

И.Ш. НЕВЛЮДОВ, докт. техн. наук, профессор, ХНУРЭ, г. Харьков
В.В. ЕВСЕЕВ, канд. техн. наук, доцент, ХНУРЭ, г. Харьков
В.О. БОРТНИКОВА, студент, ХНУРЭ, г. Харьков

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРИ РАЗРАБОТКЕ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ ТЕХНОЛОГИЧЕСКОЙ ПОДГОТОВКИ ПРОИЗВОДСТВА

Рассматривается эффективность использования моделей жизненного цикла разработки программного обеспечения в соответствии с международным стандартом ISO/IEC 12207:1995 “Information Technology – Software Life Cycle Processes” для разработки программного обеспечения для корпоративных информационных систем технологической подготовки производства.

Ключевые слова: Жизненный цикл, программное обеспечение, корпоративно-информационные системы технологической подготовки производства.

Розглядається ефективність використання моделей життєвого циклу розробки програмного забезпечення у відповідності до міжнародного стандарту ISO/IEC 12207:1995 “Information Technology – Software Life Cycle Processes” для розробки програмного забезпечення для корпоративних інформаційних систем технологічної підготовки виробництва.

Ключові слова: Життєвий цикл, програмне забезпечення, корпоративно-інформаційні системи технологічної підготовки виробництва

Efficiency of software development life cycle models using is examined in accordance with the international standard of ISO/IEC 12207:1995 “Information Technology – Software Life Cycle Processes” for technological preproduction corporate informative systems software development.

Keywords: Life cycle, software, corporately-informative systems of technological preproduction.

1. Введение

Основной задачей, с которой сталкивается разработчик корпоративно-информационных систем технологической подготовки производства, является правильность организации и подхода при проектировании программного обеспечения на раннем этапе подготовки технического задания. В ходе этого этапа составляется не только техническое задание и требования к программному обеспечению, но и предварительно выбирается модель жизненного цикла, рассчитывается трудоемкость, стоимость и степень риска. От правильности выбора модели жизненного цикла зависит и успешность реализации программного обеспечения, в соответствии с международным стандартом ISO/IEC 12207:1995 “Information Technology – Software Life Cycle Processes”, в котором предложен ряд моделей жизненного цикла [1]. Однако стоит отметить, что проектирование корпоративных информационных систем технологической подготовки производства является сложной научно-технической задачей, в которой объединяется не только реализация основных функций CAD/CAM/CAE, но и специфических математических моделей и методов решения задач технологической подготовки производства [5]. Вследствие этого возникает

вопрос об эффективности использования стандартных моделей жизненного цикла при проектировании сложных корпоративно-информационных систем технологической подготовки производства.

2. Анализ моделей жизненного цикла программного обеспечения

Корпоративная информационная система – это масштабируемая система, предназначенная для комплексной автоматизации всех видов хозяйственной деятельности больших и средних предприятий, в том числе корпораций, состоящих из группы компаний, требующих единого управления производственными процессами при изготовлении изделий в зависимости от направления деятельности предприятия. Корпоративной информационной системой может считаться система, автоматизирующая более 80% подразделений предприятия. При её разработке стоит учесть специфику предприятия, последовательность выполнения и взаимосвязи процессов, действий и задач с многомодульным подходом к организации корпоративных систем. Выбор модели жизненного цикла зависит от предъявляемых заказчиком требований к сложности, функциональности программного обеспечения [2].

Наибольшее распространение получили следующие модели жизненного цикла разработки программного обеспечения, которые предложены в стандарте ISO/IEC 12207:

- waterfall model (каскадная модель);
- v-shaped model (V-образная модель);
- prototype model (модель прототипирования);
- rapid application development model или RAD-model (модель быстрой разработки приложений);
- incremental model (многопроходная модель);
- spiral model (спиральная модель).

Проведем анализ каждой модели жизненного цикла программного обеспечения для определения её возможностей, при использовании на раннем этапе проектирования технического задания для корпоративно-информационных систем технологической подготовки производства.

