УДК 621.3.049.77–048.24:537.2

МИНИМИЗАЦИЯ СЛОЖНОСТИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Скиб А.И.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Мигалевич С.А. – магистр технических наук, ст. преподаватель кафедры ПИКС

Аннотация. Статья рассматривает вопрос минимизации сложности разработки программного обеспечения. Авторы обсуждают различные подходы и методы упрощения процесса разработки, в том числе использование библиотек и фреймворков, автоматизацию тестирования и т.д. Результатом исследования является вывод о том, что минимизация сложности разработки помогает ускорить процесс, улучшить качество программного обеспечения и повысить его эффективность.

Ключевые слова: сложность, программное обеспечение, разработка, минимизация

Введение. Программное обеспечение играет огромную роль в современной жизни, оно используется во всех отраслях, начиная от медицины и заканчивая промышленностью. Разработка программного обеспечения является длинным и сложным процессом, который связан с большим количеством проблем. Одной из основных проблем, с которой сталкиваются разработчики, является сложность процесса [1].

Цель данной статьи – выяснить, каким образом можно минимизировать сложность разработки программного обеспечения.

Основная часть. Один из способов минимизации сложности разработки программного обеспечения – применение методологий разработки. Такие методики, как *Agile* и *Scrum*, стали популярными в последние годы и помогают упростить процесс разработки. *Agile* подразумевает быстрое внесение изменений и приспособление к ситуациям, которые могут возникнуть. *Scrum* предусматривает разбиение проекта на небольшие этапы, называемые “спринтами”, чтобы быстрее выявить проблемы и внести корректировки.

Еще один способ минимизации сложности – использование инструментов автоматизации. Разработчики могут использовать такие инструменты, как *JIRA* и *REDMINE* для управления проектами, а также *Jenkins* и *Travis CI* для автоматизации процесса сборки и тестирования программного обеспечения [2,3].

Важным аспектом в упрощении процесса разработки является правильный выбор технологий и фреймворков. Разработчики должны выбирать фреймворки, которые лучше подходят для решения конкретных задач. Например, фреймворк для создания веб-приложений *Ruby* on *Rails* очень популярен, потому что он достаточно прост в использовании и позволяет быстро создавать веб-приложения. Однако, для решения более комплексных задач, таких как машинное обучение, разработчики могут выбирать более специализированные фреймворки, например, *TensorFlow* или *PyTorch*.

Также важно учитывать риски и ошибки, которые могут возникнуть в процессе разработки. Разработчики должны быть готовы к тому, что некоторые идеи или подходы могут не работать. Важный аспект – это последовательное тестирование программного обеспечения, чтобы избежать ошибок на стадии релиза. Тестирование должно быть автоматизировано, но также может проводиться и вручную.

Другим методом минимизации сложности разработки программного обеспечения является применение многоуровневой архитектуры. Многоуровневая архитектура хороша известна большинству архитекторов, проектировщиков и разработчиков. Ограничений по количеству и типу уровней никаких нет, однако в большинстве случаев такая архитектура состоит из четырех уровней: представление данных, бизнес-логика, хранение данных и база данных.

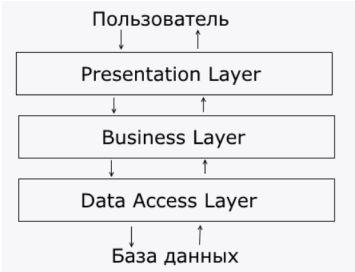

Рисунок 1 – Многоуровневая архитектура

Шаблоны проектирования также помогает разработчикам сократить сложность и повысить эффективность процесса создания программного обеспечения. Они позволяют упростить код и сделать его более понятным и поддерживаемым в будущем. Кроме того, они обеспечивают гибкость архитектуры приложения и снижают вероятность возникновения проблем с масштабированием и модификацией системы [4].

Принцип *SOLID* – это набор принципов проектирования программного обеспечения, направленые на минимизацию сложности кода и обеспечивают простоту его сопровождения. Кратко можно сформулировать эти принципы следующим образом:

– *S* - единство ответственности (*Single Responsibility Principle*);
– *O* - открытость/закрытость (*Open/Closed Principle*);
– *L* - замена Лискова (*Liskov Substitution Principle*);
– *I* - разделение интерфейса (*Interface Segregation Principle*);
– *D* - инверсия зависимости (*Dependency Inversion Principle*); [5]

Заключение. В целом, упрощение процесса разработки программного обеспечения является необходимой и важной задачей. Она может быть решена путем правильного проектирования, использования фреймворков, тестирования и автоматизации процессов.

Результатом является более эффективная и менее трудоемкая разработка программного обеспечения. Современные разработчики должны принимать во внимание лучшие практики, используемые в индустрии, и стремиться к постоянному улучшению своих проектов.

Список литературы

1. Kostoff, R.N. (1997) Complexity measures and software development costs: An empirical investigation. Journal of Information Science, 23, 373-380.
2. Joshua Bloch, Effective Java: Programming Language Guide, Addison-Wesley Professional, 2001.
3. Martin Fowler, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.
4. Steve McConnell, Code Complete: A Practical Handbook of Software Construction, Microsoft Press, 1993.
5. Р. Мартин. "Clean Code: A Handbook of Agile Software Craftsmanship", 2008.

UDC 621.3.049.77–048.24:537.2

MINIMIZING THE COMPLEXITY OF SOFTWARE DEVELOPMENT

Askiab A.I.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Migalevich S.A. – master of technical sciences, senior lecturer of the Department of ICSD

Annotation. The article considers the issue of minimizing the complexity of software development. The authors discuss various approaches and methods to simplify the development process, including the use of libraries and frameworks, test automation, etc. The result of the study is the conclusion that minimizing the complexity of development helps to speed up the process, improve the quality of the software and increase its efficiency.

Keywords: complexity, software, development, minimization