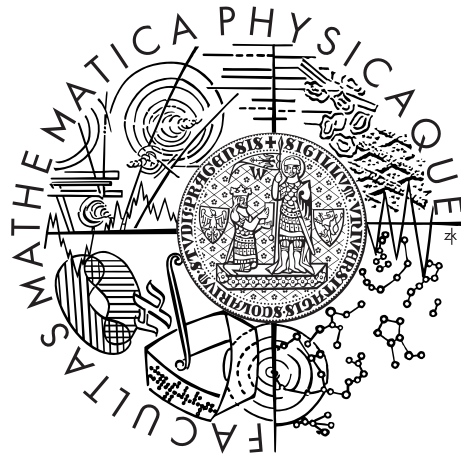


Charles University in Prague  
Faculty of Mathematics and Physics

## EXTENDED ABSTRACT



Štěpán Šindelář

## Design Patterns Support in Development Environments

The Department of Software Engineering

Supervisor of the bachelor thesis: RNDr. Filip Zavoral, Ph.D.

Study programme: Computer Science

Specialization: Programming

Prague 2011

# 1 Introduction

A pattern in object-oriented design is a description of communicating objects and classes that are customized to solve a general design problem in a particular context.

The main aim of patterns in object-oriented design is to make the design reusable and flexible. This is very important because changes in the functional requirements of software during the development, or requests for new features in an already developed software are quite usual these days.

One of the disadvantages of design patterns is that they bring a new complexity into the design. This complexity is caused by an introduction of new classes and interfaces in order to provide better flexibility and reusability. Developers often don't have enough time to create a textual documentation for their classes and so the mapping between classes and a design patterns is lost. The source code, or reverse-engineered diagrams, does not emphasize the design patterns structure, which would provide more abstract view and thus tackle some of the complexity.

Even if the implemented patterns are documented, an incorrect understanding of some design patterns may slow down the development process or even lead to an introduction of software bugs in the system. For instance, when one part of the system expects the objects of a specific type to be immutable, but a developer unaware of what the immutability means changes this behavior.

While tools for a formal verification and tools for tackling the complexity of design patterns exist, they were mainly developed as research prototypes and, except for few of them, they didn't get enough attention from the industry. Moreover, most of these tools target the Java platform, but only few target the .NET platform.

In the thesis, we further discuss the problems of existing tools for the support of design patterns. These problems are addressed by the Patterns4Net project, whose presentation is the main aim of the thesis. Besides this, we also provide a brief overview of existing approaches for design patterns formalization, which is needed for formal verification and tool support, and we give a few examples of existing tools that provide support for design patterns.

## 2 Patterns4Net

Patterns4Net is a set of tools that support the development of object oriented software on the .NET platform. These tools take advantage of a special documentation about implemented design patterns which is expressed using .NET attributes.

Patterns4Net provides two main tools. Pattern Enforcer verifies some of the structural aspects of selected design patterns implementation and Architecture Explorer generates interactive UML-like class diagrams from .NET assemblies. This tool uses the information about implemented design patterns to generate more abstract and high-level diagrams than standard UML reverse engineering tools.

**Pattern Enforcer** Pattern Enforcer provides a command line interface and also a MSBuild task, which enables an integration with Visual Studio. For the specification of structural aspects enforced on design patterns implementations, so called type-safe domain specific language embedded into the C# language was developed. This special C# API is used for specification of 14 built-in patterns, but can be also used by users to create custom patterns.

Pattern Enforcer was mainly inspired by the Pattern Enforcing Compiler for Java (PEC). In the thesis we describe the features of this software and compare it to our Pattern Enforcer. Some of the weaknesses of PEC that were admitted by its authors were solved in our Pattern Enforcer. Besides the PEC, we give a brief overview of other patterns verification software and we conclude that Pattern Enforcer is the only one tool of this type for the .NET platform we are aware of.

**Architecture Explorer** Architecture Explorer leverages the patterns documentation to generate interactive UML-like class diagrams that support a notion of zooming in and out which adds or removes details from the diagram. Such way a developer can have a general overview of the architecture or he can zoom to a specific class and see all related classes. The decision whether class should be displayed in the general overview or whether it should be displayed only in the highest zoom is based on the patterns roles it implements. Some patterns represents rather an infrastructural detail, on the other hand, for instance, patterns that are connected with the Domain-Driven-Design approach are usually represented by domain specific classes.

The idea of human-aided reverse engineering of design patterns is, to our best knowledge, unique. However, similar approaches exist (e.g., Model-Driven-Development). In the thesis we briefly discuss these approaches and tools and compare them to Architecture Explorer.

### 3 Conclusion and Future Work

Patterns4Net might enhance the development process of complex design patterns oriented systems that are created by a larger team, because it helps to discover communication errors and violations of design patterns implementations earlier and it provides visual tool to tackle some of the design complexity that is caused by design patterns usage.

Some of the more general rules from Pattern Enforcer, such as immutability check, could be extracted from it's source and proposed to open-source community as additional rules for well-established open-source project Gendarme.

Software systems are getting larger and more complex and this trend will continue. Changes in requirements are usual and reusability is important. Design patterns provide widely accepted approach for tackling the complexity of large systems and with tools such as Patterns4Net we can get even more advantages from their usage.