ISSN 2658-5014 (Print) СИИТ ISSN 2686-7044 (Online)
2024. Т. 6, № 2 (17). С. 72–77 http://siit.ugatu.su
СИСТЕМНАЯ ИНЖЕНЕРИЯ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
*Научное сообщение*

УДК 004.9 DOI 10.54708/2658-5014-SIIT-2024-no2-p72 EDN OWVACS

# Автоматизированный инструмент для рефакторинга логов в программном обеспечении

**С. А. Надеев**

**Аннотация.** В последние годы наблюдается рост научных публикаций по логированию программного обеспечения, что указывает на увеличивающийся интерес к этой теме. В частности, в областях, такие как авиация, где надежность критически важна, требования к мониторингу программ еще выше, что подчеркивает значимость работы с логами для обеспечения безопасности и стабильности программных систем. Качественное логирование обеспечивает точность, полноту и удобство логов для обнаружения и устранения ошибок, мониторинга работы программы и анализа событий. Исследования в этой области могут привести к разработке более эффективных методов логирования, улучшая процессы разработки и обслуживания программного обеспечения. В целях повышения эффективности логирования в статье предлагается новый инструмент для рефакторинга логов в исходном коде программы, основанный на их приведении к единому формату путем автоматизации методом статического анализа. Предлагаемый инструмент проходится по файлам программы и выполняет три основные задачи: находит переменные, используемые для ведения логов, добавляет аргументы в вызовы логгера и анализирует сообщения, записываемые логгером, на наличие переменных. Такой подход позволяет стандартизировать процесс логирования и улучшить информативность логов. Также в статье рассматриваются общие недостатки существующих решений, которые заключаются в том, что добавление дополнительных параметров в существующие логи не предполагает стандартизации формата, что усложняет анализ данных, поскольку различные форматы соответствуют разным типам информации и их структурам, что затрудняет выявление проблемных мест и отладку ошибок.

**Ключевые слова:** логирование; статический анализ; рефакторинг; .NET; Roslyn.

## Введение

В последние годы наблюдается увеличение научных публикаций в области логирования программного обеспечения (ПО), что свидетельствует о растущем интересе к этой теме. Под логированием понимается сохранение записей о событиях в программе с целью дальнейшего анализа. Например, сервер может сохранить в отдельном файле информацию о том, когда был получен запрос от пользователя. Таким образом, логирование представляет собой важный аспект разработки, эксплуатации и обслуживания ПО, выступая здесь не только как обязательный инструмент тестирования, но и как ключевой источник информации для мониторинга производительности и работы программы в целом [1].

Качественное логирование предполагает точность, полноту и удобство логов для чтения, что облегчает выявление и исправление ошибок программного кода, мониторинг работы программы и анализ происходящих при этом событий. Оно позволяет лучше адаптироваться к использованию современных технологий и методологий программирования, поддерживая внедрение новых методов, инструментов и технологий для сбора, анализа и визуализации данных. Исследования в этой области могут привести к разработке более эффективных и надежных методов логирования, что в свою очередь улучшит процессы разработки, обслуживания и поддержки программ.

## Актуальность области исследования

Работа [2] посвящена комплексному исследованию ПО для поддержки логирования и анализу этой области с целью выявления текущих тенденций. Авторы стараются решить проблему разрозненности предыдущих исследований по логированию, препятствующей эффективному развитию данной сферы. Приводятся данные о росте числа научных публикаций и конференций по различным аспектам логирования. Среди наиболее популярных исследовательских тем авторы выделяют анализ логов для обнаружения аномалий и сбоев в работе программы, автоматизацию добавления логов, их обработку и др.

В статье [3] содержится анализ исследований по автоматизации логирования с целью более эффективного программирования. По данным о публикациях в период с 1997 по 2020 год отмечается, что с 2015 г. происходит рост количества статей в данной области. Авторы выделяют и три основные проблемы логирования, которые могут быть сведены к вопросам: где, что и как логировать?

Рост популярности темы логирования обусловлен увеличением объемов данных и цифровизацией, повышенным вниманием к безопасности [4, 5] и надежности ПО. Например, в 2017 г. сутки простоя из-за программных сбоев обошлись компании Amazon потерей в более 150 млн долл. [3]. А в таких важных областях, как авиация или медицина, надежность еще более важна, что повышает и требования к мониторингу текущего состояния соответствующих программ [2].

Приобретают актуальность также исследования по логированию применительно к конкретной практике современного программирования. Так, на примере ведения логов в компании Microsoft осуществлена попытка выявления реальных потребностей проекта и имеющегося значительного разрыва между теорией и практикой по вопросам ведения логов [6].

