Повышение качества программного обеспечения посредством разработки на основе компонентно-ориентированного подхода: новая стратегия

В.Ч. Алисауи, И.М. Ал-Хафаджи, Х.А. Джураев , А. В. Панов

***Аннотация*— Разработка программного обеспечения на основе компонентов является эффективным средством для повышения качества программных систем.**

В этой статье рассматривается, как разработка на основе компонентов может повысить качество программного обеспечения, и какие преимущества предполагает этот подход. Также исследуются некоторые из инструментов и технологий, которые могут помочь в реализации этой стратегии.

При ее реализации существуют такие технологии, как компонентные фреймворки, библиотеки компонентов и управление зависимостями. Эти инструменты помогают управлять и устанавливать компоненты, а также обеспечивают легкое их использование в разработке программного обеспечения.

Разработка программного обеспечения на основе компонентов (Component-Based Software Development, CBSD)" - это подход, в котором приложения создаются из готовых компонентов. Эти компоненты - независимые единицы кода, которые могут быть использованы в разных приложениях. Таким образом, CBSD помогает ускорить и упростить разработку программного обеспечения, улучшить его качество и упростить поддержку и обновление.

В CBSD компоненты могут быть использованы многократно в различных приложениях, что позволяет избежать дублирования кода и улучшить эффективность разработки. Кроме того, использование готовых компонентов уменьшает риск ошибок и улучшает надежность приложения. В CBSD также можно использовать компоненты, реализованными другими разработчиками, что снижает время и ресурсы, необходимые для разработки функциональности от нуля.

Ключевые слова—Компонентный подход, Модель качества, Атрибуты качества, А-модель для CBD

I. Введение

Компонентно-ориентированная разработка «Component Based Development (CBD)» становится все более важной для индустрии программного обеспечения, по сравнению с другими моделями.

Разработка на основе компонентов — это деятельность CBSE, которая происходит параллельно с инженерной парадигмой. Использование первых трех этапов, таких как анализ и архитектурный дизайн, непосредственно коррелируется с методами проектирования. Команда разработчиков программного обеспечения уточняет архитектурный стиль, соответствующий подходу к анализу, созданному для конкретного приложения [2].

Предполагается, что CBD уменьшит стоимость и время развертывания программных приложений при одновременном повышении их качества. Поскольку компоненты используются повторно, они, вероятно, будут более надежными, чем программное обеспечение, разработанное с нуля, поскольку они были протестированы в более широком диапазоне условий.

Экономия затрат и времени является результатом усилий, которые в противном случае были бы необходимы для разработки и интеграции функциональных возможностей, предоставляемых компонентами в каждом новом программном приложении. В этом исследовании описывается и оценивается качество программных компонентов.

При использовании А-модели эта оценка должна выполняться с использованием модели качества компонентов [6]. «Разработка и проверка модели, которая отвечает на эти вопросы, предлагает очень полезный инструмент оценки для клиентов, которые нуждаются в компонентах для включения в свои программные приложения». Они могли бы использовать здесь следующее определение компонента: программный компонент — это независимо развертываемая реализация некоторой функциональности, которая может быть повторно использована в широком спектре некоторых приложений [1].

II. Теоретические основы

A. Модель качества для разработки на основе компонентов: характеристики качества

В исследовании качества программного обеспечения есть два основных направления: одно касается оценки качества процесса разработки программного обеспечения, а другое — качества самого продукта. В литературе можно найти несколько моделей качества продукта, например [3,4,5,6,7,8,9].

Среди этих моделей наиболее согласованной является модель, предлагаемая ISO 9126. Эти модели включают

общие атрибуты качества программного обеспечения. Кроме того, они были разработаны на уровне системы, а не на уровне компонентов. Хотя некоторые из их характеристик подходят для оценки программных компонентов, другие не подходят для этой задачи. Некоторые из их качественных характеристик, таких как отказоустойчивость, обычно оцениваются на системном уровне, а не для каждого из компонентов. Модель качества для компонентов должна быть адаптирована для использования только тех характеристик, которые относятся к компонентам.

В [10] такая модель предложена для оценки компонентов «COTS (Commercial Off The Shelf)». COTS является одним из программных ресурсов, которые полностью проверены и имеют относительно низкий риск [2].

Модель, предложенная в этой статье, имеет широкий спектр применения, поскольку она предназначена для последующей интеграции программных компонентов в генеральный продукт. В качестве отправной точки они будут использовать модель качества, описанную в [9] (см. Таблицу 1), которая является попыткой исправить некоторые недостатки модели ISO9126. В статье производится попытка адаптировать его к особым потребностям качественных компонентов [4].

