Голосовский Михаил Сергеевич

МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В РАМКАХ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ РАБОТ

В статье проведен обзор проблем, возникающих при разработке экспериментальных образцов программных продуктов в рамках научно-исследовательских работ. Предложена модель жизненного цикла программного обеспечения в рамках НИР. Рассмотрены особенности применения данной модели и возможности использования в ней прикладных инструментальных средств.

Адрес статьи: www.gramota.net/materials/1/2013/4/12.html

Источник

Альманах современной науки и образования

Тамбов: Грамота, 2013. № 4 (71). С. 40-43. ISSN 1993-5552.

Адрес журнала: www.gramota.net/editions/1.html

Содержание данного номера журнала: www.gramota.net/materials/1/2013/4/

Таким образом, в работе проведено исследование электродинамических характеристик одномерного фотонного кристалла, содержащего чередующиеся слои композитного материала со сверхпроводящими включениями сферической формы и слои диэлектрика. Исследована динамика изменения картины запрещенных зон в зависимости от толщины слоев фотонного кристалла и от концентрации сверхпроводящих включений в слое сверхпроводника. Проведенные расчеты позволяют выдавать рекомендации для выбора концентрации сверхпроводящих включений, толщины слоев и материала включений для достижения требуемых параметров пропускания и отражения фотонного кристалла.

Список литературы

1. **Басс Ф. Г., Булгаков А. А., Тетервов А. П.** Высокочастотные свойства полупроводников со сверхрешетками. М.: Наука, 1989. 288 с.
2. **Berman O. L., Boyko V. S., Kezerashvilli R. Ya., Lozovik Yu. E.** Anomalous Far-Infrared Monochromatic Transmission through a Film of Type-II Superconductor in Magnetic Field // Physical Review B. 2008. V. 78. P. 094506-094513.
3. **Berman O. L., Lozovik Yu. E., Eiderman S. L., Coalson R. D.** Superconducting Photonic Crystals: Numerical Calculations of the Band Structure // Physical Review B. 2006. V. 74. P. 094506-094513.
4. **Lozovik Yu. E., Eiderman S. I., Willander M.** The Two-Dimensional Superconducting Photonic Crystal // Laser Physics. 2007. V. 17. № 9. P. 1183-1186.

УДК 004.413.2
Технические науки

В статье проведен обзор проблем, возникающих при разработке экспериментальных образцов программных продуктов в рамках научно-исследовательских работ. Предложена модель жизненного цикла программного обеспечения в рамках НИР. Рассмотрены особенности применения данной модели и возможности использования в ней прикладных инструментальных средств.

Ключевые слова и фразы: модель жизненного цикла ПО; гибкие процессы разработки; *Agile*; итеративная разработка; инкрементная разработка; научно-исследовательская работа.

Голосовский Михаил Сергеевич
ЗАО «Госбук», г. Москва
golosovskiy@yandex.ru

МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В РАМКАХ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ РАБОТ©

Введение

В 1945 году начал работу первый цифровой электронный вычислитель общего назначения ENIAC. За это время архитектура вычислительных машин и процессы разработки программного обеспечения претерпели значительные изменения, во многом благодаря тому, что основные процессы разработки ПО схожи с процессами разработки в других инженерных областях и состоят из следующих стадий: проектирование, разработка (создание образца изделия), испытания (тестирование), серийное производство, сопровождение. Но, при этом, программная инженерия – относительно молодая область инженерной науки, и опыта успешных проектов очень мало (до 80% программных проектов терпят крах [1]). Сложность разработки ПО вызывается четырьмя основными причинами [2; 8]:

– сложностью реальной предметной области, из которой исходит заказ на разработку;
– трудностью управления процессом разработки;
– необходимостью обеспечить достаточную гибкость программы;
– неудовлетворительными способами описания поведения больших дискретных систем.

В проектах, содержащих в своей основе научно-исследовательскую составляющую, трудность управления процессами разработки усиливается необходимостью поиска оптимального решения задачи НИР, высокой степенью неопределённости предметной области и отсутствием точных и неизменных требований при разработке программного образца.

Модель жизненного цикла разработки программного обеспечения в НИР

В рамках проведения НИР по разработке программного обеспечения основной моделью разработки является водопадная модель, лежащая в основе ГОСТ 34-й серии. Водопадная модель разработки, предложенная Вильямом Ройсом в 1977 году [11], направлена на получение функционирующей программы за один

проход по цепи этапов от постановки требований до тестирования и ввода в эксплуатацию. Но, в условиях неопределённости предметной области, зачастую приходится дорабатывать и изменять облик разрабатываемого программного обеспечения в процессе разработки по следующим причинам:

– ошибки в модели предметной области, допущенные на начальных этапах;

– несостоятельность гипотез, проверяемых при разработке ПО;

– неточности в моделях на уровне человеко-машинного взаимодействия (необходимость доработки интерфейса пользователя);

– отсутствие точного понимания свойств и характеристик результирующего продукта;

– новые технологии и инструменты, используемые в проекте, могут оказаться не в полной мере подходящими для решения поставленных задач.