Исследуются также вопросы, связанные с обнаружением признаков нестандартного поведения работы программы – аномалий. Например, в работе [7] указывается на то, что обнаружение аномалий в логах программы играет важную роль в управлении современных крупномасштабных систем. Однако разработчики ПО в большинстве случаев не обладают знаниями о предпочтительности тех или иных методов автоматизации обнаружения аномалий. Для решения этой проблемы авторы приводят подробный обзор и оценку современных методов обнаружения аномалий в логах программы.

Таким образом, исследования по проблемам логирования становятся весьма актуальными, вызывая значительный интерес со стороны теории и практики современного программирования, направленный, в том числе, на выявление аномалий и поддержку автоматизации генерируемого кода.

## Средства улучшения качества логирования

Известна программа, которая в процессе изменения кода дает рекомендации по добавлению логов, предоставляемые на основе использования классификаторов машинного обучения [8].

Также в работе [9] предлагается инструмент для автоматизации изучения распространенных методов логирования из существующих хранилищ кода, который был внедрен в программу LogAdvisor, помогающую разработчикам выбирать место добавления лога. Для изучения практик логирования применяется набор моделей машинного обучения посредством извлечения различных признаков.

Особенностью данных разработок, ограничивающей сферу их приложения, является использование методологии машинного обучения, а также специализация на добавлении новых логов в код программы.

В статье [10] описывается инструмент LogEnhancer, который автоматизирует изменения существующих логов, добавляя к ним набор дополнительной информации, что может быть полезно при тестировании программы. Данный инструмент анализирует код методом статического анализа с помощью анализатора Saturn [11]. Идея алгоритма, по утверждению авторов,

заключается в том, чтобы сначала определить причинно-следственные связи ветвей с каждым местом логирования, а затем вывести необходимые данные. Анализатор Saturn используется также инструментом ErrLog, который анализирует исходный код программы для выявления потенциальных исключений, а затем добавляет их в логи [12].

Общий недостаток данных решений состоит в том, что реализуемое ими добавление в существующие логи дополнительных параметров не предполагает приведения обновляемых логов к единому формату. Отсутствие такого формата логов приводит к затруднениям при анализе данных, так как различным форматам соответствуют различные типы информации и их структуры, что усложняет выявление проблемных мест и отладку ошибок в программе. Неоднородность данных также увеличивает затраты времени на обработку логов и вероятность их ошибочного анализа.

Далее предлагается инструмент логирования, свободный от этих недостатков.

## Инструмент рефакторинга логов

Как известно, рефакторинг – это процесс улучшения кода программы путем его переписывания без изменения функциональности, который помогает сделать код более понятным и упрощает его дальнейшее развитие. С помощью рефакторинга можно изменять формат существующих логов, облегчая их чтение, анализ и поддержку.

Далее описывается инструмент рефакторинга или изменения существующих логов в исходном коде программы на платформе .NET методом статического анализа. Использование инструмента позволяет приводить логи программы к единому формату, а также обновлять и добавлять в них новые параметры.

### 1. Используемый метод

Как известно, статический анализ исходного кода программы направлен на выявление потенциальных ошибок и уязвимостей в программе без ее выполнения. Анализ осуществляется при помощи специальных инструментов, проверяющих программный код на соответствие
заданным правилам и стандартам, что позволяет выявлять потенциальные проблемы еще на этапе написания программы.

В контексте рефакторинга статический анализ используется для обнаружения неправильных конструкций кода. При этом улучшаются качество и поддержка кода, уменьшается количество ошибок, а за счет автоматизации процесса повышается производительность труда разработчиков.

В контексте логирования статический анализ может быть использован для выявления ошибок, неэффективного использования или недостаточной информативности логов. Это помогает автоматически обнаруживать места, где происходит логирование некорректных данных, отсутствует логирование важных событий или исключений, а также имеется дублирование сообщений в различных частях кода.

Таким образом, статический анализ может значительно улучшить качество логирования, повысить информативность логов и облегчить процесс отладки и мониторинга приложения.

### 2. Архитектура

Для достижения вышеуказанных целей используется инструмент для статического анализа Roslyn [13]. Он проходится по файлам, указанным пользователем программы, и выполняет три основные задачи:

- В каждой функции программы инструмент пытается найти переменную, используемую инструментом для ведения логов – логгером. Названия таких переменных пользователь указывает перед запуском программы. При этом предполагается, что в целях стандартизации практически во всех файлах программы каждая переменная имеет небольшое количество наименований, и чаще всего – одно.
- В найденные вызовы логгера добавляются аргументы.