Первые больше подходят для оценки качества компонентов с точки зрения разработчиков компонентов, в то время как вторые представляют особый интерес для потребителей компонентов (или разработчиков, использующих компоненты, которые уже существуют при построении своих приложений). Ниже (Таблица 1) приведены некоторые важные характеристики качества, положенные в основу модели.

Таб.1. Атрибуты качества

Функции	Части отношения
Ремонтопригодность (Обслуживаемость)	Диагностируемость, ремонтопригодность, расширяемость, тестируемость, соответствие
Надежность	Надежность, Безопасность, Сохранность, Соответствие требованиям
Юзабилити	Обучаемость, Конфигурируемость, Работоспособность
Эффективность	Поведение во времени, обеспеченность ресурсами, соответствие требованиям

B. Критерии разработки программного обеспечения на основе компонентов

Предлагаются критерии разработки программного обеспечения на основе компонентов, которые имеют пять атрибутов с акцентом на стиль нового дизайна разработки, и подробно описываются все эти функции (Рис.1)

Рис.1. Критерии разработки компонентного программного обеспечения

- ***Продуктивность программного обеспечения***

определяется путем подсчета количества произведенных единиц и деления этого на количество человеко-часов, необходимых для их производства. Однако для любой программной проблемы существует множество различных решений, каждое из которых имеет разные атрибуты [6].

Одно решение может выполняться более эффективно, другое - менее. Когда производятся сопоставления продуктов с разными атрибутами, сравнение их производительности не имеет особого смысла. Тем не менее, руководитель проекта может столкнуться с проблемой оценки продуктивности программистов [11].

Эти оценки производительности могут понадобиться вам для определения стоимости или графика проекта, для обоснования инвестиционных решений или для оценки того, эффективны ли улучшения процесса или технологии. Оценки производительности обычно основаны на измерении атрибутов программного обеспечения и делении их на общие усилия, необходимые для разработки.

Существует два типа метрик, которые использовались, например, метрики, связанные с размером, метрики, связанные с функциями, и количество строк исходного кода на человеко-месяц (LOC/pm или SLOC/pm) — это широко используемый показатель производительности программного обеспечения [13].

- ***Масштабируемость программного обеспечения***

Система, бизнес или программное обеспечение, которое описывается как масштабируемое, имеет преимущество, поскольку оно лучше адаптируется к изменяющимся потребностям или требованиям своих пользователей или клиентов [10]. Масштабируемость часто является признаком стабильности и конкурентоспособности, так как это означает, что сеть, система, программное обеспечение или организация готовы справиться с притоком спроса, повышением производительности, тенденциями, меняющимися потребностями и даже присутствием или появлением новых конкурентов [8]. Стандарт применим во многих отраслях, включая здравоохранение, фармацевтику, электронику, он описывает подход к проектированию, классификации и эксплуатации информационных систем [12].

- ***Преимущества масштабируемого программного обеспечения***

Масштабируемость имеет как долгосрочные, так и краткосрочные преимущества. Вначале это позволяет компании приобретать только то, что ей необходимо немедленно, а не все функции, которые могут быть полезны в будущем [2]. Например, компания, запускающая пилотную программу анализа данных, может выбрать массивный пакет корпоративной аналитики или начать с решения, которое поначалу выполняет только те функции, которые им нужны. Популярным выбором является информационная панель, которая извлекает результаты из своих основных источников данных и существующего корпоративного

программного обеспечения [7]. Когда они станут достаточно большими, чтобы использовать больше аналитических программ, эти потоки данных можно будет добавить на панель инструментов, вместо того чтобы заставлять компанию жонглировать несколькими программами визуализации или создавать совершенно новую систему. Подобный подход подготавливает к будущему росту, создавая более экономичный продукт, который соответствует текущим потребностям без дополнительной сложности [9]. Это также требует меньших первоначальных финансовых затрат, что является важным соображением для руководителей, обеспокоенных размером инвестиций в большие данные. Масштабируемость также оставляет место для изменения приоритетов. Этот готовый пакет аналитических данных может потерять актуальность, поскольку компания адаптируется к требованиям меняющегося рынка.

- ***Качество программного обеспечения***

Качество программного обеспечения предполагает, что последнее не содержит критических ошибок или дефектов, поставляется вовремя и в рамках бюджета, соответствует требованиям и/или ожиданиям и может поддерживаться. Стандарт ISO 8402-1986 определяет качество как совокупность свойств и характеристик продукта или услуги, которые несут в себе способность удовлетворять заявленные или подразумеваемые потребности [14].

