IMPROVING THE TEACHING PROCESS BY MOVING FROM .NET TO .NET CORE FRAMEWORK

Luka Lukić[1] Nenad Kojić[2] Milanko Kragović [3] Milena Vesić [4] Ksenija Lazić [5]

Abstract: .NET technologies are often the choice of higher education institutions as the platform on which they base their subjects. For the teaching to be successfully conducted, it is necessary that teachers, classrooms, laboratories and students are equipped with licenses and accompanying software. This requires the use of the Windows operating system, working with closed-source programs, and providing a license for each individual student. With the release of the .NET Core framework, Microsoft has developed a version of the .NET that is completely free, platform-independent and open-source. With this in mind, the aim of the paper is to explore the benefits of switching from the .NET to .NET Core framework in terms of teaching process.

Key words: .NET, framework, teaching process, Microsoft, open-source, technology

1. INTRODUCTION

.NET Framework, alongside with C# programming language, has been Microsoft's primary object-oriented technology for two decades [1]. In addition to its use in industry, the .NET framework has found its way into academic institutions, mostly where object-oriented technologies are studied. However, given its closed-source nature, and necessary licenses and tools [1], [2], academic institutions have to spend more resources in terms of cost and staff engagement than what would be the case with an open source technology. Furthermore, students, in order to be able to study and work on their projects, also need to be equipped with the same tools and licenses, affecting the institution's resource utilization even more.

In July 2016, Microsoft announced the completely free, multiplatform and open-source version of .NET Framework called .NET Core [3]. Being built on top of the same specification, it has been declared as a main platform future development will be focused on [3]. Many frameworks, such as Windows Forms or WPF, have been re-written to support the .NET Core [4]. Given its open source nature and the fact that community members can now contribute to source code, it is significantly easier to find useful information and study materials than was the case with the .NET Framework. Taking into account the changes that have come with the .NET Core framework, the goal of this paper is to explore the potential benefits of switching from the traditional .NET to .NET Core framework in educational institutions.

The second chapter discusses the .NET Framework's design principles and goals. The third chapter is dedicated to main .NET tools and libraries from the developer's standpoint, while the fourth chapter introduces the .NET Core as the platform-independent and open-source successor to the .NET Framework. Chapter five is devoted to a comparative analysis of the .NET Framework and the .NET Core. In the sixth chapter, a conclusion is drawn.

2. .NET DESIGN

[1] Teaching associate, Academy of technical and art applied studies Belgrade department School of Applied Studies for Information and Communication Technologies, Zdravka Čelara 16, Belgrade, luka.lukic@ict.edu.rs

[2] Professor of vocational studies, Academy of technical and art applied studies Belgrade department School of Applied Studies for Information and Communication Technologies, Zdravka Čelara 16, Belgrade, nenad.kojic@ict.edu.rs

[3] Lecturer, Academy of technical and art applied studies Belgrade department School of Applied Studies for Information and Communication Technologies, Zdravka Čelara 16, Belgrade, milanko.kragovic@ict.edu.rs

[4] Teaching associate, Academy of technical and art applied studies Belgrade department School of Applied Studies for Information and Communication Technologies, Zdravka Čelara 16, Belgrade, milena.vesic@ict.edu.rs

[5] Teaching associate, Academy of technical and art applied studies Belgrade department School of Applied Studies for Information and Communication Technologies, Zdravka Čelara 16, Belgrade, ksenija.lazic@ict.edu.rs

This section will discuss the .NET Framework, its infrastructure, history and available software development tools.

2.1. .NET Framework structure

Microsoft's goal with the .NET Framework was to create a programming platform that can support multiple programming languages interchangeably [1], [2]. By doing so, reusable source code can be shared more easily and software problems can be solved by the programming language that best suits the problem and still be reused by other languages. The first version of .NET Framework was published on February 13, 2002 [1].

