ISSN 2500-316X (Online)

https://doi.org/10.32362/2500-316X-2020-8-5-7-18

УДК 004.01, 004.415.2.052.03, 004.415.28

Применение высокоуровневых методов компромиссной оптимизации для управления автономной роботизированной добычей полезных ископаемых открытым способом

С.А. Головин[1],
С.В. Зыков[2, @],
Ю.П. Кораблин[1],
Д.А. Крюков[1]

[1]МИРЭА – Российский технологический университет, Москва 119454, Россия
[2]НИУ «Высшая школа экономики» Москва 101000, Россия
[@]Автор для переписки, e-mail: szykov@hse.ru

В большинстве подходов программной инженерии проектирование программного обеспечения начинается с определения функциональных требований, что вполне подходит проектам по разработке программного обеспечения, ориентированного на Web-приложения. При проектировании высококритичного крупномасштабного программного обеспечения, предназначенного для промышленного использования, необходим учет и нефункциональных требований. Основная идея предлагаемого документоориентированного подхода заключается в максимально раннем проектировании стабильного архитектурного решения, учитывающего нефункциональные характеристики программного обеспечения: надежность, безопасность, сопровождаемость и производительность (атрибуты качества). При этом ключевым вопросом является согласование функциональных требований с учетом ограничений технического характера и требований бизнеса, достигаемое в ходе устойчивого взаимодействия команд заказчика и разработчика. Для повышения гибкости конструируемых решений и предупреждения кризисных ситуаций при разработке высококритичного крупномасштабного программного обеспечения предлагается использовать подход, интегрирующий метод архитектурно-центричного проектирования (*Architecture-Centred Design Method*, ACDM), метод архитектурного анализа компро-

миссов (*Architecture-Tradeoff Analysis Method*, ATAM) с матрицей архитектуры предприятия (*Enterprise Architecture Matrix*, EAM). Это позволяет получить результат, адекватный требуемому уровню ответственности и надежности. Рассмотрение атрибутов качества в рамках метода анализа компромиссов дает возможность выбора и принятия определенных решений в проектировании программного обеспечения, учитывающих масштаб программного обеспечения и сферу его применения. Выделены основные атрибуты качества продукта (стандарт ISO 25010), для которых определены критичные сценарии. Использование указанных сценариев для детального проектирования программного обеспечения с необходимыми параметрами функциональных требований, бизнес-условий и технологических ограничений уменьшает риск развития непредсказуемого и неопределенного поведения системы. На основе предложенного подхода представлено архитектурное решение для высококритичного, ответственного, крупномасштабного программного обеспечения для управления автономной роботизированной добычей полезных ископаемых открытым способом. Выявлены и проранжированы критически важные атрибуты для создания указанного программного обеспечения и описана архитектура решения согласно стандарту разработки программного обеспечения SWEBOK. Далее, с учетом характера, масштаба и области применения программного решения даны рекомендации по высокоуровневым архитектурным решениям для проекта системы, включая слои, конвейеры и микросервисы. Предлагаемый архитектурно-ориентированный метод разработки подходит для программного обеспечения промышленного уровня различных предметных областей.

Ключевые слова: программное обеспечение, функциональные требования, атрибут качества, архитектура программного решения, метод архитектурно-центричного проектирования (ACDM), метод архитектурного анализа компромиссов (ATAM), матрица архитектуры предприятия (EAM).

Для цитирования: Головин С.А., Зыков С.В., Кораблин Ю.П., Крюков Д.А. Применение высокоуровневых методов компромиссной оптимизации для управления автономной роботизированной добычей полезных ископаемых открытым способом. *Российский технологический журнал.* 2020;8(5):7-18. https://doi.org/10.32362/2500-316X-2020-8-5-7-18

Application of high-level methods of compromise optimization for control of autonomous robotized open pit mining

Sergey A. Golovin[1],
Sergey V. Zykov[2, @],
Yurii P. Korablin[1],
Dmitry A. Kryukov[1]

[1]MIREA – Russian Technological University, Moscow 119454, Russia
[2]NRU "Higher School of Economics", Moscow 101000, Russia
[@]Corresponding author, e-mail: szykov@hse.ru