Waterfall model (каскадная модель) – применялась при разработке однородных информационных систем в 1970-1980 годах, когда программное обеспечение представляло собой единое целое. Основные этапы каскадной модели представлены на рисунке 1.



Рис. 1. Основные этапы каскадной модели

Принципиальная особенность каскадной модели заключается в том, что переход между этапами осуществляется только после полного завершения работ на предыдущем этапе модели. Это связано с тем, что результаты, полученные при выполнении этапа, являются исходными данными для следующего этапа. При

этом требования, предъявляемые в техническом задании на программное обеспечение, строго документируются и фиксируются на все время выполнения разработки.

Данная модель хорошо зарекомендовала себя при разработке программного обеспечения, для которого достаточно точно или полностью сформулировано техническое задание, а, следовательно, можно точно рассчитать трудоёмкость и стоимость разработки. Однако применение данной модели при проектировании программного обеспечения для корпоративных информационных систем невозможно с той точки зрения, что реальный процесс проектирования никогда не укладывается в жесткую схему каскадной модели, результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на предыдущих этапах. Это заставляет постоянно возвращаться к предыдущим этапам и вносить изменения и уточнения в ранее принятые решения. Вследствие этого заказчики могут внести изменения только в конце каждого этапа и при условии, что эти изменения не затрагивают требований, изложенных в техническом задании. Как правило, это является задержкой при получении результатов, в то же время возрастает риск разработки программного обеспечения, неудовлетворяющего потребностям пользователя.

V-shaped model (V-образная модель) – основана на систематическом подходе к разработке программного обеспечения и определяет четыре базовых этапа проектирования: анализ, проектирование, разработка и обзор. Она является разновидностью каскадной модели, в которой уделяется большое внимание верификации и аттестации программного обеспечения. Структура V-образной модели представлена на рисунке 2.

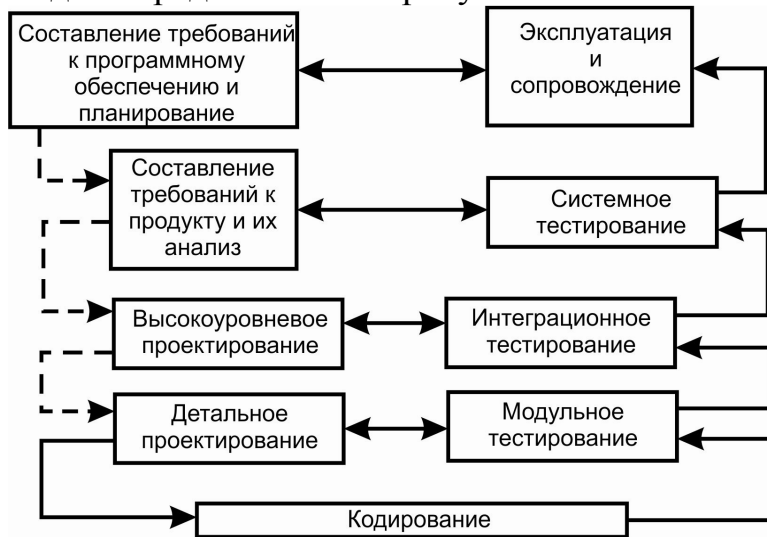


Рис. 2. Структура V-образной модели

Пунктирные стрелки указывают на выполнение этапов параллельно основным (определяются системные требования к программному обеспечению и выполняется планирование работ). Они объединены с такими этапами: составление требований к продукту и их анализ (составляется полное техническое задание); высокоуровневое проектирование (определяется

структура программного обеспечения и внутренние связи); детальное проектирование (определяется алгоритм работы каждого компонента). При таком подходе уделяется большая роль верификации и аттестации программного обеспечения, начиная с ранних этапов проектирования, что позволяет легко отслеживать ход работы, а также выполнение каждого этапа, так как завершение каждого этапа является контрольной точкой.

Однако, основными недостатками данной модели с точки зрения применения ее при проектировании корпоративных систем можно считать следующие: не учитывается интеграция между этапами; нет возможности внесения изменений на разных этапах жизненного цикла; тестирование требований происходит слишком поздно, следовательно, внесение изменений влияет на график выполнения работ. Это не позволяет точно рассчитать трудоёмкость и стоимость разработки.