К ключевым аспектам качества для потребителя относятся:

1) Хороший дизайн – внешний вид и стиль
2) Хорошая функциональность – хорошо справляется со своей задачей
3) Надежный – допустимый уровень поломок или отказов
4) Последовательность

а) Надежность программного обеспечения

Программное обеспечение должно быть доступно, когда это необходимо, а также должно работать правильно, безопасно и надежно, без каких-либо неблагоприятных побочных эффектов или проблем с безопасностью [4].

Крайне важно, чтобы программное обеспечение, используемое в системах в критических областях безопасности, было надежным, поскольку последствием отказа (например, отказа атомной электростанции) может быть огромный ущерб, ведущий к гибели людей или к угрозе их жизни. Инженерия надежности связана с методами повышения надежности систем и включает в себя использование строгого процесса проектирования и разработки для сведения к минимуму количества дефектов в программном обеспечении [10].

Надежная система, как правило, предназначена для обеспечения отказоустойчивости, когда система может справляться с ошибками (и восстанавливаться после них), возникающими во время выполнения программного обеспечения. Такая система должна быть безопасной и способной защитить себя от случайных или преднамеренных внешних атак. В таблице 2 перечислены несколько параметров надежности. Современные программные системы подвержены атакам вредоносных программ, таких как вирусы, которые изменяют поведение программного обеспечения или повреждают данные, что делает систему ненадежной [6].

Существует компромисс между надежностью и производительностью системы, поскольку надежные системы должны будут выполнять дополнительные проверки, чтобы контролировать себя и проверять ошибочные состояния, а также восстанавливаться после сбоев до того, как произойдет сбой [11].

Это неизбежно приводит к увеличению затрат на проектирование и разработку надежных систем.

Таб.2. Измерение надежности

Измерение	Описание
Доступность	Система доступна для использования в любое время
Надежность	Система работает правильно и заслуживает доверия
Сохранность	Система безопасна и не наносит ущерба окружающей среде
Безопасность	Система предотвращает несанкционированные вторжения

Доступность программного обеспечения — это процент времени, в течение которого работает система программного обеспечения или показатель времени безотказной работы / времени простоя программного обеспечения в течение определенного периода времени. Время простоя относится к периоду времени, когда программное обеспечение недоступно для использования (включая плановые и незапланированные отключения), и многие компании стремятся разработать программное обеспечение, которое будет доступно для использования 99,999 % времени в году (т. е. время простоя менее 5 мин в год) [7].

Эта цель известна как пять девяток, и это общая цель в телекоммуникационном секторе. Критические, с точки зрения безопасности, системы — это системы, в которых важно, чтобы система была безопасна для населения и чтобы люди или окружающая среда не пострадали в случае отказа системы [9].

К ним относятся системы управления самолетами и системы управления технологическими процессами для химических и атомных электростанций. Безопасность системы относится к ее способности защищать себя от случайных или преднамеренных внешних атак, которые сегодня распространены, поскольку большинство компьютеров объединены в сеть и подключены к Интернету [11].

Шифрование — это один из способов снизить уязвимость системы, поскольку зашифрованные данные не могут быть прочитаны злоумышленником. Могут быть элементы управления, которые обнаруживают и отражают атаки, и эти элементы управления используются для мониторинга системы и принятия мер

по отключению частей системы или ограничению доступа в случае атаки [13].

Важно обеспечить разумный уровень безопасности, поскольку в противном случае все другие аспекты надежности (доступность и безопасность) будут поставлены под угрозу. При разработке системы могут быть введены лазейки в системе безопасности, поэтому необходимо соблюдать осторожность, чтобы предотвратить использование хакерами уязвимостей безопасности [15].

- ***Сопровождение программного обеспечения***

Данный аспект означает исправление, обновление, обслуживание и модификацию системы или обновление программного обеспечения для повышения производительности или исправления ошибок. Ремонтопригодность также включает в себя добавление новых функциональных возможностей или адаптацию программного обеспечения, с учетом новых требований под нужды заказчика. Ремонтопригодность программного обеспечения — это степень возможности его ремонта или улучшения [2].

В течение жизненного цикла разработки системы «System Development Life Cycle (SDLC)» данный этап требует больше усилий по разработке, чем любой другой. Примерно 75 % затрат приходится на обслуживание программного обеспечения. Ремонтопригодность повышает надежность, эффективность или безопасность программного обеспечения, а также используется для облегчения будущего обслуживания и для увеличения срока службы программного обеспечения. Реализуются также ремонт или замена неисправных компонентов и улучшение программного обеспечения, по сравнению с предыдущим состоянием программного обеспечения [8].