In most software engineering approaches, software design begins with defining functional requirements, which is well suited to web-based software development projects. When designing high-critical large-scale software intended for industrial use, accounting for non-functional

software requirements is also required. The main idea of the proposed document-oriented approach is to design a stable architectural solution as early as possible, taking into account the non-functional characteristics of the software: reliability, security, maintainability and performance (quality attributes). At the same time, the key issue is the coordination of functional requirements, taking into account technical limitations and business requirements achieved during the steady interaction of customer and developer teams. To increase the flexibility of the designed solutions and prevent crisis situations when developing highly critical large-scale software, it is proposed to use the approach integrating the architecture-centric design method (ACDM), the architecture-tradeoff analysis method (ATAM) with a matrix enterprise architecture matrix (EAM). This allows getting a result that is adequate to the required level of responsibility and reliability. Consideration of quality attributes within the framework of the method of compromise analysis makes it possible to select and make certain decisions in software design taking into account the scale of the software and its scope. The main attributes of product quality are highlighted (ISO 25010 standard), critical scenarios are defined for each of them (templates and use cases). The use of these templates for detailed software design with the necessary parameters of functional requirements, business conditions and technological limitations reduces the risk of developing unpredictable and uncertain system behavior. Based on the proposed approach, an architectural solution is presented for highly critical, responsible, large-scale software for managing autonomous robotic open-pit mining of minerals. Critical attributes for creating the specified software were identified and ranked, and the architecture of the solution according to the SWEBOK software development standard was described. Further, taking into account the nature, scale and scope of the software solution, recommendations are given on high-level architectural templates for the system design, including layers, pipelines and microservices. The proposed architecture-oriented development method is suitable for industrial-level software in various subject areas.

Keywords: software development, functional requirements, quality attribute, architecture of software solutions, architecture-centric design method (ACDM), method of architectural compromise analysis (ATAM), enterprise architecture matrix (EAM).

For citation: Golovin S.A., Zykov S.V., Korablin Yu.P., Kryukov D.A. Application of high-level methods of compromise optimization for control of autonomous robotized open pit mining. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2020;8(5):7-18 (in Russ.). https://doi.org/10.32362/2500316X-2020-8-5-7-18

Введение

Целью работы является планирование документноориентированной разработки высококритичного крупномасштабного программного обеспечения, предназначенного для промышленного использования и анализ конкретных факторов, включающих атрибуты качества и технологические ограничения, сформированные ключевыми участниками проекта. Подход, не учитывающий при проектировании программного обеспечения в полной мере системообразующие характеристики, как правило, в подобных случаях приводит к негативным результатам. В первую очередь это касается рабочих параметров программных продуктов, как с точки зрения неоправданно высокой стоимости, так и в отношении соответствия решаемым задачам. Предлагаемый подход направлен на систематический учет достаточно широкого спектра основных технический требований и бизнес-ограничений.

В целях повышения гибкости конструируемых решений и предупреждения возможных кризисных ситуаций при разработке программного обеспечения предлагается интеграция архитектурно-центричного метода разработки (*Architecture-Centred Design*

Method, ACDM) [11, 12] и сервисно-ориентированной архитектуры (*Service-Oriented Architecture*, SOA). При этом наилучший синергический эффект планируется получить за счет применения наиболее гибкой и новой версии SOA в форме микросервисной архитектуры [13].

В основу разработанного архитектурного решения положен подход к разработке, известный как архитектурно-центричный метод ACDM, выбранный по следующим причинам: данный метод относится к целеориентированной разработке программного обеспечения, отвечающего заранее определенным целевым характеристикам, включающим «нефункциональные» требования, такие как надежность, безопасность, сопровождаемость и производительность. Такого рода требования, известные иначе как «атрибуты качества» (*Quality attributes*, QA), составляют в совокупности единый – архитектурный фактор процесса разработки. Другие факторы «верхнего уровня» включают в себя ограничения технического характера, а также бизнес-ограничения. Перечисленные виды ограничений зачастую оказываются критически значимыми для аналитических решений, эксплуатационных характеристик, возможностей применения, выбора инструментария, назначения ресурсов и других особенностей разработки высококритичного крупномасштабного программного обеспечения, предназначенного для промышленного использования, например, программного обеспечения управления автономной роботизированной добычей полезных ископаемых открытым способом.

В процессе разработки рассматриваемого класса программного обеспечения сосредоточимся на описании ключевых методов и технологий интеллектуальной геоинформационной платформы, предназначенной для управления транспортно-технологическими процессами при добыче минерального сырья. Рассматриваемые процессы являются сложной инженерно-технологической программно-аппаратной системой, включающей в себя различные типы устройств и механизмов, описание функционирования которых в совокупности не вполне точно представимо аналитически.