At its core, each .NET Framework consists of two key components:

- Common Language Runtime – CLR
- Base Class Library – BCL

2.2. Common Language Runtime (CLR)

CLR is an implementation of the Virtual Execution System (VES), one of the elements of the Common Language Infrastructure (CLI) specification [1]. CLI is an open specification developed by Microsoft and standardized by ISO and Ecma that describes executable code and a runtime environment that allows for multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

In order to support multiple programming languages, CLI specification defines:

- Common Type System (CTS)
- Metadata
- Common Language Specification
- Virtual Execution System (VES)
- Common Intermediate Language (CIL)

As there is a large number of programming languages supported by .NET, the CLR is actually a virtual machine that executes the code written in any .NET compatible programming language. As the CLR manages the execution of the code, the code executable by the CLR is called managed code [1]. This management CLR performs consists of tasks such as memory management, security concerns, exception handling, garbage collection, thread management, etc.

2.3. Base Class Library

Base Class Library is the core library and provides the most foundational .NET types such as arrays, collections, exceptions, types for mathematical operations, I/O operations and so on. It also forms the basis for all other .NET class libraries [1]. It is being updated with each .NET release.

In addition to the library of basic classes, there is a far larger set of classes packed in a .NET framework that covers a wide range of functionalities such as working with a database, creating web or desktop user interfaces and others.

Currently, there are four distinct .NET frameworks, all owned by Microsoft:

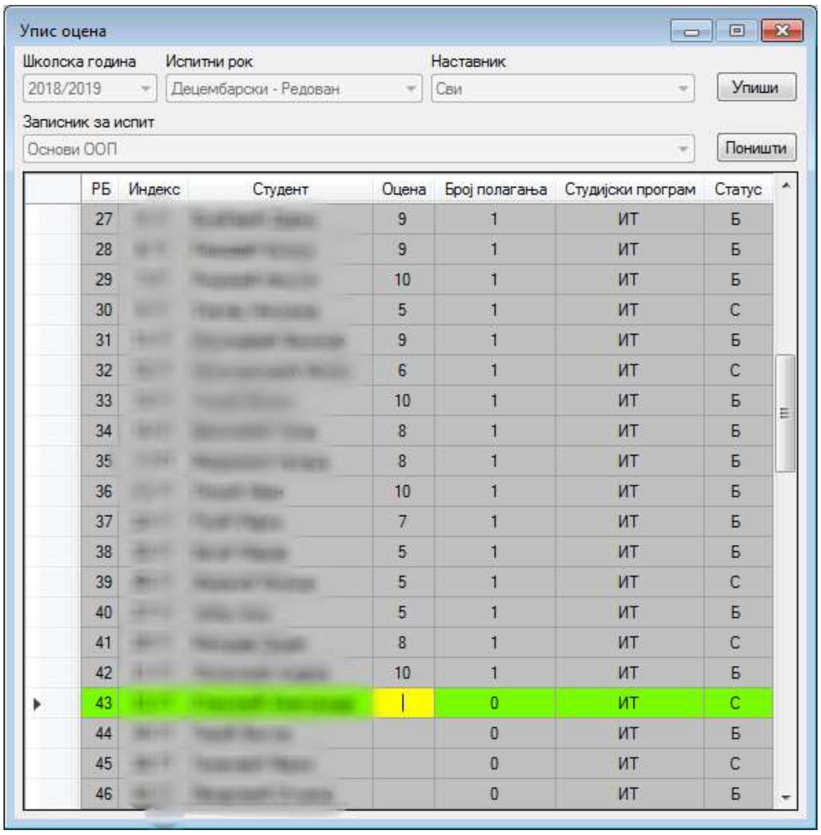
- .NET Framework
- .NET Core
- Mono
- UWP

3. .NET FRAMEWORK DEVELOPMENT TOOLS AND LIBRARIES

From the very beginning of .NET, Microsoft's goal has been to provide everything a developer needs to develop software, in one place [1], [2]. To that end, in addition to the Base Class Library, which is an integral part of the specification under which the .NET framework is developed, Microsoft has developed many other tools, such as Visual Studio as a single integrated development environment (IDE). The Framework Class Library has been expanded to support the development of desktop, mobile and web applications, each of which has a wide range of capabilities.