Сопровождение программного обеспечения требуется, когда заказчик требует реализации новых функций в программном обеспечении. Иногда требуется техническое обслуживание, когда меняется аппаратное обеспечение системы, тогда требуется модификация программного обеспечения. Рыночные условия и организационные изменения также являются причинами модификации программного обеспечения. Это также включает в себя то, что при обнаружении проблемы немедленно исправьте ее, прежде чем она станет большой проблемой [10].

Иногда в программном обеспечении обнаруживаются вирусы и вредоносные программы, которые вызывают проблемы у пользователя, и для их устранения или повышения производительности требуется техническое обслуживание программного обеспечения. Идея, лежащая в основе атрибутов ремонтопригодности, заключается в изменении требований в соответствии с потребностями клиента. Как показано ниже (рис. 2), для представления атрибутов ремонтопригодности для CBD выделяют следующие группы пользователей. Первая группа — пользователи, вторая — разработчики по, третья — владельцы ИТ-компаний. Таким образом, люди начинают вступать в различные формализованные и неформализованные группы по интересам, такие как: комьюнити, хакатон, коворкинг, хакерспейс, сети, ИТ-кластер и другие [17].

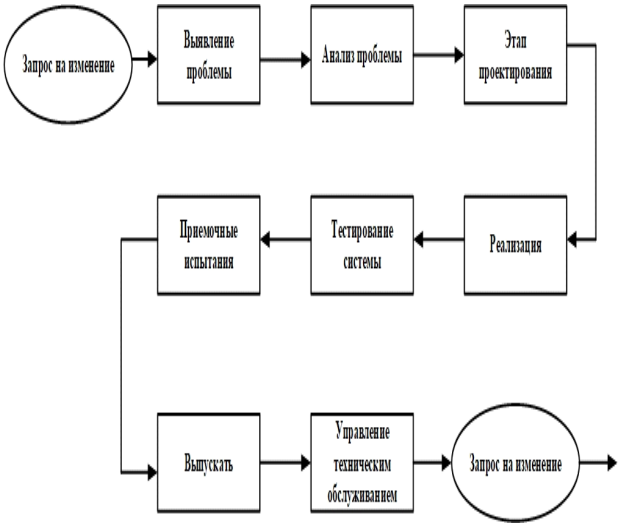

Рис.2. Атрибуты ремонтопригодности для CBD

III. Методология исследования

A. А-модель для разработки на основе компонентов

Предлагается новая модель подхода к разработке на основе компонентов для реализации возможности повторного использования, которая обеспечивает снижение затрат и сокращение времени при сохранении высокого уровня качества. А-модели для CBD предлагается повторно использовать с процедурами, с учетом нового подхода к разработке с различными этапами новой модели.

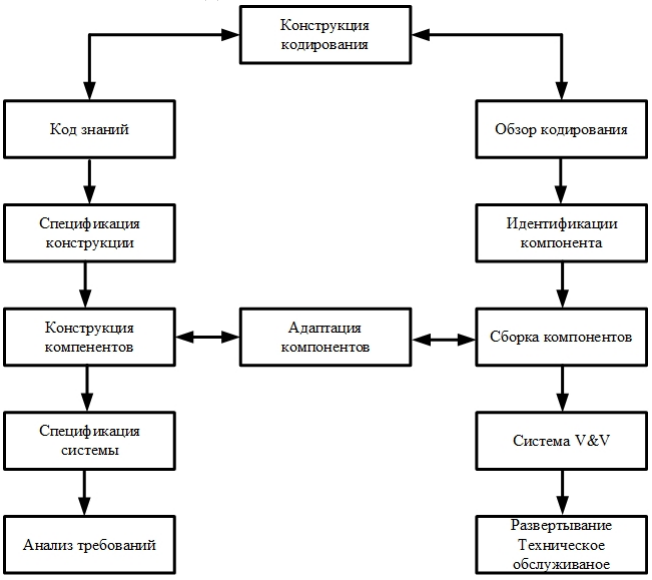

Рис.3. А-Модель для CBD

B. Анализ требований

Процесс определения ожиданий пользователей от приложения, которое должно быть создано или изменено. Анализ требований включает в себя все задачи, которые проводятся для выявления

потребностей различных заинтересованных сторон [1].

Следовательно, анализ требований означает анализ, документирование, проверку и управление требованиями к программному обеспечению или системе. Требования высокого качества документированы, применимы к действиям, измеримы, проверяемы, отслеживаемы, помогают определить возможности для бизнеса и определены для облегчения проектирования системы [3].

C. Дизайн компонентов

Архитектура продукта определяет набор адаптируемых компонентов, которые можно использовать для реализации семейства рабочих продуктов. Проект компонента — это спецификация проекта для одного из этих адаптируемых компонентов [9].