В то же время данная система на верхнем уровне ее представления может быть охарактеризована следующими особенностями:

- распределение основных параметров в многомерном пространстве;
- нелинейность и стохастичность отношений между основными параметрами, обусловленная частыми изменениями как способов добычи, так и технологической и горно-геологической среды;
- отсутствие универсального формального критерия для оптимизируемых функций;
- использование различных эмпирических и экспертных правил при планировании горных работ и управлении процессами добычи и транспортировки минерального сырья.

Перечисленные особенности разрабатываемого программного обеспечения с учетом их совокупности и взаимовлияния значительно усложняют корректность требований к программному обеспечению. Кроме того, разработка программного обеспечения в постоянном взаимодействии с конечным пользователем продукта может вызывать сложности.

Рассмотрение атрибутов качества в ходе анализа компромиссов предоставляет возможность более адекватного выбора и принятия более обоснованных решений при проектировании программных продуктов. При этом анализ существующих методов оптимизации компромиссов подчеркивает такие преимущества архитектурно-ориенти-

рованного метода, как значительное облегчение переговоров с заинтересованными сторонами, возможность раннего анализа рисков и, как следствие, снижение рисков на начальном этапе – архитектурном проектировании, которое зачастую является критической стадией разработки [1, 2]. В этой связи, осведомленность, согласованность и одобрение заинтересованных сторон, являются важными факторами успеха программного проекта, в особенности при интенсивной разработке крупномасштабного и ответственного программного обеспечения. Исследования показали, что заинтересованные стороны используют известный подход на основе архитектурно-центричного метода; в настоящей работе предлагается усовершенствовать этот подход за счет синергии с матрицей архитектуры предприятия (*Enterprise Architecture Matrix*, EAM) [13].

Учитывая все вышеизложенное, предлагаемый в данной статье подход на основе высокоуровневой компромиссной оптимизации представляется продуктивным.

Подход на базе архитектурных компромиссов: условия и принципы

Архитектурно-ориентированный метод предполагает, что функционал разрабатываемого программного обеспечения должен соответствовать бизнес-модели. Такое соответствие обеспечивается благодаря систематическому учету комплекса формирующих архитектуру программного обеспечения факторов, включающих в себя как атрибуты качества (QA), так и ограничения с точки зрения технологий и бизнес-требований. Что касается атрибутов качества, стандарт ISO 25010 классифицирует их следующим образом [1–3, 10]:

- качество продукта;
- качество процесса;
- качество использования.

С учетом характера и масштаба программного обеспечения, а также сферы его применения, представляется целесообразным выделить следующие важнейшие атрибуты качества, соответствующие основной категории стандарта ISO 25010:

- функциональная пригодность;
- производительность;
- эргономичность;
- надежность;
- безопасность;
- сопровождаемость;
- совместимость.

Заметим, что на базе упомянутого стандарта предлагается в дополнение к имеющейся в нем классификации:

- формирование приоритетов для атрибутов качества;
- создание «сценариев качества» для каждого из значимых атрибутов качества в помощь разработчикам программных систем (прежде всего, архитекторам).

Подобные «сценарии качества» позволяют явно выделить типовые ситуации, возникающие в том числе при эксплуатации высококритичного по качеству программного обеспечения. При этом архитектурно-ориентированный метод разработки становится не только теоретически адекватным, но и практически полезным, так как предоставляет конкретные пути применения и сценарные варианты использования.

В критических условиях, включающих высокую загруженность, интенсивное использование, а также особые требования к безопасности и надежности, архитектурно-ориентированный метод разработки помогает избежать стратегических ошибок при выполнении архитектурного проектирования крупномасштабного программного обеспечения (в т. ч. в форме «просчетов» и «подводных камней»). Такое преимущество обеспечивается благодаря наличию тщательно разработанных, сбалансированных сценариев, предназначенных для архитектурного и детального планирования и проектирования крупномасштабного программного обеспечения. При этом предлагаемый подход обеспечивает возможность последующего обобщения и распространения указанных сценариев на весь жизненный цикл разработки с необходимыми параметрами в функциональных и бизнес-требованиях, а также технологических ограничениях. Сценарно-ориентированная, управляемая документами разработка программного обеспечения снижает риски проектных ошибок, которые могут привести к развитию системы по пути непредсказуемого (или недокументированного) поведения. При этом в ряде иных, как правило, более строго формализованных, методов отсутствует детальный анализ упомянутых ограничений программного обеспечения на основе атрибутов качества, традиционно называемых «нефункциональными» требованиями.

