E. L. Kalishenko, Yu. V. Romanenko

TECHNOLOGICAL ASPECTS OF REALIZATION OF COMPUTER TRAINING SYSTEMS WITH THREE-DIMENSIONAL GRAPHIC IN QNX ENVIRONMENT

Article covers technological aspects of 3D-applications development as part of computer-based training systems. Graphical applications optimization techniques are considered for QNX realtime operating system.

Computer-based training, simulators, 3D-applications, QNX operating system, graphical applications optimization

УДК 004.41

С. А. Романенко, А. С. Скрипникова

Управление процессом разработки на основании требований

Рассматривается процесс разработки программных изделий, применяемый в открытом акционерном обществе «Научно-инженерный центр Санкт-Петербургского электротехнического университета». Обсуждаются особенности подхода, его достоинства и недостатки, сложности, возникавшие в процессе его внедрения и применения. Подчеркивается положительное влияние выбранного подхода на процесс взаимодействия заинтересованных лиц на всех этапах жизненного цикла разрабатываемых программных изделий.

Домен, матрица трассировки, оптимизация процессов, предметно-ориентированное проектирование, процесс разработки, управление требованиями

С начала апреля 2011 г. в открытом акционерном обществе «Научно-инженерный центр Санкт-Петербургского электротехнического университета» (ОАО «НИЦ СПбЭТУ») ведутся работы по внедрению на уровне предприятия процесса разработки программных изделий на основании требований. Целью внедрения описываемого процесса является оптимизация разработки за счет снижения временных затрат на управление, издержек, связанных с доработкой программных изделий, а также достижения требуемого уровня качества выпускаемых программных изделий.

На сегодняшний день описываемый процесс внедрен в пилотной зоне, в качестве которой выбрана группа проектов по разработке программных продуктов для Федеральной таможенной службы Российской Федерации (ФТС России). В данной статье рассматриваются результаты поэтапного внедрения, приводятся решения возникших проблем.

На момент принятия решения о необходимости внедрения существующий процесс разработки программных изделий не содержал в себе отдельно выделенных подпроцессов сбора, описания требований и управления ими. Поэтому возникла необходимость их выделения и формализации.

В соответствии с международным стандартом IEEE Std 610.12 требованием было принято считать условие или возможность, требуемые пользователем для решения задач или достижения цели; условие или возможность, которые должны удовлетворяться системой (компонентом системы), или которыми система (компонент системы) должна обладать для обеспечения условий контракта, стандартов, спецификаций или других регулирующих документов.

Учитывая требования международного стандарта IEEE Std 830-1998 и особенности существующего на тот момент процесса разработки, в качестве атрибутов требования были выделены:

1. Идентификатор требования (обязательный).
2. Наименование требования (обязательный).
3. Тип требования (обязательный).
4. Описание требования (обязательный).

4.1. Перечень потребителей (обязательный).

4.2. Состав и структура входных и выходных данных, ограничения (условия) на их значения (обязательный).

4.3. Описание или формализованные модели алгоритмов реализации требования (обязательный для следующих типов требований: «функциональные требования», «ограничения»).

4.4. Математическое обоснование (необязательный).

4.5. Описание прототипов интерфейса пользователя (если интерфейс пользователя необходим для реализации требования) (обязательный для следующих типов требований: «внешние интерфейсы»).

4.6. Спецификации и описание протоколов взаимодействия (необязательный).

4.7. Ограничения на реализацию требования (необязательный).

5. Наименования компонентов (обязательный).

6. Способ проверки выполнения требования (обязательный).

7. Описание взаимосвязи с другими требованиями (обязательный).

8. Источник требования (обязательный).

9. Приоритет (обязательный).

10. Срок реализации (обязательный).

