Improving the Quality and Security of .NET-based Applications

Theses book

Pócza Krisztián

Supervisor:
Dr. Porkoláb Zoltán

Eötvös Loránd University
PhD School of Informatics
Program of the basics and methodology of informatics
Head of school and program: Dr. Demetrovics János

Budapest, 2010

1 Introduction

Since the first issue of the standard Microsoft .NET technology in 2002, it has gone through some dynamic development. As we are speaking about a mature and well designed framework, it is essential to mention that the main concepts of the first version are still alive - which is characterized via its being continually and dynamically extended as a framework and in language support.

The .NET framework itself helps to create programs with higher quality and security parameters. These tools are, for example, the handling of managed references, protection against buffer overflow errors, and runtime type checking. However, there are areas that are not covered by this system, either as a framework or a base system. It has to be considered, too, that handling and maintaining the security of precise applications (e.g. distributed systems) is the task of the designer and developer of the application. This is a sub-area that scientific research work and opportunities must pay attention to.

I shall analyze the exact sub-area of the widely-defined security and quality that concentrates on the phases of the software development process; and I will note and group together the methodologies that make programs more secure and of higher quality.

Access control related to a distributed system's public services is a widely studied area. One of the early results from such work can be tied to the sophisticated access control mechanism of the Eiffel language. Yet there is no known generic method that would extend the access controls mechanism of the Eiffel language to distributed systems. A further problem of the workflows hosted by distributed systems is that these are not connected with the service interfaces that provide the public facade of workflows. Using declarative programming, even a method level role-based rule-hierarchy user access control system, or a network-segment restricting mechanism might be developed. A further general problem with current frameworks is that restriction of permissions does not have the correct granularity level, i.e. they are not implemented at the method level. To handle these problems generally a formal model has to be constructed that ensures platform independency. This is the topic of the first thesis.

Based on the formal model, implementation can be created. To make it easier to create the implementation, it has to be considered while constructing the formal model. The aim is the creation of a bijective rewriting system that results in a platform dependent

way of operating or solution for any platform. The .NET framework is a current platform onto which a rewriting system can be constructed. This is the topic of the second thesis.

There is no logging mechanism for the .NET framework that would be able to create instruction or even variable reading and a writing level runtime trace. A logging solution like this would be usable as input for dynamic program slicing or in the developing and testing phases. The dynamic slicing algorithms require a log that contains variable reads and writes alongside the executed instructions. Another important use case is when we examine the log after execution of the program. The log can serve up information that would be hard or impossible to retrieve while debugging. This solution would be interesting, too, in the case of multi-threaded applications, giving more usable results. This is the topic of the third theses.

2 Goals and methods

In my research work I have created methods that can improve the quality and security of both standalone and distributed applications. The methods were implemented based on the .NET framework, and this implementation – with some minor changes, though retaining the general conceptions – can be adapted to other platforms.

My results have been achieved using iterative methods all the time. This means that I examine the area being researched continually diving deeper into the topic, so that I am able to make more and more contributions.

In the first iteration the aim was always to become acquainted with the research area as much as possible, to be able to find the field where effective research can be done. This will always mean a sub-area that others have not covered as yet; or perhaps more researchers are working on this topic but their approach or methods of research are different from mine.

In the subsequent iterations I have been continually publishing my results; and the dissertation will make a summary of such results often following the iterative approach.

3 Results

3.1 Security and the quality of distributed applications

I have created a method that advances the work of other researchers, while it additionally systematizes the usage of already existing but independently handled security restrictions.

According to a well-known practice, an interface is created using the facade design pattern which publishes the services of a system to the outer world.

These services define no restrictions at a contract level- which poses a problem that one needs to solve. Thus, a method has been created that can be used to add restrictions to the contracts used during the communication between the client and server.

The method combines the dynamic user access control rules of services and workflows, extends the runtime access control to distributed applications, and extends the functionality of user-level access policies and network policies. I have joined the independent, non-integrated business services and workflows and rule-based access control; also I have defined an extension to the classic access control in the context of distributed applications.

The resultant solution is a formally-defined, and platform independent restriction and validation system, that can solve industrial problems.

The results for this topic have been published in [1] [2] [3].

Thesis 1. I have shown the limits of the currently used access control mechanisms in the case of distributed applications. A formal model was defined that answers the most important questions of distributed access control; and it connects the permission management of workflows and services, which extending the role-based access control and the network level security mechanism. Additionally, it guarantees platform- and implementation-independency. The applicability of the formal model has been validated through industrial case studies.

3.2 Implementation of the formal model

The formal model shown would be unusable if it could not help the practical work of programmers. So in the next iteration a precise implementation of the formal model was demonstrated. First, I introduce the goals and enumerate the programming paradigms whose principles can help transform the formal model into a precise programming system.

The transformation is a projection - a rewriting system - between the formal world and the friend of it, the C# attributes suggesting a declarative paradigm.

The case studies that were formalized have been transformed into a C#-language

implementation, with which the operability of the method can be seen. I have designed the architecture of the system and assigned system components for every building block. In the course of implementation more .NET technologies were employed, the most important being the WCF (Windows Communication Foundation) for communication and the WF (Workflow Foundation) for the workflows.

A previous version of the shown way of proceeding is used in applications created for government and enterprise customers.

The results of the topic are published in [3].

Thesis 2. Using the formal model I have designed a framework, and based on the design, have created a working implementation in the standard C# language for the .NET platform. A rewriting system was shown, which can be used to transform the restrictions defined by the formal model into the C# language. On the bases of this framework I have implemented formally-defined case studies.