Другим важным преимуществом предлагаемого подхода на основе архитектурно-ориентированного метода является его ориентированность на «мягкие навыки», включающие процессы взаимодействия, обмена знаниями и технологиями между командой разработчиков, и всеми заинтересованными лицами. При этом в основу подхода положено динамическое назначение и ранжирование совокупности атрибутов качества программного обеспечения, а также экспресс-оценка трудоемкости проектных задач. Прозрачная интеграция подхода со стандартными моделями жизненного цикла (такими как, например, спиральная и эволюционная, описанными в ГОСТ Р ИСО/МЭК ТО 15271-2002), а также с наиболее распространенными методологиями разработки крупномасштабного программного обеспечения (в частности, MSF и RUP, а также *Agile*), открывает возможности гибкого и надежного проектирования ответственного программного обеспечения для указанной предметной области и соответствия требованиям современных стандартов.

В дополнение к перечисленным «гибким» аспектам, предлагаемый подход на основе архитектурно-ориентированного метода разработки учитывает «жесткие» проектные ограничения технического и прикладного характера, которые находятся вне сферы влияния команды разработки и других заинтересованных лиц. Такие ограничения включают *a priori* детерминированные процессы, а также предопределенные инструментальные средства и программно-аппаратные платформы. Примерами подобных ограничений могут служить:

- степень компетентности аналитиков в отношении предметной области (прежде всего, в условиях существенной экономии проектных ресурсов);
- предопределенный язык программирования;
- параллельное развертывание продукта на различных программно-аппаратных платформах.

Процесс проектирования программного обеспечения с учетом особенностей предметной области

Основной процесс архитектурно-ориентированного метода включает представление ключевых компонентов системы и особенностей их взаимодействия (рис. 1) [12]. При