3.1. .NET Framewrk desktop development tools

For the purposes of desktop development, since the first version of the .NET Framework there has been a class library called Windows Forms [5]. All visual elements share the same base class, *Control*, so the UI elements are collectively called windows forms controls. Windows forms are developed over the already existing Windows API, and a large number of controls are implemented only as a wrapper around the already existing Windows components [5]. Figure 1 shows an example of a screen form implemented using Windows Forms. The example uses the standard controls that come with the frame: ComboBox, Button, DataGridView, Label, etc.


Упис оцена

Школска година: 2018/2019
Испитни рок: Децембарски - Редован
Наставник: Сви
Записник за испит: Основи ООП

	РБ	Индекс	Студент	Оцена	Број полагања	Студијски програм	Статус
	27	[redacted]	[redacted]	9	1	ИТ	Б
	28	[redacted]	[redacted]	9	1	ИТ	Б
	29	[redacted]	[redacted]	10	1	ИТ	Б
	30	[redacted]	[redacted]	5	1	ИТ	С
	31	[redacted]	[redacted]	9	1	ИТ	Б
	32	[redacted]	[redacted]	6	1	ИТ	С
	33	[redacted]	[redacted]	10	1	ИТ	Б
	34	[redacted]	[redacted]	8	1	ИТ	Б
	35	[redacted]	[redacted]	8	1	ИТ	Б
	36	[redacted]	[redacted]	10	1	ИТ	Б
	37	[redacted]	[redacted]	7	1	ИТ	Б
	38	[redacted]	[redacted]	5	1	ИТ	Б
	39	[redacted]	[redacted]	5	1	ИТ	С
	40	[redacted]	[redacted]	5	1	ИТ	Б
	41	[redacted]	[redacted]	8	1	ИТ	С
	42	[redacted]	[redacted]	10	1	ИТ	Б
▶	43	[redacted]	[redacted]	|	0	ИТ	С
	44	[redacted]	[redacted]		0	ИТ	Б
	45	[redacted]	[redacted]		0	ИТ	С
	46	[redacted]	[redacted]		0	ИТ	Б

Figure 1 – An example of Windows Forms based application

With .NET Framework’s third release, Microsoft has released the second desktop application development tool, WPF (Windows Presentation Foundation), along with a user interface language called XAML (*Extensible Application Markup Language*) [6]. XAML is a derivative of XML and is an extensible markup language that resembles HTML in structure and usage, giving the user more flexibility than previous Windows Forms. In 2018, Microsoft declared WPF an open source project under the MIT license. Since its first version, WPF has gained considerable popularity, and the Visual Studio IDE graphical interface itself is written mostly using WPF. Figure 2 shows an application whose user interface is implemented using WPF.