На основании этих данных был сформирован стандарт предприятия «Постановка задачи. Требования к содержанию и оформлению» (далее Стандарт), который ввел понятие документа «Постановка задачи» и формализовал процесс его создания. Применение данного Стандарта на практике позволило еще на стадии описания проверить полноту и непротиворечивость требований, используя атрибут «Способ проверки выполнения требования», и отследить изменения во взаимосвязанных требованиях за счет наличия типизированных связей. Более того, получаемое описание стало основой для создания тестовых сценариев проверки реализуемого функционала, сократив тем самым время на подготовку тестового окружения, а значит, и время на подготовку версии выпускаемого программного изделия. Однако временные затраты на создание документов с описанием постановки задачи в значительной мере возросли за счет необходимости ручного отслеживания взаимосвязей и ввода однотипных данных. В связи с этим было принято решение об автоматизации подпроцесса описания требований.

Для этих целей было выбрано программное средство для управления бизнес-процессами JIRA. В соответствии со Стандартом в данном программном средстве были настроены формы для описания требований. Это позволило сократить временные затраты за счет использования разного рода справочной информации и автоматизированной работы со связями. Практика применения JIRA показала, что при таком варианте реализации все участники процесса разработки программного изделия в любой момент времени имеют доступ к любому из требований и в автоматическом режиме получают информацию о имеющихся между ними связях, что в значительной мере упростило коммуникации внутри проектной команды.

Параллельно с формализацией подпроцесса описания требований большое внимание уделялось также подпроцессу сбора требований. На момент принятия решения о необходимости внедрения рассматриваемого процесса за сбор требований практически полностью отвечал аналитик проекта. Он выступал в качестве посредника между конечными пользователями и заказчиком (с одной стороны) и проектной командой, полностью взяв на себя обязанности по преобразованию пользовательских требований и требований, содержащихся в техническом задании на разработку, в формализованное описание требований, соответствующее принятому Стандарту. С одной стороны, это давало положительный эффект от того, что все коммуникации между заинтересованными лицами шли через одного человека, что не приводило к искажению информации при передаче ее между несколькими посредниками. С другой стороны, аналитику приходилось постоянно общаться на «двух языках»: на языке предметной области заказчика и пользователей и языке технических специалистов проектной команды. Опыт разработки программных продуктов с использованием такого взаимодействия показал, что в значительной мере возрастают риски, связанные с работой аналитика, его постоянной доступностью для заинтересованных лиц, способностью быстро и точно транслировать запросы заказчика и пользователей проектной команде и наоборот. Для того чтобы снизить указанные риски, было принято решение обратиться к предметно-ориентированному проектированию (domain-driven design), суть которого заключается в создании исключительно простых (при этом не примитивных) моделей, понятных всем участникам

процесса разработки. Изучив особенности данного подхода [1] и его внедрения [2], было принято решение о поэтапном внедрении в процесс управления разработкой программных изделий. На сегодняшний день для выбранной в качестве пилотной зоны группы проектов введено понятие домена и универсального языка.

Для понимания и описания ключевых деталей, специфичных для предметной области, абсолютно необходимо использование непротиворечивого и четкого языка. Под языком в предметно-ориентированном проектировании понимаются концепции и их связи. Универсальный язык одинаково понятен всем участникам процесса. Использование универсального языка внутри проектной команды позволило ее участникам более глубоко вникнуть в предметную область, понять потребности заказчика и пользователей. Кроме того снизились риски, связанные с работой аналитика, за счет отсутствия необходимости преобразования терминологии, используемой заказчиком и пользователями, в терминологию, используемую проектной командой.

Следующим шагом внедрения предметно-ориентированного проектирования в подпроцесс сбора требований стал ввод понятия «Домен», который формируется в ходе стратегического проектирования. Домен представляет собой совокупность наиболее общей и целостной модели и универсального языка. В центре стратегического проектирования и получаемого в результате домена лежит то множество частей, из которых состоит общая модель, а также взаимосвязи между этими частями. Такие части, которые при предметно-ориентированном проектировании можно считать «малыми моделями», существуют в ограниченных контекстах. Для каждой «малой модели» определяется контекст, в котором она существует. При работе с контекстами создавались максимально выразительные модели – такие, которые дают представление именно о назначении системы, а не о конкретной реализации. Когда удается этого достичь, то концепции предметной области, определяющие универсальный язык, естественным образом оказываются «на поверхности». Модели получаются гибкими и удобными для дальнейших изменений. В результате проектирование способа реализации, описываемое в соответствии со Стандартом в виде формализованных требований, отходит на второй план и происходит всегда к контексте спроектированного домена.

