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».

Автоматизированное тестирование использует программные средства для выполнения тестов и проверки результатов, что помогает сократить время тестирования и упростить его процесс*. При этом существует 2 основных подхода: тестирование на уровне кода, в частности, модульное тестирование, и тестирование интерфейса пользователя, когда осуществляется имитация действий пользователя с помощью специальных тестовых фреймворков. Современные инструменты тестирования интерфейса пользователя в общем случае поддерживают не только выполнение сценария теста, но и его запись, когда действия пользователя записываются и затем могут воспроизводиться в автоматическом режиме. Сценарии сохраняются в формате, поддерживаемом конкретным фреймворком, и могут изменяться пользователем перед выполнением. В частности, фреймворк WindowTester Pro** сохраняет сценарии в формате модульных тестов на языке Java, что позволяет объединить в одном инструменте как модульное тестирование, так и тестирование интерфейса пользователя.

Ключевые преимущества автоматизированного тестирования***:

– исключение «человеческого фактора» – автоматическое выполнение сценария не пропустит тест «по неосторожности» и не ошибется в результатах, что повышает качество результата;

– быстрое выполнение – автоматическое тестирование не требуется сверки с инструкциями и документацией;

– небольшие затраты на поддержку – когда сценарии уже написаны, на их поддержку и анализ результатов требуется, как правило, меньше времени, чем на проведение того же объема тестирования вручную;

– автоматическое формирование отчетов о результатах тестирования;

– непрерывное выполнение тестов, в том числе и в «нерабочее время».

Ключевые недостатки автоматизированного тестирования****:

– все написанные сценарии всегда будут выполняться однообразно, что не позволит обнаружить ошибку, которая была бы «видна» человеку, например, на экранной форме отобразились не те данные, но сценарий предусматривает проверку других элементов;

– чем чаще изменяется приложение, тем выше затраты на поддержку сценариев;

– разработка автоматизированных тестов – это сложный процесс, так как фактически идет разработка приложения, которое тестирует другое приложение, что повышает требования к знаниям и умениям тестировщика.

Таким образом, выбор подхода к тестированию существенно зависит от особенностей тестируемого приложения и процесса его разработки, а также квалификации тестировщика и требований к качеству.

Основные особенности программных комплексов обработки измерительной информации:

– обработка большого массива исходных данных;

– наукоемкие подходы по обработке данных;

– отличия между программными комплексами заключаются в основном в способе обработки данных, а не в способах представления.

Для разработки программных комплексов обработки измерительной информации используется модель инкрементальной разработки, когда разные части программного кода разрабатываются в разное время и в общем случае не должно возникать изменений в уже завершенных частях.

Учитывая указанные особенности программных комплексов и процесса их разработки, появляется возможность применять автоматизированное тестирование. Рассмотрим в качестве языка разработки – Java, в качестве средства тестирования – WindowTester Pro. В результате сценарии генерируются на языке Java в виде Unit-тестов. В связи с незначительными отличиями в интерфейсе пользователя возможна однократная генерация сценария, а после простой доработки Java-кода – получение однотипных сценариев для всех программных комплексов. Такой подход позволяет с использованием рефакторинга [см. лит.] выделить в сценариях общие части и сформировать обобщенный сценарий, который обеспечит тестирование нескольких программных комплексов.

Основные отличия программных комплексов обработки измерительной информации, которые должны учитываться при формировании обобщенных сценариев:

– отличия в интерфейсе пользователя;

– отличия в алгоритмах, реализующих серверную логику;

– отличия в исходных данных;

* Автоматизированное тестирование // http://ru.wikipedia.org/wiki/Автоматизированное%20тестирование.

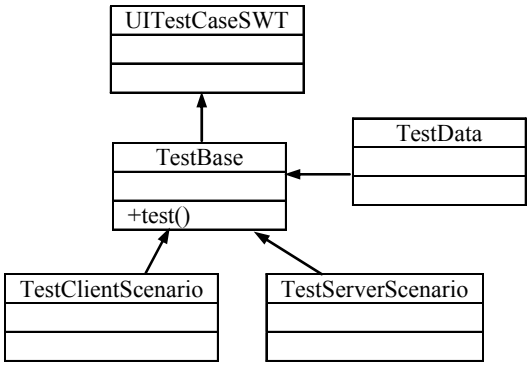
** WindowTester Pro User Guide //https://developers.google.com/java-dev-tools/wintester/html/index.

*** Автоматизация тестирования программных систем // http://habrahabr.ru/post/160257/.

**** Там же.

– особенности процесса работы конкретного программного комплекса.

С учетом перечисленных особенностей разработана структура классов, позволяющая создавать обобщенные сценарии (рисунок). UITestCaseSWT – базовый класс, обеспечивающий тестирование. В рамках TestBase реализуется базовая функциональность, позволяющая исполнять обобщенные сценарии в интересах всех программных комплексов обработки измерительной информации, TestData предоставляет уникальные данные, TestClientScenario позволяет тестировать уникальные особенности клиентских приложений, TestServerScenario – серверных приложений.