Разработчики приложений, используя процедуру генерации, могут адаптировать и составить набор этих компонентов для реализации определенных рабочих продуктов или частей [11]. Каждый компонент должен быть разработан с учетом соответствующих аспектов требований к продукту и всех проектных структур архитектуры продукта.

D. Технические характеристики конструкции

Описывается, как система выполняет требования, изложенные в функциональных требованиях [5].

В зависимости от системы, это может включать инструкции по тестированию конкретных требований, параметров конфигурации или просмотру функций или кода [12].

Должны быть учтены все требования, изложенные в функциональной спецификации; осуществляется связывание требований между функциональными требованиями и проектной спецификацией. Примеры спецификации проекта.

Спецификации проекта могут включать:

1) Конкретные входные данные, включая типы данных, которые необходимо ввести в систему;
2) Расчеты, используемые для выполнения определенных требований;
3) Выходные данные, генерируемые системой
4) Объяснение технических мер по обеспечению безопасности системы;
5) Определение, насколько система соответствует минимальным нормативным требованиям.

E. Код знаний

Программист начинает с того, что выбирает инструменты, которые позволяют конструировать код, определяя подходящий язык программирования с операционными системами и избирая важные моменты с различными представлениями о форме программного обеспечения [8]. Роль консультанта, имеющего опыт разработки программного обеспечения, позволяет представить решение проблемы [14]. Характеристики различных языков программирования зависят от типа проблемы и способа поиска решения.

F. Конструкция кодирования

На данном этапе «начинается настоящая работа» и мы «строим то, что нужно».

1) Разработчики начинают писать код в соответствии с требованиями и разработанным дизайном;
2) Вместе с кодированием начнутся все другие необходимые настройки, то есть база данных, настроенная администратором базы данных, а также создание интерфейса и графического интерфейса разработчиками внешнего интерфейса и т. д.;
3) Наряду с кодированием, для разработчиков также важно разрабатывать модульные тесты для своего модуля, рецензировать модульные тесты других модулей, развертывать сборки в предполагаемой среде и выполнять модульные тесты.

G. Обзор кодирования

Проверка кода — это этап процесса разработки программного обеспечения, на котором авторы кода, рецензенты и, возможно, тестировщики обеспечения качества «Quality Assurance (QA)» собираются вместе для проверки кода [3].

Поиск и исправление ошибок на этом этапе являются относительно недорогими и имеют тенденцию сокращать более дорогостоящий процесс обработки, обнаружения и исправления ошибок на более поздних этапах разработки или после того, как программы доставлены пользователям [16].

Рецензенты читают код построчно, чтобы проверить:

1) Недостатки или потенциальные недостатки;
2) Согласованность с общим дизайном программы;
3) Качество комментариев;
4) Соблюдение стандартов кодирования.

Проверка кода может быть особенно продуктивной для выявления уязвимостей в системе безопасности. Доступны специализированные прикладные программы, которые могут помочь в этом процессе. Автоматизированная проверка кода облегчает систематическое тестирование исходного кода на наличие потенциальных проблем, таких как переполнение буфера, переполнение памяти, нарушение размера и повторяющиеся операторы [4].

H. Идентификация компонентов

Идентификация компонентов представляет собой одну из самых важных и сложных задач при разработке компонентных систем. Существующие подходы к разработке компонентов не имеют единого стандарта для идентификации компонентов и зависят от опыта разработчиков. Разработчик предполагает идентифицировать компоненты внутри библиотеки для повторного использования, а не разрабатывать с нуля [3].

I. Сборка компонентов

Компоненты могут быть интегрированы через некоторую форму общей инфраструктуры. Для согласованной сборки компонентов инфраструктура должна состоять как из физической инфраструктуры

связи (база данных или инфраструктура обмена сообщениями), так и из набора концептуальных соглашений, таких как соглашения об именах, которые воплощают общую семантику между компонентами [8].

Эта инфраструктура будет поддерживать сборку компонентов и координацию, а также отличать запланированную, скоординированную сборку компонентов от объединения другого характера. Ряд отраслевых стандартов для компонентных инфраструктур был разработан для облегчения связи между различными реализациями инфраструктуры [2].

J. Адаптация компонентов

Адаптация компонентов подразумевает, что внесены поправки для устранения потенциальных источников конфликта между компонентами, которые должны быть собраны для формирования прикладной системы. Обычно простые скрипты пишутся как буфер между запросами пользователя и действиями компонентов. Буфер можно использовать для предоставления компонентам информации по умолчанию, исключения доступа к нежелательному поведению компонентов, слоя изоляции для замены компонентов. На рисунке подразумевается своего рода «накрутка» компонента, но возможны и другие подходы (например, использование агентов, интеллектуальных посредников, трансляторов) [11].