Prototype model (модель прототипирования) – позволяет создать прототип программного обеспечения до или в течение составления технического задания. Потенциальный заказчик работает с прототипом, определяет его сильные и слабые стороны, а результаты сообщает разработчику [3]. В результате обеспечивается взаимная связь между разработчиком и заказчиком, которая используется для изменений и корректировки прототипа. Модель прототипирования представлена на рисунке 3.



Рис. 3. Модель прототипирования

Этот процесс продолжается до тех пор, пока заказчик не будет удовлетворен степенью соответствия прототипа требованиям, предъявляемым к программному обеспечению в техническом задании. Данный подход обладает рядом преимуществ: взаимодействие заказчика и разработчика программного обеспечения начинается на раннем этапе; заказчик всегда видит прогресс в процессе разработки; прототип представляет собой формальную спецификацию будущего программного обеспечения. Однако, кроме указанных достоинств, данная модель имеет ряд недостатков при проектировании корпоративных систем: решение сложных технических, технологических и специфических задач отодвигается на последний этап, что неприемлемо при разработке корпоративных систем вследствие того, что они есть первичные функции, необходимые заказчику (заказчик может предпочесть получение прототипа, а не законченную полную версию программного обеспечения). Разработка прототипа может неоправданно затянуться в связи с неизвестностью количества выполненных итераций во время разработки прототипа, что увеличивает трудоёмкость и стоимость разработки.

Rapid Application Development model (RAD-model) – в отличие от предыдущих моделей, в RAD-model заказчик играет решающую роль, в тесном взаимодействии с разработчиками программного обеспечения он участвует в

формировании технического задания и апробации прототипа. Модель обладает рядом преимуществ: использование современных инструментальных средств позволяет сократить трудоёмкость разработки; привлечение заказчика на этапах составления требований и разработки интерфейса пользователя, что сводит к минимуму риск получения программного продукта, которым останется не доволен заказчик; повторно используются компоненты уже существующих программ. Основные этапы RAD-model показаны на рисунке 4.

Однако данная модель при ряде преимуществ имеет ряд недостатков, связанных с разработкой корпоративных систем: в зависимости от специфики разрабатываемого программного обеспечения, использование компонентов существующих программ невозможно или сведено к

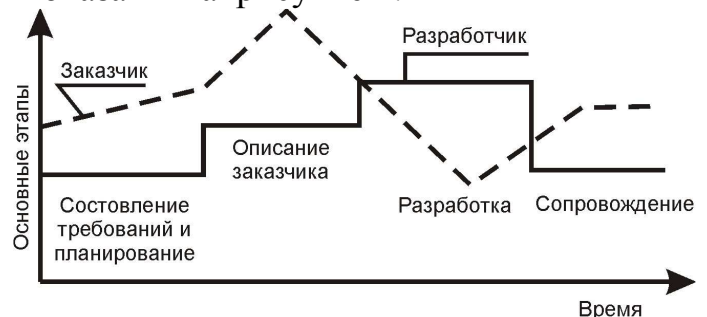


Рис. 4. Основные этапы RAD-model

минимуму; существует большой риск, что разработка корпоративной системы никогда не будет закончена, в связи с заикливанием работ по ее разработке.

Так, RAD-model рекомендуется использовать при разработке программного обеспечения, которое хорошо поддается моделированию, когда требования технического задания хорошо известны, и заказчик может принимать непосредственное участие в процессе разработки.

Incremental model (многопроходная модель) – состоит из объединения процесса построения программного обеспечения с добавлением на каждой интеграции новых функциональных возможностей или повышением эффективности программного обеспечения. Данная модель предполагает, что на начальном этапе жизненного цикла разработки выполняется конструирование программного обеспечения в целом и определяется число инкрементов и относящихся к ним функций. Далее каждый инкремент проходит через оставшиеся этапы жизненного цикла (кодирование и тестирование) [4].