Использование обобщенных сценариев изменяет классическую технологию автоматизированного тестирования. При создании обобщенного сценария для тестирования интерфейса пользователя необходимо выполнить следующие шаги:

1) запись сценария с использованием штатных инструментов тестирования на одном из программных комплексов;

2) определение уникальных для всех программных комплексов последовательностей действий;

3) в сгенерированных Unit-тестах для уникальных последовательностей действий вынесение с использованием рефакторинга фрагментов Java-кода в классе на базе TestClientScenario;

4) в случае отличий в поведении серверной части описание соответствующей логики на базе TestServerScenario;

5) выделение данных, необходимых для исполнения сценария, и помещение их в классы TestData;

6) формирование TestClientScenario и TestData для остальных программных комплексов с учетом их особенностей.

Выполнение шага 4 требует от тестировщика не только хорошего знания инструментов автоматизированного тестирования, программирования на Java и умения писать Unit-тесты, но и знания логики построения серверной части, что в общем случае известно только разработчику. Целесообразным представляется данный шаг выполнять авторам серверной части приложений.

Указанную последовательность шагов необходимо повторить столько раз, сколько обобщенных сценариев необходимо создать для полноценного тестирования всех программных комплексов обработки измерительной информации.

В процессе выполнения тестов по обобщенным сценариям на основании указанной структуры классов будут автоматически формироваться уникальные сценарии тестирования для конкретного программного комплекса, учитывающие как особенности построения клиентской и серверной части, так и входных и выходных данных.

Преимущества обобщенных сценариев:

– при наличии обобщенного сценария, созданного для одного из программных комплексов, формирование сценариев для остальных программных комплексов является нетрудоемкой задачей;

– в случае изменений и исправления ошибок в общей части программных комплексов изменяется общая часть сценариев тестирования, что сокращает время внесения изменений;

– при появлении новых аналогичных программных комплексов сложность создания сценариев существенно меньше, чем при написании их с самого начала;

– при появлении новой функциональности требуются незначительные изменения обобщенного сценария.

Недостатки обобщенных сценариев:

– существенное время разработки первого обобщенного сценария, которое может окупиться только при условии его повторного использования;

– высокая сложность выделения общих частей, которая требует соответствующей квалификации тестировщика;

– объем обобщенного сценария в общем случае больше обычного сценария.

В результате преимущества обобщенных сценариев заключаются в возможности их повторного использования для идентичных программных комплексов, что значительно снижает трудоемкость разработки новых сценариев. Данный подход с учетом указанных особенностей применим для программных комплексов обработки измерительной информации, но может быть недостаточно эффективен при выполнении автоматизированного тестирования в других областях.

СПИСОК ЛИТЕРАТУРЫ

Рефакторинг: улучшение существующего кода / М. Фаулер, К. Бек, Д. Брант и др. М.: Символ-Плюс, 2009.

O. S. Chernaya, Yu. Yu. Fedorova, S. A. Belyaev

APPLYING OF METHODS AND TOOLS TO AUTOMATED TESTING OF TELEMETRIC DATA PROCESSING SOFTWARE

The method of common automated testing scenarios design, classes structure for such scenarios, advantages and lacks are described in the article.

Automated testing, common scenarios, WindowTester Pro

УДК 681.3, 681.5

М. С. Куприянов, Ю. А. Шичкина

Схема укрупнения операций распараллеливаемого алгоритма

Рассматривается построение информационного графа по последовательному алгоритму и его преобразование к параллельной форме с достаточной шириной ярусов и с последующей реализацией на параллельной вычислительной среде. Показано также, что формализация данного процесса позволяет найти оптимальное решение задачи распараллеливания алгоритма с учетом таких параметров, как количество процессоров, время вычислений и плотность вычислений на единицу времени.

Параллельный алгоритм, информационный граф, ширина графа, список следования

Практически одновременно с появлением компьютера человечеству стало ясно, что созданные компьютеры не в состоянии решить за приемлемое время многие задачи. Несмотря на то, что производительность компьютеров росла экспоненциально начиная с 1945 г. и продолжает расти в настоящее время, потребность в увеличении скорости обработки данных превышает показатели этого роста.

Параллельное программирование, как и обычное программирование, опирается на алгоритмы и структуры данных. Неэффективно работающая программа – это прямые потери производительности компьютера, средств на его приобретение, усилия на освоение и т. п. Таких потерь хотелось бы избежать или, по крайней мере, их минимизировать. Для этого необходимы апробированные методика анализа алгоритмов и методы их оптимизации.

Чаще всего оптимизация параллельных алгоритмов рассматривается в условиях неограниченного параллелизма, которые предполагают использование идеализированной модели параллельной вычислительной системы и которых на практике не существует.

Эффективность параллельного алгоритма зависит от многих параметров. Одним из них является равномерная загрузка процессоров вычислительной системы. При этом важно, с одной стороны, сохранив высоту параллельного алгоритма, минимизировать его ширину. Эту проблему можно решить с помощью метода, представленного в [1]. Другим важным параметром является время выполнения процессов. Если предполагать, что время выполнения всех процессов, отраженных в информационном графе в виде отдельных вершин, одинаково, тогда ориентированный ациклический информационный граф является наиболее подходящим инструментом для построения оптимального по высоте и ширине параллельного алгоритма. Но на практике такие условия – крайняя редкость. Следовательно, из-за неравномерности выполнения отдельных процессов часть процессоров будет простаивать в ожидании результатов от своих предшественников. Поэтому сама собой напрашивается замена ориентирован-