Однако некоторые авторы предлагают разделить системы на две категории: технические компоненты (статистический контроль процессов, использование бенчмаркинга или внедрение гибких процессов) и социальные или контекстуальные компоненты (лидерство, участие персонала, ориентация на клиента, обучение и развитие навыков персонала, работа в команде, общение, общее видение) [4].

На риски проекта влияют такие факторы, как уровень новизны для организации, сложность проекта, его продолжительность, наличие ресурсов, в том числе высококвалифицированных специалистов и др. [16].

K. Проверка системы

Можно столкнуться со всевозможными вариантами использования и интерпретации этих терминов, и наша скромная попытка здесь как можно четче провести различие между ними [12].

Верификация — это процесс оценки рабочих продуктов определенной фазы разработки (а не фактического конечного продукта), чтобы определить, соответствуют ли они указанным требованиям для этой фазы.

Цель верификации — убедиться, что продукт создается в соответствии с требованиями и проектными спецификациями. Другими словами, чтобы рабочие продукты соответствовали заданным требованиям [10].

Валидация — это процесс оценки программного обеспечения во время или в конце процесса разработки, чтобы определить, удовлетворяет ли оно заданным бизнес-требованиям. Цель состоит в том, чтобы гарантировать, что продукт действительно соответствует потребностям пользователя и что спецификации изначально были правильными [6].

L. Развертывание и обслуживание

Фаза развертывания системы эволюционировала, выпуская продукт для клиента, другими словами, программное обеспечение становится доступным для использования клиентом. Развертывание системы должно осуществляться с использованием какого-либо специализированного инструмента, чтобы упростить развертывание для клиента. Различные версии выпуска системы должны поддерживаться и обрабатываться должным образом. Если в системе есть обновление, оно должно поддерживаться в метаданных и репозитории [10].

Реконфигурация, адаптация, переустановка установленной системы должны быть решены должным образом, чтобы избежать проблем во время выполнения, обычно возникающих при установке системы. Модернизация и замена компонентов – основная работа по обслуживанию системы. Обновление системы обычно происходит, когда поставщик COTS выпускает новую версию компонента или когда новый компонент COTS устаревает. Внесение изменений в код требуется при обслуживании системы. Подход к удовлетворению потребностей клиентов является базовой основой стратегии компании с момента ее создания. Все структурировано вокруг своих аспектов с центральной целью, которая состоит в том, чтобы завоевать новые отрасли и сохранить существующие [13].

Таким образом, внедрение стандартизации также сопровождалось усилением систем планирования и инструментов контроля. Управление процессом качества организовано за счет его обобщения на все процессы и обеспечения логики измерения через разнообразные показатели. Создание акционерной стоимости больше не является единственным показателем эффективности [5].

IV. Результаты и обсуждение

Итак, рассматривается новая модель под названием A-Model, которая описывает атрибуты повторного использования, снижая стоимость разработки программного обеспечения и время, сохраняя качество программного обеспечения и повышая производительность за счет использования специальных инструментов и новых методов разработки. Мы должны предложить обновить систему в соответствии с любыми новыми требованиями с адаптацией некоторых атрибутов внутри модели для получения подходящих результатов при продолжении работы над атрибутами качества.

V. Заключение

Предлагается новая модель сертификации программных компонентов — это процедура с новым подходом. В качестве долгосрочной цели можно стремиться внести свой вклад в разработку процесса сертификации, соответствующего компонентам

программного обеспечения. Первым шагом на пути к этой цели является определение модели качества, которую можно использовать в качестве основы в компонентных процедурах.

Библиография

[1] А. И. Хан, «Улучшенная модель компонентной разработки программного обеспечения», Научно-академическое издательство, 2012.B. Meyer, “Object-Oriented Software Construction”, *2nd ed. NJ, USA: Prentice Hall PTR*, 1997.

[2] Б. В. Бём, Б. К. Кларк, Э. Горовиц, Р. Мадучи, Р. Селби и К. Вестленд, «Обзор модели стоимости программного обеспечения COCOMO 2.0», 1995 г.

[3] К.Э.Мохлис, А. Эльмортада, С.Марване, «Диагностика организационных изменений: многоуровневый подход PEN Vol. 7, № 3, сентябрь 2019 г., стр. 1177–1185 1185 (Пример французского МСП, сертифицированного по стандарту ISO 9001))», Периодические издания по инженерным и естественным наукам, Том 7, № 2, август 2019 г., стр. 932-943.