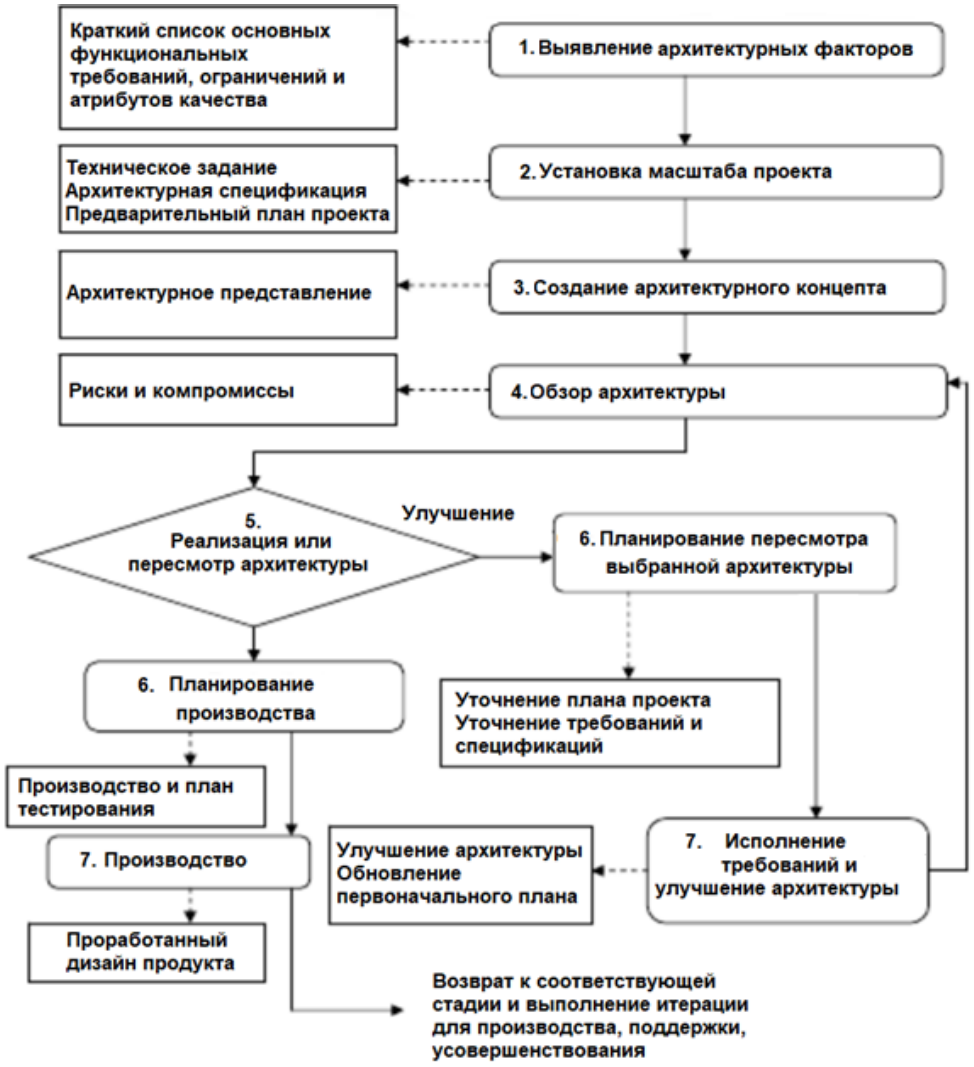

Рис. 1. Базовый процесс архитектурно-ориентированного метода.

этом предполагается последовательное уточнение упомянутых проектных артефактов на уровне ключевых компонентов продукта и их взаимосвязей. Процесс включает семь этапов, которые условно разделяются на фазы «неопределенности» (№№ 1–4) и «определенности» (№№ 6–7). Фаза № 5 считается «кризисной» и предполагает принятие решения о подтверждении или пересмотре выбранной архитектуры [12, 13].

Принимая во внимание предметную область, характер и масштаб рассматриваемого программного обеспечения, представляется целесообразным считать важнейшей фазой создание высокоуровневой концептуальной схемы для проектируемой системы в форме архитектурного проекта. При этом концептуальная схема архитектуры должна включать определенные артефакты, которые выявляются и детализируются в процессе проектирования, основанном на принципах архитектурно-ориентированного метода. На данном этапе упомянутые артефакты должны строиться с учетом таких ключевых особенностей проектируемого программного обеспечения, как:

1) среда выполнения;
2) представление кода;
3) физическое представление системы.

При этом каждый из аспектов (1–3) характеризуется определенным набором атрибутов качества с учетом их ранжирования по степени значимости. В частности, детализация представления среды выполнения позволяет разработчикам анализировать такой атрибут качества, как надежность программного обеспечения. В то же время, в ходе детализации представления на уровне кода, разработчикам следует использовать формальные методы для проверки корректности и эксплуатационных характеристик программного обеспечения.

При этом в целях повышения качества и минимизации ошибок архитектурного проектирования рекомендуется согласовывать вышеприведенные представления с тремя уровнями матрицы архитектуры предприятия, а именно:

- процессы;
- данные;
- компоненты.

С учетом особенностей разрабатываемого программного обеспечения, в т. ч. сферы его применения, в числе приоритетных атрибутов качества (QA) были выделены следующие:

- QA1 – модифицируемость (т.е. переносимость и расширяемость) – в целях интеграции компонент разнородного программного обеспечения в единую информационную среду предприятия для эффективного взаимодействия с существующим функционалом, а также оперативного внесения изменений с учетом технических и технологических требований конкретных горнодобывающих предприятий;
- QA2 – масштабируемость – в целях обеспечения работоспособности программного решения, в том числе в условиях переменной (пониженной или повышенной) производительности в пиковые периоды, с сохранением качества обработки разнородных данных;
- QA3 – безопасность – в целях обеспечения конфиденциальности коммерческих и других секретов горнодобывающих предприятий, целостности и доступности данных, защиты от несанкционированного доступа, ошибочных параметров систем управления, а также самодиагностики и оперативного предупреждения/устранения сбоев;
- QA4 – производительность – с учетом необходимости параллельных вычислений и обработки значительных объемов разнородных данных.

Анализ выявленных атрибутов качества (QA1 – QA4) показывает, что наиболее существенными компромиссными парами для них являются:

1) QA1 ↔ QA2,
2) QA1 ↔ QA4,
3) QA3 ↔ QA4.

В качестве важнейших рекомендаций к высокоуровневому архитектурному проекту программного решения имеет смысл отметить, прежде всего, следующие:

1. В целом архитектура программного решения должна быть сервисно-ориентированной и строиться на принципах SOA (Service-Oriented Architecture). При этом для достижения гибкости, обусловленной пунктами №№ 1–4 базового процесса архитектурно-ориентированного метода, программное решение должно быть разбито на микросервисы. Такое разделение системы на микросервисы призвано обеспечить решение для компромиссов атрибутов качества QA1 и QA2.