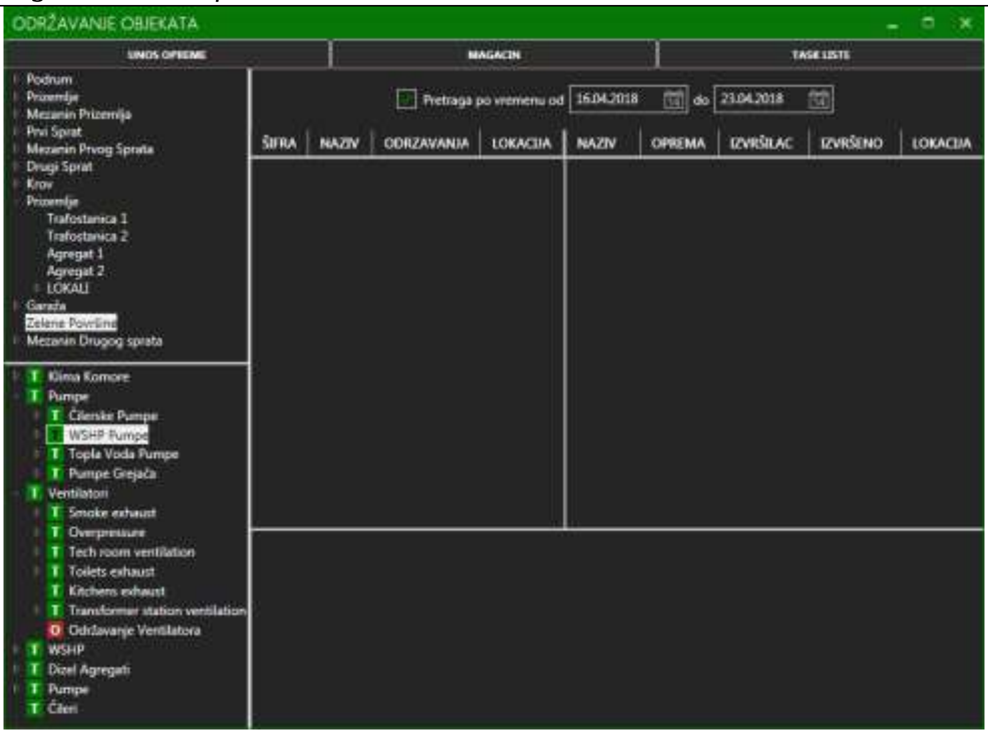


Figure 2 – An example of WPF based application

Use of Windows forms is still very common, which is evidenced by the fact that, although not previously planned, subsequent releases of the .NET Core included them at the initiative of the community [4]. Although .NET Core is supported by multiple platforms, Windows Forms can only work under the Windows operating system [4].

3.2. .NET Framework web development tools

In the early days of .NET, at the beginning of the new millennium, desktop applications were the industry standard. It was a time before browser applications shaped the software development industry [7]. Back then, web applications were scarce and mostly made up of textual content, the so-called web presentations. However, what distinguished them from desktop applications was the availability. There was no need for complex installations and they could work on any device that had a web browser. Due to their availability, the complexity and prevalence of web applications are constantly increasing.

Since their first version, the authors have built support for web application development into .NET as an integral part of the framework class library [8]. The web application development tool was called ASP.NET Web Forms, in line with the desktop application development tool [8]. Developers could quickly, using already defined, desktop-like controls, implement websites [8]. However, the closed nature of Web Forms, which was the basis of rapid development, has increasingly been the target of criticism. Namely, the development worked well as long as there was no need for additional adjustment of the existing UI elements [8]. At the same time, the frameworks of other programming languages gave developers complete control over client code, and client languages became more prevalent. Alongside with Web Forms, Microsoft has developed another web oritented framework called Windows Communication Foundation (WCF), designed using service-oriented architecture principles to support distributed computing [9].

Web Forms were Microsoft’s primary tool for web application development until 2009, when the first version of the ASP.NET MVC framework was released [8]. It was released as a solution to the problem of closed Web Forms, designed using standard software principles for web application development at the time [8].

MVC unlike Web Forms, was designed to be fully compatible with all elements of the web, in stark contrast to Web Forms. This way, the developers had complete control over the management of HTTP requests, the appearance and definition of web elements, ie. all the elements that were missing, or were criticism to the previous framework.

The ASP.NET MVC had 5 major releases, while in the meantime another framework, the ASP.NET Web API, was published [10]. The Web API shared the source code with the MVC framework, had the same directory structure, but, unlike the MVC, exchanged data with the client exclusively in the form of REST (Representational State Transfer), via XML or JSON (*JavaScript ObjectNotation*). Both projects continued their further development as part of the .NET Core project [3], [4].

3.3. .NET Framework requirements

.NET Framework is dependent on the Windows operating system and comes alongside with an operating system [1][2][4]. Furthermore, parts of the operating systems are built using the framework [1]. Each Windows installation comes with the preinstalled version of the latest .NET Framework version at the time. Consequently, .NET Framework applications can be run only on Windows [1].

In terms of hardware requirements, .NET Framework requires a minimum of:

- Processor: 1 GHz
- RAM: 512 MB
- Disk space: 4.5 GB

When it comes to the development process, as mentioned earlier, there is the Visual Studio IDE, supporting a wide range of development tasks and application types, from desktop and web to mobile and game development. Since 2014, the Community version of the Visual Studio has been available as a free version of the IDE to be used by individuals, academic research, classroom learning environments and small teams. Being the only free tool that seamlessly integrates with the .NET development tools, its system requirements should also be taken into account (Visual Studio 2019):

- Operating system: Windows 7 SP1 or newer
- Processor: 1.8 GHz (Quad-core or better recommended)
- RAM: 2 GB (8 GB recommended)
- Disk space: 20-50 GB (Typical installation, SSD recommended)
- Video card that supports a minimum display resolution of 720p
- .NET Framework 4.5.2 or above to install Visual Studio, .NET Framework 4.7.2 to be able to run it

For the teaching to be conducted successfully and in a timely manner, it is essential that laboratories, as well as students are equipped with optimal computers and licenses. To do so, educational institutions where .NET courses are taught should follow the before mentioned hardware and software requirements and their recommended values.