- Происходит анализ сообщения, которое записывает логгер, на наличие переменных. Если удается определить, что переменная относится к определенному критерию, то она также добавляется как отдельный аргумент в приведенном ниже формате. Разделение на категории осуществляется путем поиска соответствующей подстроки в названии переменной:

```
[Название критерия]-переменная.
```

Переменные добавляются с помощью предоставленных методов анализатора на языке программирования C#.

```
var argumentSyntax = SyntaxFactory.Argument(
SyntaxFactory.InterpolatedStringExpression(
   SyntaxFactory.Token(SyntaxKind.InterpolatedStringStartToken))
   .WithContents(
         SyntaxFactory.SingletonList<InterpolatedStringContentSyntax>(
               SyntaxFactory.InterpolatedStringText()
               .WithTextToken(
                     SyntaxFactory.Token(
                           SyntaxFactory.TriviaList(),
                           SyntaxKind.InterpolatedStringTextToken,
                           argument,
                           argument,
                           SyntaxFactory.TriviaList())))));
```

Здесь `argument` – это название добавляемого аргумента.

На данный момент набор критериев строго задан в коде программы: role, request, response, workorder, event, state, exception, name, id. В этот список входят как общезначимые, подходящие для большинства программ (id, name, exception, event, role), так и специфические критерии (request, response, workorder, state).

### 3. Пример работы

В следующем примере кода на языке программирования C# к аргументам логгера добавляются названия класса и функций. В отдельный аргумент выделяется переменная Roles.TestRole, для которой автоматически определилась категория – role.

```
_logger.Info($"Log message for {Roles.TestRole}"); // вид лога до изменения
_logger.Info($"{nameof(Test)}",
             $"{nameof(Test.MethodTest)}",
             $"Log message for {Roles.TestRole}",
             $"role-{Roles.TestRole}"); // вид лога после изменения
```

### 4. Практическая ценность

Такой формат логов удобен для преобразования в JSON – универсальный формат хранения и передачи данных. Он позволяет легко манипулировать логами с помощью соответствующих инструментов, таких как ELK stack [14], для последующего их анализа и обработки.

Выделение в отдельные аргументы названий класса, функций и переменных из сообщения позволит эффективно фильтровать данные в системе ELK stack. К примеру, фильтрация по идентификатору и названиям класса позволит определить, проходил ли данный объект (договор, заявка, запрос и т. д. – в зависимости от программы) через набор определенных стадий обработки.

Инструмент также будет полезен для проектов, уже находящихся в разработке, где требуется изменить формат логов. Благодаря автоматизации процесс занимает гораздо меньше вре-

мени, чем если бы разработчики меняли логи вручную. Например, в проекте, содержащем более чем 200 000 строк кода на языке C#, инструмент способен изменить 456 логов всего за одну минуту.

### 5. Дальнейшие перспективы

Инструмент на данный момент находится в активной разработке. Предполагается, что в последующих версиях программы будут сняты ряд текущих ограничений:

- Инструмент доступен только для .NET платформы. Данное ограничение связано с тем, что на этой платформе реализован анализатор, используемый инструментом для статического анализа.
- После изменения логов требуются незначительные ручные правки в функциях используемого логгера. Данное ограничение обусловлено зависимостью вносимых корректив от особенностей программы.
- Набор критериев, на которые разделяются параметры, строго фиксирован. В дальнейшем планируется создать функцию, с помощью которой пользователь получит возможность добавлять необходимые ему правила разделения параметров на критерии, учитывающие особенности программы.
- За исключением названий класса и метода, отсутствует возможность добавления дополнительной информации в аргументы логгера. Планируется расширить функционал инструмента за счет предоставления пользователю возможности самому добавлять код, учитывающий требования как можно большего числа приложений.

## Заключение

В данной работе представлен инструмент рефакторинга существующих логов в исходном коде программы на платформе .NET методом статического анализа. Инструмент будет полезен для проектов, уже находящихся в разработке, где требуется изменить формат логов. Благодаря автоматизированной обработке логов с помощью инструмента удается сократить время, затрачиваемое на процесс программирования.

## Благодарности

Автор благодарит научного руководителя д-ра филос. наук, канд. техн. наук, доц. Р. А. Ярцева за полезные замечания в ходе работы над статьей.