2. Для верхнего слоя архитектуры рекомендуется использовать сервисную шину уровня предприятия, построенную на принципах корпоративной шины типа ESB (*Enterprise Service Bus*).

3. На среднем уровне архитектуры (включающем, в частности, подсистемы управления горнодобывающим предприятием) рекомендуется использовать многослойную архитектуру. Подобное решение адекватно поддерживает ключевую многоуровневую систему управления предприятием на основе матрицы корпоративной архитектуры (Enterprise Architecture Matrix, EAM), в которой выделяются, по крайней мере, четыре уровня-«слоя»:

1) Уровень аппаратной интеграции (подсистемы диспетчерского управления типа SCADA, устройства типа IoT);

2) Уровень краткосрочного планирования ресурсов (подсистемы оперативного управления типа MES);

3) Уровень среднесрочного планирования ресурсов (подсистемы корпоративного планирования типа ERP);

4) Уровень стратегического планирования (включая подсистемы поддержки принятия решений типа DSS).

Уровень (1) поддерживает сбор необработанных (аналоговых) данных, а также управление роботами и сенсорами, включая устройства для «интернета вещей» (IoT, *Internet of Things*). Уровень (2) поддерживает оперативное управление, включая складской учет и управление запасами. Уровень (3) поддерживает планирование ресурсов на период 1–3 года. Уровень (4) поддерживает долгосрочное планирование ресурсов и принятие стратегических решений. Каждый из перечисленных уровней агрегирует, консолидирует и перенаправляет данные на последующие, более высокие уровни планирования и поддержки принятия решений. При этом связи между уровнями рекомендуется обеспечивать посредством серии конвейеров и многоагентной среды, которые в совокупности включают (рис. 2) [1, 2, 4] следующее:

- роботизированные мобильные объекты в форме горнотранспортного оборудования с расширенным комплексом бортовых систем, обеспечивающие сбор данных с датчиков для построения вычислительных моделей автономного управления;

- информационную систему предприятия (с базой данных), обеспечивающую доступ к производственной, технологической, горно-геологической и другой статистической и нормативной информации, необходимой для построения прогнозных вычислительных моделей автономного управления мобильными объектами (в т. ч. самосвалами);

- среду визуализации, обеспечивающую диспетчерское управление программной системой, инструменты взаимодействия, а также представление производственной и технологической информации и выполнение технологических операций в дистанционном режиме;

- программные и алгоритмические решения для дистанционного управления роботизированной техникой, а также вспомогательных технологических операций для взаимодействия мобильных объектов (в т. ч. самосвалов) с маломобильными и инфраструктурными объектами предприятия в автономном режиме.

Для поддержки процессов архитектурного проектирования многоуровневого программного обеспечения по приведенной выше схеме рекомендуется в дополнение к методам разработки архитектуры на базе ACDM использовать методы анализа архитектур-