4. .NET CORE

The .NET Core framework is the open source, cross-platform successor to the .NET framework [3], [4]. Code written in the .NET Core framework can be executed on Linux, MacOS and Windows operating systems, and all source code is available on the GitHub platform, published under the MIT license [3], [4]. The first version of the framework was presented on November 12, 2014, and the first version released on July 27, 2016, along with an update for Visual Studio 2015 including the support for .NET Core. In terms of programming languages, there is a native support for C#, Visual Basic and F# [3], [4].

Ever since the release of the .NET MVC framework, Microsoft has increasingly taken the initiative to develop open source software. jQuery was included in Visual Studio 2010 as the first open source library not designed by Microsoft [8]. Today, the focus of Microsoft Corporation from the aspect of software development is on open source projects [3], [4], and the current version at the time of writing is .NET Core 5.0.

As .NET Core represents a completely new version of the framework, it is not compatible with libraries written for the traditional .NET Framework. About the same time, Microsoft acquired Xamarin, a company that owns the Mono Framework , another .NET implementation. By doing so, Microsoft owned three different versions of the .NET framework: the *.NET Framework*, the *.NET Core*, and the *Mono*.

Although all three versions were written according to a common specification [1], each of them had its own versions of the BCL, as required by the CLI. This meant the existence of three different code bases that served the same functionality because the frameworks, in terms of their core, were comparable. As a solution to the compatibility problem, the .NET Standard framework was developed, defining the base class libraries whose code would be available to each of the frameworks. In that way, class libraries that were ment to be shared across frameworks should target the .NET Standard instead of specific framework [4].

4.1. .NET Core desktop development tools

The first two versions of .NET Core framework are mostly web oritented. Starting with .NET Core 3.0, .NET Core supports Windows Forms and WPF [4]. Both frameworks are designed for Windows only, so they are not cross-platform. WPF is a rich layer over DirectX and Windows Forms a thin layer over GDI+ [4]. Bearing in mind that the .NET Core framework is where all future development efforts are going to occur, now there is ability to port the existing desktop applications from .NET Framework to .NET Core and therefore proceed with further development using .NET Core framework.

4.2. .NET Core web development tools

ASP.NET Core is the successor to the ASP.NET Framework. There are three types of web applications .NET Core offers, each of which is fully cross-platform [3]:

- ASP.NET Core MVC
- ASP.NET Core Web API
- ASP.NET Core Razor Pages

ASP.NET Core MVC and Web API are considered a continuation of the previous .NET MVC and Web API frameworks [3]. Programmers familiar with the previous versions should be able to program with .NET Core versions with a little training. Unlike desktop frameworks, ASP.NET Web Forms and WCF are not ported to the .NET Core ecosystem. Currently, there is an open-source community project as an attempt to enable moving existing WCF projects to .NET Core framework. The project has not been ready for production yet.

4.3. .NET Core requirements

.NET Core is cross-platform and currently can be run under Microsoft, Linux and MacOS operating system. There are no operating system dependencies other than appropriate SDK (Software Development Kit) to be able to start developing .NET Core applications. In terms of minimal hardware needed to run .NET Core application, minimal operating system requirements are enough for the application to run [3], [4].

Visual Studio IDE supports .NET Core development as well, but its full version is only available on Windows operating system. There is a version of Visual Studio designed for MacOS. In terms of Linux operating system, Visual Studio Code editor is recommended.

