APPLICATIONS OF CODE ANALYSIS IN INFORMATION SYSTEMS USING TYPE REFLECTION

Nikola Valchanov[1]

[1] Faculty of Mathematics and Informatics, University of Plovdiv “Paisii Hilendarski”, 236 Bulgaria Blvd., 4003 Plovdiv, Bulgaria
Email: nvalchanov@uni-plovdiv.bg

Abstract. This paper discusses the opportunities that type reflection introduces in modern object-oriented programming languages. It outlines how modern information systems can benefit from code analysis done by tools based on type reflection. The paper presents an application of code analysis through type reflection that implements indexation of application areas that are subject to user access restriction

Key Words: *design patterns, information systems, software architecture, code analysis, type reflection, MVC.*

Introduction

Automated code analysis is not something new in modern software engineering. By nature, software programs are a list of instructions to a processor (hardware or software). These instructions are stored in a format and can be read, analyzed, and modified if needed. Due to the broad theoretical background of the term “reflection” [17] in computer science we will try and stay with a simpler definition that will serve the purposes of the paper:

Type reflection in object-oriented languages is the ability of a program to analyze its components and use the meta-data from the analysis to manipulate instances.

As mentioned above reflection is not new to programming languages. It is applicable both in low- and high-level programming language and has been around since the 1980s.

Type reflection on the other hand applies only to object-oriented languages and has recently gained popularity through modern programming languages – PHP, Perl, Python, C#, Java, Ruby etc.

The basics of type reflection

Though different programming languages implement type reflection at different extent, there is a basic set of features that each implementation provides. Those are indexation of members of a type, indexation of member of an instance of unknown type, instantiating objects of a dynamically determined type, accessing and modifying property values of objects of unknown type by property name, invocation of methods of an instance of unknown type by method name, extracting a list of classes declared within a given scope.

This set of features gives a certain flexibility to the programming language. Type reflection can be used to evade the strictness of the object-oriented paradigm. This opens the possibility for both introducing bad coding practices that make the code unsupportable and implementing complex design patterns [10] that increase system flexibility [12].

To get a better understanding of type reflection we will consider two popular programming languages – PHP [13] and C# [8].

Table 1. Type reflection capabilities in PHP and C#

	PHP	**C#**
Class descriptor	✓	✓
Method descriptor	✓	✓
Create instance from descriptor	✓	✓
Obtain a list of descriptors for declared classes	✓	✓
Ability to call a method using a descriptor or equivalent feature	✓	✓
Ability to extract a value using a descriptor or equivalent feature	✓	✓

From Table 1 it is evident that both languages, regardless if they are static typed or not, have capabilities to reflect types and manipulate instances either through type descriptors or equivalent language feature.

Type reflection and code analysis

Code analysis is a powerful multi-purpose technique that is vastly used in modern programming [1-4], [6], [7], [11-16], [18]. Contemporary code analysis is usually associated with tools like FxCop and StyleCop that analyze the code and suggest better approaches, coding styles and enforce programming and naming conventions based on a set of rules that synthesize best practices. Other tools that benefit from code analysis are refactoring and development assisting tools like ReSharper. There are tools like SolidOpt that introduce code optimizations based on analysis.

When considering code analysis outside of the static code analysis scope [1-4], [6], [7], we can identify some applications in feature development. The introduction of code analysis tools in information systems can simplify support, improve flexibility, scalability, and code reuse [1-4], [6], [7], [11-16], [18].

Modern information systems are usually built on object-oriented enterprise frameworks. These frameworks introduce structure and conventions that simplify the analysis of the system structures. If we consider the benefits from code analysis, we can see that they are not limited to source code analysis and code quality metrics extraction [3-6], [14]. If the focus is to be shifted towards the system itself, the analysis can gather meta-data about the inner structure and processes thus contributing not to the performance and maintainability of the code itself but to the flexibility of the system as a whole [12].