В связи с этим, классическая водопадная модель оказывается неприменимой, и возникает необходимость использования итеративной модели, ориентированной на частые изменения разрабатываемой системы. В мировой практике при разработке программного обеспечения в условиях изменяющихся требований и высокой неопределённости предметной области применяются итеративные гибкие процессы (методологии) разработки семейства *Agile* [3; 5; 6; 10], такие как экстремальное программирование, *Scrum*, *Kanban*, *OpenUP* и др. Основной идеей, лежащей в основе этих процессов, является частая смена итераций с получением на каждой итерации работающего программного образца, пусть и реализующего не весь функционал, заложенный на этапе постановки задачи и анализа требований. Главным достоинством этого подхода является возможность быстрого реагирования на возникающие изменения в процессе разработки. Но перед внедрением процессов семейства *Agile* (гибкие процессы) стоит следующий ряд препятствий:

– в результате проведения НИР требуется полный комплект документации на разработанное программное обеспечение, что является противоречием основному принципу гибких процессов разработки: «Работающий программный продукт важнее полной документации»;

– снижение качества получаемого программного продукта в тех случаях, когда невозможно получить новую версию за короткую итерацию;

– все проекты в рамках выполнения НИР, как правило, выполняются в жесткой функциональной структуре организации, что препятствует внедрению принципа использования самоорганизующихся команд.

Для решения вышеуказанных проблем был предложена модель разработки программного обеспечения, основанная на смеси инкрементной и итеративной разработок с параллельным формированием программной документации. Существует различие между итеративной и инкрементной разработками программного обеспечения [4; 9]. Итеративная разработка подразумевает создание системы (пусть с ограниченным функционалом) и её последующее улучшение в течение нескольких итераций. Достоинством итеративной разработки является получение работающего образца на каждой итерации с возможностью оценки его соответствия требованиям и оценки необходимости продолжения дальнейшей работы над программным образцом (модулем). В случае если необходимо продолжение разработки, то всегда можно выбрать: продолжать работу с текущей итерацией или развивать альтернативное направление, с одной из более ранних итераций. В этом кроется один из недостатков итеративного подхода – сложность управления итеративными проектами высока в связи с тем, что не всегда можно предсказать количество итераций, которое может потребоваться для разработки программного образца.

При инкрементной разработке программный продукт разбивается на несколько законченных функциональных частей, и в рамках каждого инкремента производятся разработка одной функциональной части и её тестирование. Финальным инкрементом является интеграция всех модулей и тестирование всего программного продукта. Достоинством инкрементной стратегии разработки является легкое управление сроками и расписанием процесса разработки за счет разбиения системы на части (количество частей известно, и можно отслеживать разработку каждой части системы).

Недостатками инкрементной разработки являются:

– возможные ошибки в определении количества модулей на начальном этапе разработки, что может привести к срыву плана разработки;

– отсутствие точного прогнозирования качества результирующего продукта, полученного в результате сборки исходных частей.

Модель разработки ПО в нотации диаграммы деятельности языка UML представлена на Рисунке 1. Она состоит из четырех основных этапов: инициация разработки, постановка инкремента, исполнение инкремента и завершение разработки. На этапе инициации разработки производится определение требований к разрабатываемой системе, анализ предметной области, на основе результатов которого строится модель предметной области, формируются требования и критерии завершения разработки. На этапе постановки инкремента, с использованием модели предметной области, разрабатывается общая архитектура системы, выделяются модули, разрабатываемые в отдельных инкрементах, определяются их интерфейсы для связи между собой и последующей интеграции. В крайнем случае, если систему нельзя разбить на несколько слабо связанных между собой частей (модулей), модель выродится в итеративную. В завершение этапа постановки разрабатываются критерии достижения цели, определяющие, в каком случае можно считать инкремент завершенным.