Given .NET Core applications can be more easily developed using code editors or IDEs other than Visual Studio, and since the Visual Studio Code is the recommended one, we will investigate its hardware requirements as baseline requirements to develop .NET Core applications [11]:

- Disk space: 200 MB (Plus the application size)
- Processor: 1.6 GHz
- Ram: 1 GB

Given that Visual Studio Code can run under all three mentioned operating systems, there are operating system level requirements:

- Windows: .NET 4.5.2
- Linux: GLIBCXX 3.4.15 or later, GLIBC 2.15 or later

5. COMPARISON OF THE .NET FRAMEWORK AND .NET CORE FOR THE PURPOSES OF TEACHING PROCESS

For the teaching process to be successfully conducted, academic institutions, professors and students need to be equipped with the appropriate hardware and software. In terms of .NET based subjects, that means relatively powerful computers with acompanying software and licenses, until .NET Core framework was released. This chapter will discuss pros and cons of moving from .NET Framework to .NET Core framework.

5.1. Potential benefits of using the .NET Core

Unlike .NET Framework, .NET Core is platform-independent and open-source [1], [3], [4]. This platform independence is reflected in its ability to be run on Windows, Linux and MacOS operating systems. Furthermore, as discussed in the previous chapter, .NET Core requires far less in terms of hardware to be able to run and perform. Considering that, there are several potential benefits to using .NET Core. Namely, being open-source .NET Core is completely free and could therefore reduce cost for both the institution and its students. This cost reduction is further influenced by the fact that hardware requirements are lower than those of .NET Framework.

Also, bearing in mind its platform-independence, the students would not have to match institution’s setup in terms of operating system and software in order to perform student assignments and projects.

Being open-source, .NET Core allows both students and their teachers to inspect the source code and allows for a more comprehensive understanding of the underlaying principles upon which the framework is built. This also allows academic researchers to be actively involved in the further development of the framework.

5.2. Potential disadvantages of using .NET Core

By being less mature than .NET Framework, there are relatively frequent changes to the framework itself. Those changes can have negative effects on the teaching process due to the frequent need to change and adjust subject materials and examples.

If the subject’s primary goal is desktop development, it might be better to use the .NET Framework due to the fact that not all available development tools are still ported to the .NET Core and Visual Studio, and can thus slow down the teaching and development process.

And last, but not least, it is important to emphasize that the .NET Framework is still supported and there is no announced date when this support will end. Also, there are a lot of companies that have projects implemented using .NET Framework that are looking for new developers.

6. CONCLUSION

As discussed in previous section, moving from .NET Framework to .NET Core can potentially improve the teaching process in educational institutions by several factors. Those include cost efficiency, more choices in terms of hardware and software for both the institutions and students, more resources to learn from, and being open-source, the ability to be involved with the further framework development. On the other hand, .NET Core is still considered a new framework, with relatively frequent changes, and has not yet fully adopted previous framework's desktop development aspects. Furthermore, .NET Framework is still supported and many companies are still using it. While there are benefits of using .NET Core instead of .NET Framework, it would probably be optimal to keep using both frameworks where they best fit the subjects, while leaning towards .NET Core as it progresses further development.

7. REFERENCES

[1] Richter, J. (2006). *CLR via c#* (Vol. 4). Redmond: Microsoft Press.

[2] Skeet, J., &Simeloff, E. (2014). *C# in Depth*. Manning.

[3] Freeman, A. (2016). *Pro Asp. net Core Mvc*. Apress.

[4] Troelsen, A., &Japikse, P. (2017). *Pro C# 8: With .NET Core 3:Foundational Principles and Practices*. Apress.

[5] Sells, C., &Weinhardt, M. (2006). *Windows Forms 2.0 Programming (Microsoft Net Development Series)*. Addison-Wesley Professional.

[6] Nathan, A. (2006). *Windows presentation foundation unleashed*. Pearson Education.

[7] Fowler, M. (2018). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

[8] Freeman, A. (2014). Pro AsP. Net MVC 5 platform. In *Pro ASP. NET MVC 5 Platform* (pp. 3-8). Apress, Berkeley, CA.

[9] Klein, S. (2007). *Professional WCF Programming:. NET Development with the Windows® Communication Foundation*. John Wiley & Sons.

[10] Kurtz, J., & Wortman, B. (2014). *ASP. NET Web API 2: Building a REST Service from Start to Finish*. Apress.

[11] Johnson, B. (2019). *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*. Wiley.