Let us review the structure of a modern application. Programming languages usually use an object-oriented library of tools for rapid application development that facilitate and speed up the development process. The programming language and the set of libraries are the building blocks of software applications. To take an application to enterprise level one applies proper software architecture that usually involves adequate design patterns [10] selected for the scope of the system. General purpose code analysis comes in at this level of application development. It provides suggestions for source code optimizations. It identifies common design flaws in the application architecture and misuse of design patterns and suggests better coding practices. This type of analysis focuses on the code itself, specifically on structure and code quality.

Code analysis can be taken a step further by analyzing the application block as a whole (either compiled assembly or loaded module) instead of the code reuse [1-4], [6], [7], [11], [12]. The code analysis tool does not have any knowledge of the structure or the purpose of the application. The tools that provide such services are generic and are applicable to all systems written in their target language.

To scale the analysis, the analyzer process needs knowledge of the system structure. These special purpose code analyzers are designed for specific tasks and extract meta-data from a specific targeted domain of the application. They rely on the structure and conventions of the enterprise framework that the application is built on and use the obtained meta-data information to provide framework specific metric data, identify deviations from conventions, generate documentation, simplify system support or act as a data feed for system features.

Type reflection is a great tool for implementation of such analyzers. Types are the building blocks of object-oriented software. They have a hierarchical structure, can be marked by attributes and are indexable using type reflection. Such code analyzers can use the types they index, instantiate objects and use them based on common interface contracts. This way the analyzer is no longer a static code analysis tool, but a proactive instrument with behavior on its own [1].

Application of code analysis with type reflection

To illustrate better the concept of application-wide code analysis with type reflection we will focus on two implementations.

Analyzer as data feed

Let us first focus on a tool that limits user access per controller and action in an ASP.NET MVC application [9].

As suggested above the analyzer needs to have knowledge of the system or the framework. In our case, we are using the ASP.NET MVC framework. The assembly of the MVC application has controller classes that inherit from the Controller class of the MVC framework. All public methods of a controller that return ActionResult are treated as actions.

To implement that we first build a tool that analyses the classes in the ASP.NET MVC application. For each class that inherits from Controller we place a record in our database and analyze its public methods that return an ActionResult. For each action, we add an additional record in our database. This way when we create new controllers, and our application grows the analyzer tool will re-index the code and fill in the new controllers/actions.

The right management part can be visualized using endpoints that list all users and enable each user to be associated with a controller or action. This way access rights to controllers and actions can be given dynamically per user.

It is obvious that the code analyzer here acts as a data feed that always gets the structure of the system. It extracts all resources of the web application that are reachable through the web server and provides them to the system itself for further use.

Pro-active analyzers

For illustrating pro-active analyzers we will discuss a tool that provides remote access to functional objects.

The problem that this tool targets is the lack of a lightweight mechanism that can provide distributed access to the classes from a given class library. This is needed in the cases where we do not want to use off the shelf application bus for exposing distributed objects but at the same time, we need to implement this distributed model.

The approach here is to build pro-active analyzers that index libraries of classes. The functionality of these analyzers is limited to obtaining a request through a socket, locating the requested type, instantiating it, locating the requested method, invoking it with the supplied arguments and returning the result over the socket.

When designing frameworks that use such architecture the logical approach is to build a tool that allows the distributed execution of the different layers of the application. To be native to the system these tools need to have knowledge of the architecture and structure of the application. Using this knowledge, the code analyzer indexes the types within the class library that implements the specific system layer, instantiates the requested tool, executes the requested functionality and returns appropriate response.

The direct benefits from pro-active code analyzers here are evident. They can implement complex functionality that is powered by the meta-data gathered by the analysis process. This technique opens vast horizons to implement pluggable models in modern software applications.

Conclusion

Code analysis is a widely used concept in the modern software development process. This technique can be applied in many ways [6], [12], [16]. Depending on the scale it can work on both code and application level.