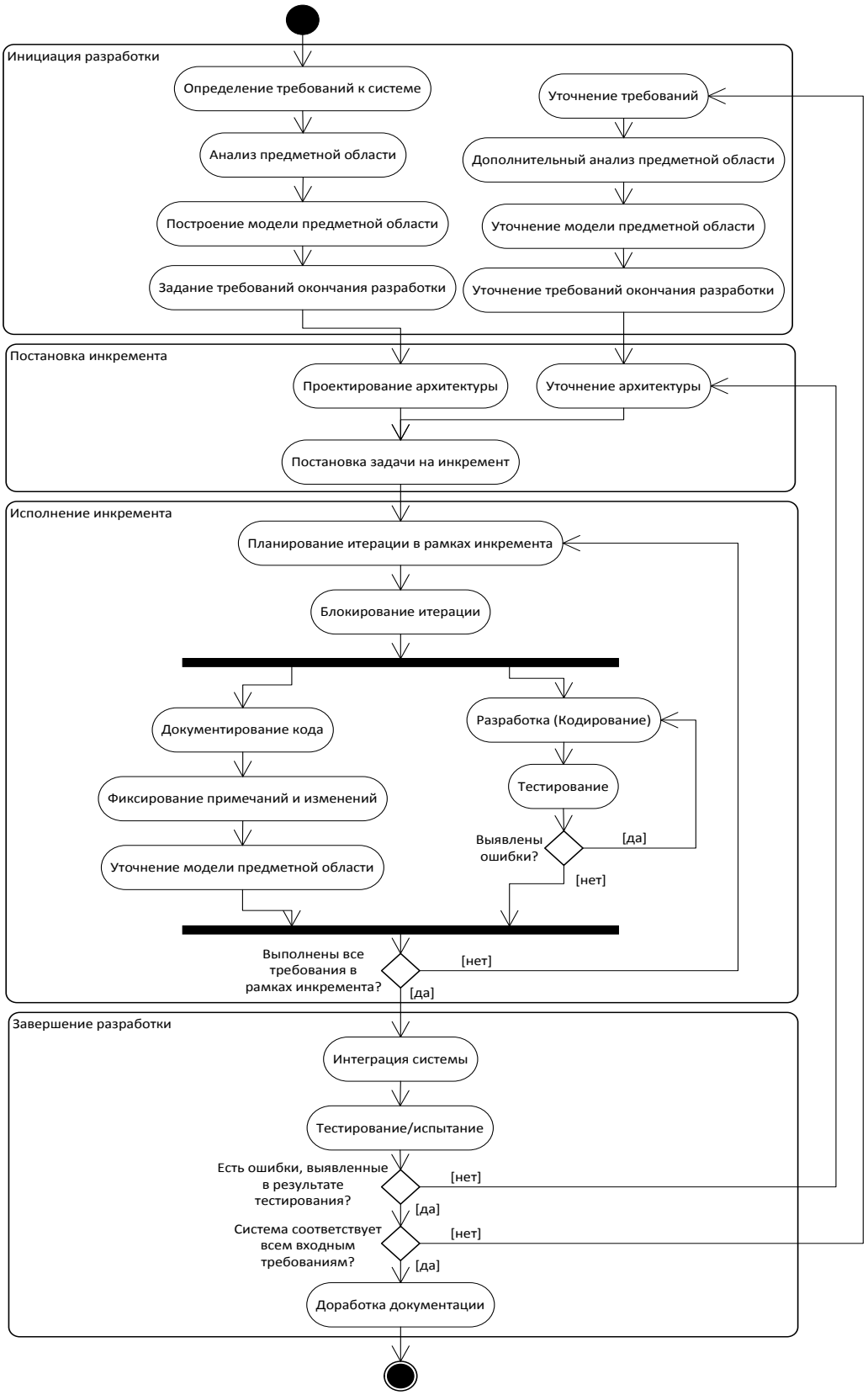

Рис. 1. *Модель жизненного цикла разработки программного образца в рамках НИР*

Следующим этапом после постановки инкремента идет этап исполнения. На этом этапе разрабатывается работающий образец по итеративной стратегии. Начинается этап с планирования итерации. При планировании описываются возможные пути решения задачи и необходимые для их реализации инструменты и методы. На этой стадии можно применять различные методы коллективной генерации идей. После планирования проводится блокирование итерации, при котором выбирается один, наиболее подходящий, вариант плана, методов и инструментов. В случае необходимости, для выбора могут использоваться различные методы