[4] Ж. О’Реган, «Краткое руководство по формальным методам», Темы бакалавриата по компьютерным наукам, Springer International Publishing AG, 2017.

[5] Ж. МакКолл, «Факторы качества», в Энциклопедии программной инженерии, том. I+II, Ж. Ж. Марчиняк, Ред.: John Wiley & Sons, 1994, стр. 958.

[6] Л. Басс, К. Буман, С. Комелла-Дорда, Ф. Лонг, Ж. Роберт, Р. Сикорд и К. Валлнау, «Том I: Оценка рынка компонентной разработки программного обеспечения», Институт программной инженерии, Технический институт. Примечание CMU/SEI-2001-TN-007, май 2001 г.

[7] М. Бертоа и А. Валлесильо, «Атрибуты качества для компонентов COTS», представленные на 6-м международном семинаре по количественным подходам в объектно-ориентированной разработке программного обеспечения (QAOOSE'2002), Малага, Испания, 2002 г.

[8] Р.С.Прессман, книга «Программная инженерия», 5-е издание, McGraw-Hill, 2001 г.

[9] V. Honcharenko, A. Panteleimonenko, A. Pozhar, V. Stetsenko, "Cooperatives in IT sector: theoretical and practical aspects", *Periodicals of Engineering and Natural Sciences*, Vol.7, No.2, August 2019, pp.597-607.

[10] Ламхарбах Ясин, Бази Фаталлах, Хаджи Латифа, «Исследование влияния модели статистики на использование и соответствие результатов новой стандартной версии ISO 14644, часть 1», Периодические издания по инженерным и естественным наукам, Vol. T. 6, № 2, декабрь 2018 г., стр. 436-446.

[11] Л. Басс, К. Буман, С. Комелла-Дорда, Ф. Лонг, Дж. Роберт, Р. Сикорд и К. Валлнау, «Том I: Оценка рынка компонентной разработки программного обеспечения», Институт программной инженерии, Технический институт. Примечание CMU/SEI-2001-TN-007, май 2001 г.

[12] Л. Ж. Артур, «Измерение производительности программистов и качества программного обеспечения»: Wiley-Interscience, 1985 г.

[13] М. Бертоа и А. Валлесильо, «Атрибуты качества для компонентов COTS», представленные на 6-м международном семинаре по количественным подходам в объектно-ориентированной разработке программного обеспечения (QAOOSE'2002), Малага, Испания, 2002 г.

[14] Р. Б. Грейди и Д. Л. Касвелл, «Метрики программного обеспечения: создание общекорпоративной программы». Энглвуд Клиффс, Нью-Джерси, EUA: Прентис-Холл, 1987 г.

[15] Р.С. Прессман, книга «Программная инженерия», 5-е издание, McGraw-Hill, 2001.

[16] Виталина Бабенко, Людмила Ломовских, Альвина Орехова, Любовь Корчинская, «Особенности методов и моделей в управлении рисками ИТ-проектов», Журналы инженерных и естественных наук, Vol. T. 7, № 2, август 2019 г., стр. 629-636.

[17] В. Гончаренко, А. Пантелеймоненко, А. Пожар, В. Стеценко, «Кооперативы в сфере ИТ: теоретические и практические аспекты», Журналы технических и естественных наук, Том 7, №2, август 2019 г., стр.597-607.

Статья получена 08 Сентября 2023 г.

Алисауи Висам Чаяд, *аспирант. техн. наук, кфд. информатика и вычислительная техника, институт информационных технологий, РТУ МИРЭА-Российский Технологический Университет. Москва, Россия, Пр. Вернадского, д. 78, +7 (916) 085-61-92, E-mail: wisam.chyad@qu.edu.iq, ORCID: https://orcid.org/0000-0003-4581-8917.*

Ал-Хафаджи Исра М. Абдаламир, *аспирант, техн. наук, кфд. информатика и вычислительная техника, институт информационных технологий, РТУ МИРЭА-Российский Технологический Университет. Москва, РФ, Пр. Вернадского, д. 78, +7 (499) 215-65-65, E-mail: Misnew6@gmail.com.*

Джураев Халимжон Акбарович, *аспирант, техн. наук. кфд. корпоративных информационных систем, институт информационных технологий, РТУ МИРЭА-Российский Технологический Университет. Москва, РФ, Пр. Вернадского, д. 78, +7 (499) 215-65-65, E-mail: djuraevx@mail.ru.*

Панов Александр Владимирович, *канд. техн. наук. доцент. кфд. информатика и вычислительная техника, институт информационных технологий, РТУ МИРЭА-Российский Технологический Университет. Москва, РФ, Пр. Вернадского, д. 78, +7 (499) 215-65-65, E-mail: Iks.ital@yandex.ru.*