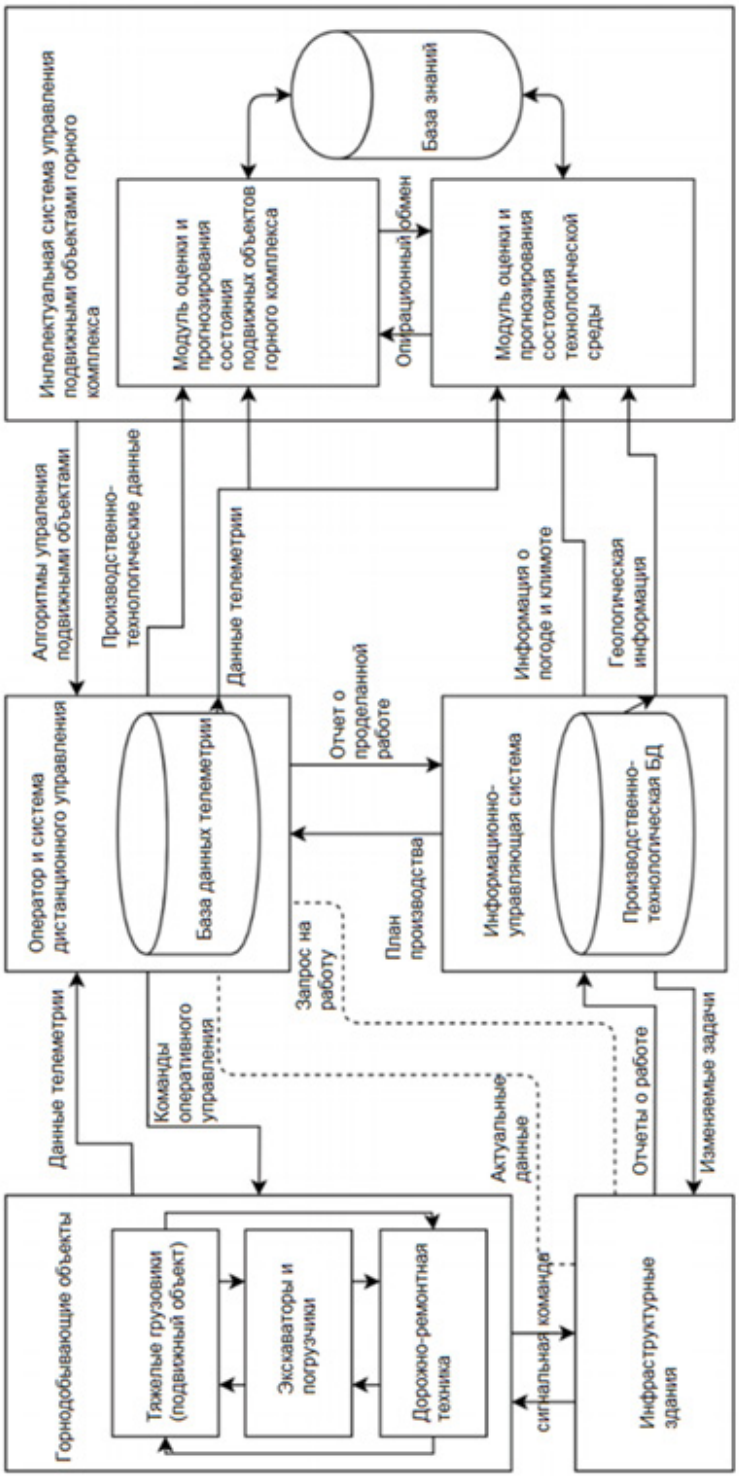

Рис. 2. Обобщенная схема функционирования системы горнодобывающего предприятия.

ных компромиссов (*Architecture Tradeoff Analysis Method*, ATAM) и совещания по ревизии архитектуры (*Architecture Review Board,* ARB).

Оба названных класса методов (ATAM и ARB) являются архитектурно-центрическими и применяются на ранних стадиях разработки, прежде всего, в ходе высокоуровневого проектирования программных решений. При этом требуется уделять особое внимание

оценке атрибутов качества, а также выявлению предпосылок для их применения. Рекомендуется синергия методов ATAM и ARB вместе с ACDM и EAM, что потенциально позволит повысить качество верификации архитектурной схемы программного решения уже на ранних стадиях разработки. Подобный подход существенно снизит критические риски и обеспечит более полное согласование атрибутов качества проектируемого программного решения с ограничениями с точки зрения технологий и бизнес-требований.

Заключение

Предложенный в настоящей работе подход на базе архитектурно-ориентированного метода разработки применим для разработки ответственного программного обеспечения промышленного уровня в различных предметных областях. При этом ввиду высокой сложности процессов проектирования эффективное применение данного подхода требует, как высокого уровня профессиональных знаний, так и методологического обобщения (в том числе за счет интеграции с такими методами как ATAM и ARB). Комплексное использование изложенных принципов и методов обеспечивает потенциальное повышение эффективности при разработке программного обеспечения в рассматриваемых областях.

В настоящей статье разработан высокоуровневый подход к представлению архитектуры высококритичного крупномасштабного программного обеспечения, предназначенного для автономной роботизированной добычи полезных ископаемых открытым способом. В качестве важного момента необходимо отметить возможность более эффективного (с точки зрения адаптивного реагирования) взаимодействия между представителями заказчика и разработчика проектируемого программного обеспечения.

В основу предложенного подхода положен принцип возможно более ранней – в начале фазы проектирования – стабилизации архитектурного представления программного обеспечения в форме семейства компонентов и связей, как это предписывается соответствующим стандартом программной инженерии (SWEBOK).

Для достижения этой цели предлагается использовать комплексный подход на базе архитектурно-ориентированного метода разработки, что позволит обеспечить более оперативное и точное соответствие бизнес-ограничений, функциональных и технических требований.