экспертных оценок (голосование, ранжирование, метод анализа иерархий (МАИ) и др.). В общем случае в литературе по гибким процессам разработки рекомендуется применение как коротких итераций – 2-3 дня, так продолжительностью порядка месяца. На практике, в рамках проведения НИР, наиболее эффективной длительностью итерации оказалась неделя. При этом завершение итерации совпадает с еженедельным подведением итогов. В связи с тем, что планируемая длительность итерации оказывается равной 5 дням, на планирование и блокировку итерации отводится порядка 4 часов. На выходе процесса блокировки итерации формируется план выполнения, содержащий в себе задачи на итерацию, список исполнителей и их распределение по задачам, приоритеты или последовательность выполнения задач. После выполнения блокировки итерации выполнение идет по двум параллельным ветвям: разработка программного образца и написание документации. При разработке программного образца рекомендуется применять стандарты кодирования, единые в рамках организации, а также выполнять комментирование программного кода. Для быстрого получения документированного программного кода можно использовать системы документирования исходных текстов *Doxygen* или *PHPDocumentor*. Также в процессе разработки часто возникают идеи, как можно сделать эту итерацию другими способами или улучшить разрабатываемый программный образец в рамках итерации. При выполнении этой модели должно выполняться основное правило: любые изменения в рамках итерации запрещены. Это связано с тем, что новые идеи могут возникать довольно часто, при этом частые изменения состава задач и инструментов в течение итерации могут привести к тому, что в результате итерации ничего не будет реализовано. Но возникающие идеи могут нести рациональное зерно, к тому же в результате разработки могут обнаруживаться ранее не учтенные пробелы в предметной области. Все предложения фиксируются в журнале предложений и изменений, там же фиксируются все выполненные за время итерации запланированные изменения. В случае необходимости, вносятся изменения в модель предметной области. Ветка разработки (написания кода) завершается тестированием, если программный код не проходит тестирование, он возвращается на этап разработки. Рекомендуется в процессе тестирования использовать технологии TDD (Test Driven Development) [7] или модульное тестирование (Unit Testing). После прохождения тестирования проверяются все требования, поставленные на инкремент, выполнены или нет, в случае, если все требования не выполнены, производится новая итерация, но при планировании и блокировке итерации учитываются предложения и изменения, полученные на предыдущей итерации. За этапом исполнения инкремента следует этап завершения разработки. На этом этапе производится интеграция модулей, полученных в рамках предыдущих инкрементов. По завершению интеграции производятся тестирование и проверка всей системы на наличие ошибок. В случае, если выявляются ошибки, полученные в результате интеграции, система отправляется на доработку, начиная с этапа постановки инкремента, на котором производится уточнение архитектуры системы и выполняется одна или несколько итераций по доработке. Тестирование на соответствие исходным требованиям (проведение испытаний) производится после устранения ошибок, возникших в результате интеграции модулей. Недостатки и несоответствия, выявленные в процессе испытаний, могут носить более глобальный характер и потребовать уточнения требований, проведения дополнительного анализа предметной области, уточнения требований окончания разработки. По результатам выполненных испытаний формируется акт проведения испытаний. Заключительным процессом этапа завершения разработки является процесс доработки документации.

Заключение

Предложенная модель жизненного цикла разработки программного обеспечения была применена при создании экспериментального образца программного комплекса (ЭОПК) платформы в рамках НИР «Исследование и оптимизация модели сетевого взаимодействия участников распределенной редакционной коллегии в процессах создания, редактирования, рецензирования и публикации материалов электронных изданий» при финансовой поддержке Министерства образования и науки Российской Федерации. Использование данной модели позволило получить работающий ЭОПК платформы, сократить время на разработку отчетной и программной документации.

Список литературы

1. **Брукс Ф.** Мифический человеко-месяц, или Как создаются программные системы. М.: Символ-Плюс, 2007. 304 с.
2. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008. 720 с.
3. **Кон М.** Scrum: гибкая разработка ПО. М.: Вильямс, 2011. 576 с.
4. **Ларман К., Базили В.** Итеративная и инкрементальная разработка: краткая история // Открытые системы. 2003. № 9. С. 43-53.
5. ***Agile*-манифест разработки программного обеспечения** [Электронный ресурс]. URL: http://agilemanifesto.org/iso/ru/ (дата обращения: 24.01.2013).
6. **Beck K.** Embracing Change with Extreme Programming // Computer. 1999. Vol. 32. № 10. P. 70–77.
7. **Beck K.** Test Driven Development by Example. Addison-Wesley Professional, 2002. 252 p.
8. **Brooks F.** No Silver Bullet — Essence and Accident in Software Engineering // IEEE Computer. 1987. № 20. P. 10–19.
9. **Cockburn L.** Using Both Incremental and Iterative Development // Cross Talk. 2008. May. P. 27-30.
10. **Larman C.** Agile and Iterative Development: a Manager's Guide. Addison-Wesley Professional, 2004. 342 p.
11. **Royce W.** Managing the Development of Large Software Systems // TRW. 1970. August. P. 328-338.