Improving Software Quality Through Component-Based Development: A New Strategy

Wisam Ch. Alisawi, Israa M. AL-Khafaji, K. A. Djuraev, A. V. Panov

***Abstract*— Software operations and their development nowadays are considered one of the most important challenges we face, which the software engineer develops and creates the most important tools that help developers as well as designers of these programs. One of the things that we must prioritize is how to improve the methods that produce software tools for us in a good way, through high quality and lower cost, and follow the methods that achieve the goal, which is based on several methods. One such practice is development of software using «Component-Based Software Development (CBSD)» [1,3]. This method recommended building reusable software systems to take advantage of previous experiences as well as use good features instead of starting from scratch. Where most of the characteristics and features of the CBSD method are a good gateway to enter the software industry as well as provide a good model for software development. The steps to develop programs go through many stages. In this research paper, we have taken development steps based on a proposed model that we called the A-model. Where we have provided a reduction in the many steps that lead to the loss of long times for delivery, as well as the costs involved while ensuring that quality is maintained.**

***Keyword* — Component Based Approach, Quality Model, Quality Attributes, A-Model for CBD.**

REFERENCES

[1] A. I. Khan, "An Improved Model for Component Based Software Development", *Scientific & Academic Publishing*, 2012.

[2] B. Meyer, “Object-Oriented Software Construction”, 2nd ed. NJ, USA: *Prentice Hall PTR*, 1997.

[3] B. W. Boehm, B. K. Clark, E. Horowitz, R. Maduchy, R. Selby, and C. Westland, “An overview of the COCOMO 2.0 software cost model”, 1995.

[4] C. E. Mokhlis, A. Elmortada, M. Sbihi, K. Mokhlis, "The impact of ISO 9001 Quality Management on organizational learning and innovation: Proposal for a conceptual framework", *Periodicals of Engineering and Natural Sciences*, Vol.7, No.2, August 2019, pp.944-951.

[5] C. E. Mokhlis, A. Elmortada, S.Marwane, "Diagnosis of Organizational Change: A multi-level approach PEN Vol. 7, No. 3, September 2019, pp.1177- 1185 1185 (Case study of a French SME certified ISO 9001)", *Periodicals of Engineering and Natural Sciences*, Vol.7, No.2, August 2019, pp.932-943.

[6] F. B. Abreu, “Comparing Software Quality Models,”: *INESC Technical Report (in Portuguese)*, 2001.

[7] G. O’Regan, “Concise Guide to Formal Methods”, *Undergraduate Topics in Computer Science*, Springer International Publishing AG 2017.

[8] ISO 9126, “Information Technology - Software Product Evaluation – Software Quality Characteristics and Metrics,”. *Geneva, Switzerland: International Organization for Standardization.*

[9] J. Mc. Call, “Quality Factors,” *in Encyclopedia of Software Engineering*, vol. I+II, J.J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 958-ff.

[10] Lamkharbach Yassine, Bazi Fathallah, Haji Latifa, "Study of changing statistics model’s influence on the exploitation and conformity of results in the new standard version ISO 14644 part 1", *Periodicals of Engineering and Natural Sciences*, Vol. 6, No. 2, December 2018, pp.436-446.

[11] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K.Wallnau, “Volume I: Market Assessment of Component-Based Software Engineering,” *Software Engineering Institute*, Technical Note CMU/SEI-2001-TN-007, May, 2001.

[12] L. J. Arthur, “Measuring Programmer Productivity and Software Quality”: *Wiley-Interscience*, 1985.

[13] M. Bertoa and A. Vallecillo, “Quality Attributes for COTS Components”, *presented at 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002), Malaga*, Spain, 2002.

[14] R. B. Grady and D. L. Caswell, Software Metrics: Establishing a Company-Wide Program. Englewood Cliffs, NJ, EUA: Prentice-Hall, 1987.

[15] R. S. Pressman, “Software Engineering" book, 5th Edition, McGraw-Hill, 2001.

[16] Vitalina Babenko, Liudmila Lomovskykh, Alvina Oriekhova, Liubov Korchynska, "Features of methods and models in risk management of IT projects", *Periodicals of Engineering and Natural Sciences*, Vol. 7, No. 2, August 2019, pp.629-636.

[17] V. Honcharenko, A. Panteleimonenko, A. Pozhar, V. Stetsenko, "Cooperatives in IT sector: theoretical and practical aspects", *Periodicals of Engineering and Natural Sciences*, Vol.7, No.2, August 2019, pp.597-607.