the requisite code to attain the required functionality before and after the feature was introduced and determine and identify such occurrences across the entire examined project and then create statistics to describe which .NET/C# features are the most popular and which are the least popular.

## III. Methodology

To be able to undertake these analyses, we plan to use our own tool - which is in the form of a C# application in which each analysis will be a stand-alone plug-in. This tool will automatically obtain the input data and run the analyses.

As input data for our research, we plan to use C# projects hosted on GitHub. GitHub is a Web-based Git repository-hosting service. It offers all of the distributed revision control and the Git Source Code Management (SCM) functionality as well as adding its own features. As of 2015, GitHub reports having over 11.3 million users and over 28.1 million repositories [4], which makes it the largest host of source code in the world [5].

GitHub also provides API for programmatically acquiring the necessary information. We plan to use this API to get a list of popular projects and download those projects.

Then we must prepare an environment to be able to build downloaded projects. Each project needs to have properly set up dependencies. This is mostly done with the Nuget package system [6]. When the environment is prepared, we can then run our analyses; and plan to use Roslyn for code analysis.

## IV. Current progress

Currently, we are able to use the GitHub API to obtain a sorted list of projects and are also able to download them. In addition, we have prepared a high-level structure of our analysis tool. We have a few beta versions of plug-ins (we plan to use one plug-in per feature) which will analyze code to search for C# language features

## V. Expected problem areas

There is a standard technology called MsBuild [7] for building .NET projects. Developers can also use the Nuget packaging system to manage external dependencies. However, in some cases, the projects are too complex to be handled by MSBuild and must be built using a different technology. We will try to adapt these other technologies to build projects to run our code analyses.

The next problem is to define code snippets to measure the acceptance of new features. For some features, it is easy to implement such code snippets (e.g. “Property with Backfield” vs. the “Auto-Property” feature); however, for other features it is very difficult - or even impossible, to cover all snippets (like a code for searching for the maximum vs. the LINQ IEnumerable.Max function).

## VI. Expected contribution

The main contributions of our research will be:

- To create a graph of .NET framework version as well as C# language version distributions to identify the most, and least, popular .NET framework and C# language versions.
- To create a histogram of platforms supported by PCL and to identify just how much developers use PCL and identify widely-used PCL profiles a s well as lesser-used profiles.
- To create and compile complex statistics on the adoption rate of the .NET framework and C# language features. Then, to try to identify the factors (like project size, project age, etc …) that affect the adoption rate of new .NET framework and C# language versions.
- To create a set of complex statistics on the usage share of new .NET framework and C# language features; as well as to identify the most and least usable features.

## VII. Conclusion

In this work, we have described future research focused on investigating .NET framework and C# language feature usage. We began by defining the research objectives that we want to focus on. Then, we went on to describe how we want to obtain the input data for our research, as well as the methodology with which we want to collect such information.

## Reference

[1] R. Dyer, H. Rajan, H. A. Nguyen and T. N. Nguyen, "Mining Billions of AST Nodes to Study Actual and Potential Usage of Java Language Features," in *ICSE ’14*, Hyderabad, India, 2014.

[2] D. Kim, E. Murphy-Hill, C. Parnin, C. Bird and R. Garcia, "The Reaction of Open-Source Projects to New Language Features: An Empirical Study of C# Generics," in *Journal of Object Technology, vol. 12, no. 4*, 2012.

[3] R. Dyer, H. A. Nguyen, H. Rajan and T. N. Nguyen, "Boa: A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories," in *ICSE 2013*, San Francisco, USA, 2013.

[4] "GitHub," GitHub, Inc., 2015. [Online]. Available: https://github.com/about/press.

[5] G. Gousios, B. Vasilescu, A. Serebrenik and A. Zaidman, "Lean GHTorrent: GitHub Data on Demand," in *MSR'14*, Hyderabad, India, 2014.

[6] "NuGet," .NET Foundation, 2015. [Online]. Available: https://www.nuget.org/.

[7] "Microsoft.Build (MSBuild)," Microsoft, 2015. [Online]. Available: https://github.com/Microsoft/msbuild.