3.3 Runtime trace

I have examined the tools that can be used when developing .NET-based applications. Via this analysis it becomes clear that there is no detailed or cross-language runtime tracing method that might be used as input for dynamic program slicing. So I set myself the goal of identifying the requirements of a runtime tracing method that will be usable even in the cases of dynamic program slicing.

While designing and implementing the solution an iterative method was used. First of all, I created a trace at the level of instructions bounded by sequence points. After the way of operating was fulfilled, I moved to the creation of a variable level trace. Here, on can see which variables are read and written during program execution. The method is non-intrusive, meaning it does not need manual or automatic modification of the original source code to generate a detailed runtime trace of executing applications. The trace-writer methods are inserted into the IL code of the application being trace while the time of the JIT compilation.

The solution has been tested via applications having different running characteristics.

The results from this topic have been published in [4] [5] [6].

Thesis 3. I have shown the necessity of a detailed runtime tracing solution in the standard .NET platform. I have outlined requirements and created a cross-language implementation that does not require alteration of the original source code. The solution can work in a parallel environment, too, with adequate quality and performance.

4 Further work

When a result comes to light, it is important that one can easily integrate it with existing way of working, that it is easy to advance it, and that one is able to connect with further solutions and procedures. There is no research result that would be perfect first time round - every result goes through a process of evolution and development while it becomes more generalized or more specific.

I intend to continue the two most important topic of my dissertation which are also those appearing the most details: these are improving the quality and security of distributed .NET applications, and additionally, the creation of a runtime trace for higher quality .NET-programs.

What can be seen in the solution presented in the first and second theses is that two services are entirely independent of one another, so there is no possibility of creating a restriction - when calling a method - related to the first service based on the state of the second service. When it one needs to define a common package of restrictions for more services, their context- and session-dependent states have to be handled in common storage.

I intend to make use of the results achieved within the context of distributed applications while designing and implementing industrial applications. Industrial scenarios are able to show more specific research directions.

My tracing solution based on its detailed output is able to support detailed dynamic slicing algorithms working with .NET applications that have complex expressions at a source code level.

When we add preconditions, post-conditions, and invariants to our programs, we define a specification; and then based on the specification the correctness of the program may be verified. Using the runtime trace - after full execution of the program - the program path and the values of the variables can be post-validated.

The constantly developing .NET framework leads to the continual addition of ever

newer services to my tracing mechanism. Alongside the handling of more modern framework-level functionality, it is also important to improve the performance of the trace.

With the help of the philosophy of modern design paradigms (e.g. Domain Driven Design) mentioned in this work and with the integration of the now fashionable generative programming, I intend to take a step in the direction of methods that support the creation of high quality, secure and loosely coupled applications.

The goal is to support the use of paradigms going alongside the object-oriented and component-oriented paradigms in the case of distributed applications which, in a certain context, with the integration of object-oriented and component-oriented paradigms can help in the design and implementation of distributed applications.

These are the declarative, the aspect-oriented, the functional, and the generative programming paradigms.

References

1. Biczó, M., Pócza, K., Porkoláb, Z.: Runtime access control in C# 3.0 using extension methods. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica 30, 41-60 (2009)

2. Pócza, K., Biczó, M., Porkoláb, Z.: Runtime Access Control in C#. In : Proceedings of the 7th International Conference on Applied Informatics (ICAI), Eger (Hungary), pp.28-31 (2007)

3. Pócza, K., Biczó, M., Porkoláb, Z.: Securing Distributed.NET Applications Using Advanced Runtime Access Control. Frentiu et al ed.: Studia Universitatis Babes-Bolyai Informatica LIII(2008/2), 39-54 (2008)

4. Pócza, K., Biczó, M., Porkoláb, Z.: Cross-language Program Slicing in the.NET Framework. In : Conference proceedings of.NET Technologies 2005, Plzen (Czech Republic), pp.141-150 (2005)

5. Pócza, K., Biczó, M., Porkoláb, Z.: Towards Effective Runtime Trace Generation Techniques in the.NET Framework. In : Short communication papers proceedings of.NET Technologies 2006, Plzen (Czech Republic), pp.9-16 (2006)

6. Pócza, K., Biczó, M., Porkoláb, Z.: Towards detailed trace generation using the profiler in the.NET Framework. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica 30, 21-40 (2009)

7. Biczó, M., Pócza, K., Forgács, I., Porkoláb, Z.: A New Concept of Effective Regression Test Generation in a C++ Specific Environment. Acta Cybernetica 18(3), 481-501 (2008)

8. Biczó, M., Pócza, K., Porkoláb, Z.: A Cache-Based Interprocedural Static Slicing Algorithm. In : Proceedings of the 7th International Conference on Applied Informatics (ICAI), Eger (Hungary), pp.207-218 (2007)

9. Biczó, M., Pócza, K.: Generating Functional Implementations of Finite State Automata in C# 3.0. Electronic Notes in Theoretical Computer Science (ENTCS) 238(2), 3-12 (2009)

10. Pócza, K., Pataki, N.: An Improvement on the Access Control Features of C#. In : Proceedings of Sixteenth Electrotechnical and Computer Science Conference (ERK 2007), Portoroz, vol. B, pp.33-41 (2007)

11. Pócza, K., Biczó, M., Porkoláb, Z.: docx2tex: Word 2007 to TeX. TUGBoat, TUG 2008 Conference Proceedings 29(3), 392-400 (2008)

12. Pócza, K., Biczó, M., Porkoláb, Z.: FC#: Designing an Internal Functional DSL to C# 3.0. In : Proceedings of the Implementation and Application of Functional Languages 20th International Symposium, IFL 2008, Hatfield, Hertfordshire (UK), vol. Technical Report no. 474 (2008), pp.299-314 (2008)