Опыт применения внедренных подходов предметно-ориентированного проектирования показал, что проектирование домена до момента описания требований и проектирования способов реализации позволяет в значительной мере снизить временные затраты на разработку программного изделия. Снижение достигается за счет уменьшения количества ошибочных решений, связанных с недостаточным пониманием целей и потребностей заказчика и пользователей, которые приводят к необходимости повторного проектирования и реализации части функционала. Кроме того в значительной степени упрощается коммуникация и увеличивается вовлеченность заказчика и пользователей в процесс разработки программного изделия, так как модели, построенные в рамках проектирования домена, просты и понятны всем участникам в отличие от существовавших ранее технических моделей, описывающих исключительно варианты реализации. Основной сложностью при использовании подходов предметно-ориентированного проектирования является необходимость выделения значительного времени на проектирование домена и разработку универсального языка. При этом должны пресекаться любые попытки проектирования реализации. После получения некоторого опыта и вовлеченности всех участников проектной команды в процесс проектирования этот этап естественным образом вошел в процесс разработки программных изделий.

Следующим шагом внедрения процесса управления разработкой программных изделий на основании требований стало введение подпроцесса управления требованиями, включающего идентификацию, сбор, описание, анализ, отслеживание, приоритезацию требований, достижение соглашения по требованиям и затем управление изменениями на всем жизненном цикле разрабатываемого изделия. В соответствии с рекомендациями стандарта ANSI PMI PMBOK® A Guide to the Project Management Body of Knowledge (PMBOK® Guide) [3] для этих целей была введена матрица взаимосвязи или трассировки требований, содержащая информацию об источнике требования, связанных с данным источником формализованных описаниях, подготовленных в соответствии со Стандартом, приоритетах, сроках разработки, тестовых сценариях и разделах документации, содержащих описание реализованного функционала. Использование матрицы трассировки позволило связать между собой ре-

зультаты таких подпроцессов, как сбор и описание требований, реализацию требований, тестирование, документирование результатов. Применение матрицы трассировки в процессе управления разработкой программных изделий позволило достичь ряда положительных результатов. Снизились временные затраты на контроль полноты выполнения требований технического задания за счет того, что матрица трассировки позволяет увидеть полную картину покрытия высокоуровневых требований, описанных в техническом задании, детальными и формализованными требованиями, описанными в соответствии с принятым Стандартом. Выпускаемые версии программных продуктов более полно соответствуют текущим потребностям заказчика и пользователей, что было достигнуто за счет разработки в первую очередь более приоритетных требований и возможности изменения приоритетов в зависимости от потребностей. Достигнуто снижение трудоемкости разработки программного продукта за счет отсечения требований, которые не являются потребностями заказчика и отсутствуют в техническом задании на разработку. Детализация высокоуровневых требований и управление приоритетами позволили, при сокращении сроков выпуска версии программного изделия, выпускать версию, содержащую минимальный допустимый набор функций, расширяя в дальнейшем, с выпуском следующей версии, наполнение каждой функции. Данный подход позволяет конечным пользователям приступить к работе сразу после выпуска версии, не дожидаясь выхода последующих. Еще одним положительным результатом применения матрицы трассировки стало сокращение временных затрат на тестирование и написание документации для конечных пользователей в случае изменения отдельных функций разрабатываемого программного изделия, что дотигается за счет четких связей между требованием на разработку, связанными с ним тестовыми сценариями и разделами документации, описывающими реализацию изменяемой функции. Кроме того использование матрицы трассировки позволило упростить процесс управления ожиданиями пользователей. Каждый запрос конечного пользователя фиксируется в матрице трассировки. Появление в соответствующем разделе матрицы информации по данному запросу позволяет говорить об изменении его статуса. Например, при появлении идентификаторов тестовых сценариев, проверяющих требование, можно говорить, что версия, содержащая изменения, запрашиваемые пользователем, находится на тестировании. Таким образом, матрица трассировки на данном этапе является основным источником информации для процесса управления разработкой программного изделия, позволяя не только оценивать текущее состояние процесса, но и осуществлять как краткосрочное, так и долгосрочное планирование. Стоит отметить, что при таком количестве положительных результатов нельзя не обратить внимание на тот факт, что опыт построения матриц трассировки для проектов, находящихся на момент начала внедрения описываемого процесса на завершающих стадиях разработки или сопровождении, показал достаточно высокую сложность и трудоемкость построения матриц. В данном случае был применен упрощенный подход. Матрица составлялась не на программное изделие в целом, а лишь на версию, выпускаемую на момент внедрения практики использования матрицы трассировки. Такой подход незначительно увеличил трудоемкость по данным проектам, но при этом позволил создать отправную точку для дальнейшего расширения матрицы трассировки при планировании и разработке следующих версий. С целью сокращения временных затрат на построение матрицы используемое для описания требований программное средство JIRA было настроено таким образом, чтобы на основании сведений, указанных в описании каждого требования, и информации о взаимосвязи требований матрица трассировки строилась в автоматическом режиме. Автоматическое построение обеспечило возможность получения актуальных сведений о состоянии процесса разработки программного изделия в любой момент времени.