Compared to static code analyzers, application-level analyzers focus on the system as a whole. This additional abstraction layer allows the analyzer to work with the system architecture. As a result, the information that is extracted and managed can be used for introduction of system optimizations or for implementation of functional tools that either support or are integrated into the application.

References

[1] F. Affonso, E. Nakagawa, A Reference Architecture Based on Reflection for Self-Adaptive Software, *IEEE*, 2013, 129–138, doi:10.1109/SBCARS. 2013.24, ISBN: 978-1-4799-2531-5.

[2] A. Ahmada, M. Babar, Software architectures for robotic systems: A systematic mapping study, *Journal of Systems and Software*, Vol. 122, 2016, 16–39, ISSN: 0164-1212.

[3] S. Alama, Z. Qub, R. Rileyc, Y. Chenb, V. Rastogid, DroidNative: Automating and optimizing detection of Android native code malware, variants, *Computers & Security*, Vol. 65, 2017, 230–246, ISSN: 0167-4048.

[4] A. Prajapati, J. Chhabra, Improving modular structure of software system using structural and lexical dependency, *Information and Software Technology*, Vol. 82, 2017, 96–120, ISSN: 0950-5849.

[5] M. Babar, A. Brown, K. Koskimies, I. Mistrik, *Agile Software Architecture, Aligning Agile Processes and Software Architectures*, Morgan Kaufmann, 2014, ISBN: 978-0-12-407772-0.

[6] T. Bao, S. Liu, Quality evaluation and analysis for domain software: Application to management information system of power plant, *Information and Software Technology*, Vol. 78, 2016, 53–65, ISSN: 0950-5849.

[7] J. Buckleya, N. Alib, M. Englisha, J. Rosika, S. Herolda, Real-Time Reflexion Modelling in architecture reconciliation: A multi case study, *Information and Software Technology*, Vol. 61, 2015, 107–123, ISSN: 0950-5849.

[8] B. De Smet, *C# 5.0 Unleashed*, Pearson Education Inc., 2013, ISBN: 0672336901.

[9] J. Galloway, P. Haack, B. Wilson, K. S. Allen, *Professional ASP.NET MVC 4*, John Wiley & Sons, Inc., Indianapolis, Indiana, 2012, ISBN: 111834846X.

[10] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1994, ISBN: 0201633612.

[11] D. Gurunule, M. Nashipudimath, A Review: Analysis of Aspect Orientation and Model Driven Engineering for Code Generation, *Procedia Computer Science*, Vol. 45, 2015, 852–861, ISSN: 1877-0509.

[12] T. Haitzer, E. Navarro, U. Zdun, Reconciling software architecture and source code in support of software evolution, *Journal of Systems and Software*, Vol. 123, 2017, 119–144, ISSN: 0164-1212.

[13] http://php.net/, Aug. 2016.

[14] N. Matsubara, K. Nakakoji, Y. Shirai, Y. Yamamoto, Toward Unweaving Streams of Thought for Reflection in Professional Software Design, *IEEE Software*, Vol. 29, 2012, 34–38, ISSN: 0740-7459, doi:10.1109/MS. 2011.125.

[15] I. Mistrik, R. Soley, J. Grundy, B. Tekinerdogan, N. Ali, *Software Quality Assurance*, Morgan Kaufmann, 2016, ISBN: 978-0-12-802301-3.

[16] M. Mouzarani, B. Sadeghiyan, Towards designing an extendable vulnerability detection method for executable codes, *Information and Software Technology*, Vol. 80, 2016, 231–244, ISSN: 0950-5849.

[17] B. Smith, *Procedural Reflection in Programming Languages*, Massachusetts Institute of Technology, PhD Thesis, 1982.

[18] O. Yazdanbakhsh, S. Dick, I. Reay, E. Mace, On deterministic chaos in software reliability growth models, *Applied Soft Computing*, Vol. 49, 2016, 1256–1269, ISSN: 1568-4946.