## Список литературы / References

1. Schmidt K., Phillips C., Chuvakin A. Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management. Newnes, 2012.

2. Gholamian S., Ward P. A. S. A comprehensive survey of logging in software: From logging statements automation to log mining and analysis // arXiv preprint arXiv: 2110. 12489. 2021.

3. Chen B., Jiang Z. M. A survey of software log instrumentation // ACM Computing Surveys (CSUR). 2021. V. 54. No. 4. Pp. 1–34.

4. Орлов Г. О. Подход к обеспечению безопасности программного кода в веб-ориентированной среде // СИИТ. 2023. Т. 5. № 5(14). С. 68–77. EDN: HPCIMR. [[ Orlov G. O. “An approach to ensuring the security of program code in a web-oriented environment” (in Russian), System Engineering and Information Technologies. 2023. Vol. 5, No. 5(14), pp. 68-77. EDN: HPCIMR. (In Russian). ]]

5. Васильев В. И., Картак В. М. Применение методов искусственного интеллекта в задачах защиты информации (по материалам научной школы УГАТУ) // СИИТ. 2020. Т. 2. № 2(4). С. 43–50. EDN: ZTQFCW. [[ Vasiliev V. I., Kartak V. M. “Application of artificial intelligence methods in information security problems (based on materials from the scientific school of UGATU)” (in Russian) // SIIT. 2020. Vol. 2, No. 2(4), pp. 43-50. EDN: ZTQFCW. (In Russian). ]]

6. He S. et al. An empirical study of log analysis at Microsoft // Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022. Pp. 1465–1476.

7. He S. et al. Experience report: System log analysis for anomaly detection // 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). IEEE. 2016. Pp. 207–218.

8. Li H. et al. Towards just-in-time suggestions for log changes // Empirical Software Engineering. 2017. V. 22. Pp. 1831–1865.

9. Zhu J. et al. Learning to log: Helping developers make informed logging decisions // 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE. 2015. V. 1. Pp. 415–425.

10. Yuan D. et al. Improving software diagnosability via log enhancement // ACM Transactions on Computer Systems (TOCS). 2012. V. 30. No. 1. Pp. 1–28.

11. Aiken A. et al. An overview of the Saturn project // Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. 2007. Pp. 43–48.

12. Yuan D. et al. Be conservative: Enhancing failure diagnosis with proactive logging // 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12). 2012. Pp. 293–306.

13. Vasani M. Roslyn Cookbook. Packt Publishing Ltd, 2017.

14. Chhajed S. Learning ELK Stack. Packt Publishing Ltd, 2015.

*Поступила в редакцию 8 апреля 2024 г.*

**МЕТАДАННЫЕ / METADATA**

**Title:** Automated tool for refactoring logs in software.

**Abstract:** In recent years, there has been an increase in scientific publications on software logging, which indicates an increasing interest in this topic. Particularly in areas such as aviation, where reliability is critical, the requirements for program monitoring are even higher, highlighting the importance of working with logs to ensure the security and stability of software systems. High-quality logging ensures the accuracy, completeness and convenience of logs for detecting and eliminating errors, monitoring program operation and analyzing events. Research in this area could lead to the development of more efficient logging methods, improving software development and maintenance processes. To increase the efficiency of logging, the article proposes a new tool for refactoring logs in the source code of a program, based on bringing them to a single format through automation using the method of static analysis. The proposed tool walks through program files and performs three main tasks: finds variables used for logging, adds arguments to logger calls, and analyzes messages recorded by the logger for the presence of variables. This approach allows you to standardize the logging process and improve the information content of the logs. The article also discusses the general disadvantages of existing solutions, which are that adding additional parameters to existing logs does not imply standardization of the format, which complicates data analysis, since different formats correspond to different types of information and their structures, which makes it difficult identifying problem areas and debugging errors.

**Key words:** logging; static analysis; refactoring; .NET; Roslyn.

**Язык статьи / Language:** русский / Russian.

**Об авторе / About the author:**

**НАДЕЕВ Сергей Александрович**
ФГБОУ ВО «Уфимский университет науки и технологий», Россия.
Магистрант ин-та информатики, математики и робототехники. Дипл. программный инженер (Уфимск. гос. авиац. техн. ун-т, 2022).
E-mail: nadeevsa@mail.ru

**NADEEV Sergey Alexandrovich**
Ufa State Aviation Technical University (UGATU), Russia.
Master's student at the Inst. of Computer Science, Maths and Robotics. Dipl. Software Engineer (Ufa State Aviation Technical University, 2022).
E-mail: nadeevsa@mail.ru