Таким образом, если обобщенно описать процесс управления разработкой программных изделий, внедренный в ОАО «НИЦ СПбЭТУ», то он представляет собой создание и управление матрицей трассировки требований в контексте спроектированного домена. Такой процесс показал свою эффективность. Он позволяет в значительной мере снизить временные и трудозатраты на разработку. Более того, позволяет вовлечь заказчика и конечных пользователей в процесс проектирования, что не характерно для проектов с государственным участием, которыми являются проекты пилотной зоны. Это позволило приблизить процесс разработки программных изделий для государственных заказчиков к процессу раз-

работки коммерческих программных изделий, повысить качество выпускаемой продукции. При этом процесс достаточно гибкий, за счет изменения структуры матрицы трассировки можно добиться управления любым количеством важных для проекта атрибутов. В ходе внедрения и использования процесса был выявлен ряд проблем, решения которых были найдены и описаны в данной статье. В будущем планируется внедрение описанного процесса управления разработкой программных изделий на всем предприятии. На сегодняшний день развитие процесса не прекращается. Успешный опыт использования подходов предметно-ориентированного проектирования позволяет предположить, что его внедрение уже непосредственно в этап проектирования реализации будет способствовать дальнейшему улучшению процесса, уменьшению трудоемкости и временных затрат на разработку.

СПИСОК ЛИТЕРАТУРЫ

1. Эванс Э. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем / Пер. с англ. М.: Вильямс, 2011.

2. Vaugh V. Implementing Domain-Driven Design. Addison-Wesley, 2013.

3. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 4th ed. Newtown Square (PA) / Project Management Institute. 2008.

S. A. Romanenko, A. S. Skripnikova

THE DEVELOPMENT PROCESS BASED ON THE REQUIREMENTS

The article deals with the development process of software products. The features of the process, its advantages and disadvantages, the difficulties that arose in the course of its implementation and use, are discussed. The positive impact of the process chosen by the interaction of stakeholders at all stages of the life cycle of software products is underlined.

Domain, traceability matrix, process optimization, domain-driven design, process development, requirements management

УДК 681.5.017

О. С. Черная, Ю. Ю. Федорова, С. А. Беляев

Применение методов и средств автоматизированного тестирования для проверки качества программных комплексов обработки измерительной информации

Описывается подход к построению обобщенных сценариев автоматизированного тестирования, показана структура классов, позволяющая строить такие тесты, рассматриваются преимущества и недостатки указанного подхода.

Автоматизированное тестирование, обобщенные сценарии, WindowTester Pro

Уровень сложности современных программных комплексов настолько велик, что уже никого не удивишь переходом от ручного тестирования к автоматизированному. При выполнении ручного тестирования проверка функционирования осуществляется человеком, который вводит данные, изменяет состояние элементов управления и контролирует поведение программного комплекса. При этом время тестирования существенно зависит от количества элементов ввода-вывода данных и возможностей по использованию элементов управления, сложность растет экспоненциально. Ключевыми недостатками данного подхода являются время проверки, которое зависит от количества проверяемых комбинаций, и «человеческий фактор».