Разработанный комплексный подход включает разработанные в научных центрах CMU и SEI методы компромиссного анализа и оптимизации (в т. ч. ATAM, ACDM и ARB), а также оригинальный метод на основе матрицы корпоративной архитектуры (EAM). На основе предложенного подхода выявлена, сформирована и проранжирована совокупность критически значимых атрибутов для построения программных решений рассматриваемого класса и масштаба, с учетом особенностей предметной области. В результате разработана высокоуровневая архитектура, представленная в форме компонентов и связей, а также даны рекомендации по архитектурным шаблонам для будущего программного решения.

Благодарности

Работа выполнена при поддержке гранта Российского научного фонда (проект № 19-17-00184).

Литература / References

1. Maranzano J.R., Rozsypal S.A., Zimmerman G.H., Warnken G.W.,Wirth P.E. Architecture reviews: practice and experience. *IEEE Software.* 2005;22(2):34-43.
https://doi.org/10.1109/MS.2005.28

2. Ferber S., Heidl P., Lutz P. Reviewing Product Line Architectures: Experience Report of ATAM in an Automotive Context. In: van der Linden F. (Ed.) Software Product-Family Engineering. PFE 2001. Lecture Notes in Computer Science, V. 2290. Heidelberg Berlin, Heidelberg: Springer; 2002. P. 364-382.
https://doi.org/10.1007/3-540-47833-7_33

3. Harley N. 11 of the most costly software errors in history, 2018. https://raygun.com/blog/costly-software-errors-history/

4. Clements P., Kazman R., Klein M. Evaluating Software Architecture: Methods and Case Studies. Addison-Wesley; 2002. 304 p.

5. Bass L., Clements P., Kazman R. Software Architecture in Practice (2nd Edition). Addison-Wesley; 2003. 560 p.

6. Carnegie Mellon University / Software Engineering Institute. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley; 1995.

7. Humphrey W.S. Introduction to the Team Software Process. Addison-Wesley; 1999. 496 p.

8. Boehm B. Software Engineering Economics. New Jersey: Prentice Hall; 1981. 767 p.

9. Patel J., Lee R.Y., Kim H-K. Architectural View in Software Development Life-Cycle Practices. In: 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007). Melbourne, Australia; 2007.
https://doi.org/10.1109/ICIS.2007.64

10. Norman D.A. The Invisible Computer. Cambridge, MA: MIT Press; 1998. 340 p.

11. Cabrera A.A.A., Komoto H., van Beek T.J., Tomiyama T. Architecture-centric design approach for multidisciplinary product development. In: T. Simpson, J. Jiao, Z. Siddique, K. Hölttä-Otto (Eds.). Advances in Product Family and Product Platform Design. Methods & Applications. New York: Springer; 2014. P. 419-447.
https://doi.org/10.1007/978-1-4614-7937-6_17

12. Lattanze A.J. Architecting software intensive systems: a practitioner's guide. CRC Press; 2008. 416 p.

13. Zykov S.V., Singh A. Agile Enterprise Engineering: Smart Application of Human Factors. Models, Methods, Practices, Case Studies. Springer Nature Switzerland AG; 2020. 158 p.
https:// doi.org/10.1007/978-3-030-40989-0

Об авторах:

Головин Сергей Анатольевич, доктор технических наук, профессор, заведующий кафедрой математического обеспечения и стандартизации информационных технологий Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

Зыков Сергей Викторович, доктор технических наук, доцент, профессор Департамента программной инженерии факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» (101000, Россия, Москва, ул. Мясницкая, д. 20).

Кораблин Юрий Прокофьевич, доктор технических наук, профессор, профессор кафедры математического обеспечения и стандартизации информационных технологий Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

Крюков Дмитрий Алексеевич, кандидат технических наук, доцент кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

About the authors:

Sergey A. Golovin, D.Sci. (Engineering), Professor, Head of the Department of Mathematical Support and Standardization of Information Technologies, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo Pr., Moscow, 119454, Russia).

Sergey V. Zykov, D.Sci. (Engineering), Associate Professor, Professor of School of Software Engineering of Faculty of Computer Science, Higher School of Economics, National Research University (20, Myasnitskaya Str., Moscow, 101000, Russia)

Yurii P. Korablin, D.Sci. (Engineering), Professor, Professor of the Department of Mathematical Support and Standardization of Information Technologies, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo Pr., Moscow, 119454, Russia).

Dmitry A. Kryukov, Cand. Sci. (Engineering), Associate Professor, Associate Professor of the Department of ERP, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo Pr., Moscow, 119454, Russia).

Поступила: 23.01.2020; получена после доработки: 02.07.2020; принята к опубликованию: 06.07.2020.