Сначала выполняется конструирование, тестирование и реализация базовых функций, а в последующих интеграциях разработка программного обеспечения направлена на улучшение таких возможностей, как: создание генераторов отчетов, построение графиков зависимости и т.д. Используя данный подход, Incremental model обладает такими преимуществами: в начале разработки требуются средства для реализации основных функций программного обеспечения; после каждого инкремента заказчик получает функциональный продукт; инкременты функциональных возможностей легко поддаются тестированию. Многопроходная модель представлена на рисунке 5.

Однако, данная модель имеет следующие недостатки: не предусмотрены этапы интеграции внутри каждого инкремента; полная функциональность программного обеспечения должна быть указана в начале жизненного цикла в техническом задании; есть возможность тенденции оттягивания решений сложных задач на завершающем этапе; общие затраты трудоёмкости и стоимости программного обеспечения не снижаются по сравнению с другими моделями;

обязательным условием данной модели является наличие хорошего планирования и проектирования программного обеспечения.

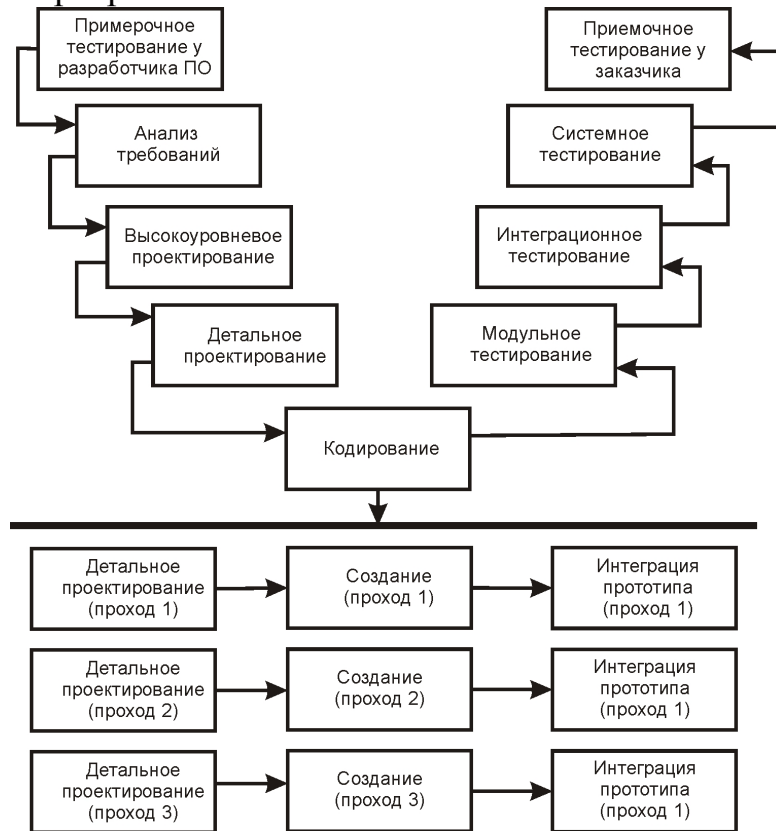


Рис. 5. Многопроходная модель

В результате приведенных выше недостатков данную модель рекомендуют использовать при проектировании программного обеспечения при условии заранее сформулированных требований и выделения для выполнения большого периода времени.

Spiral model (спиральная модель) – особенность данной модели заключается в том, что прикладное программное обеспечение создается не сразу, а по частям (модулям) с использованием метода прототипирования. В данной модели прототип – действующее программное обеспечение, реализующее отдельные функции и внешний интерфейс пользователя. Создание прототипа осуществляется за несколько итераций (витков спирали), каждая итерация соответствует созданию фрагмента или версии программного обеспечения, на которой уточняются цели и характеристики, оценивается качество полученных результатов, а также проводится тщательный анализ риска превышения трудоёмкости и стоимости программного обеспечения. Разработка программного обеспечения итерациями отображает объективно существующий спиральный цикл, позволяя переходить на следующую стадию, не дожидаясь полного завершения работ на текущей стадии, поскольку при итеративном способе разработки недостающие работы можно сделать на следующей итерации. Основные этапы спиральной модели представлены на рисунке 6.

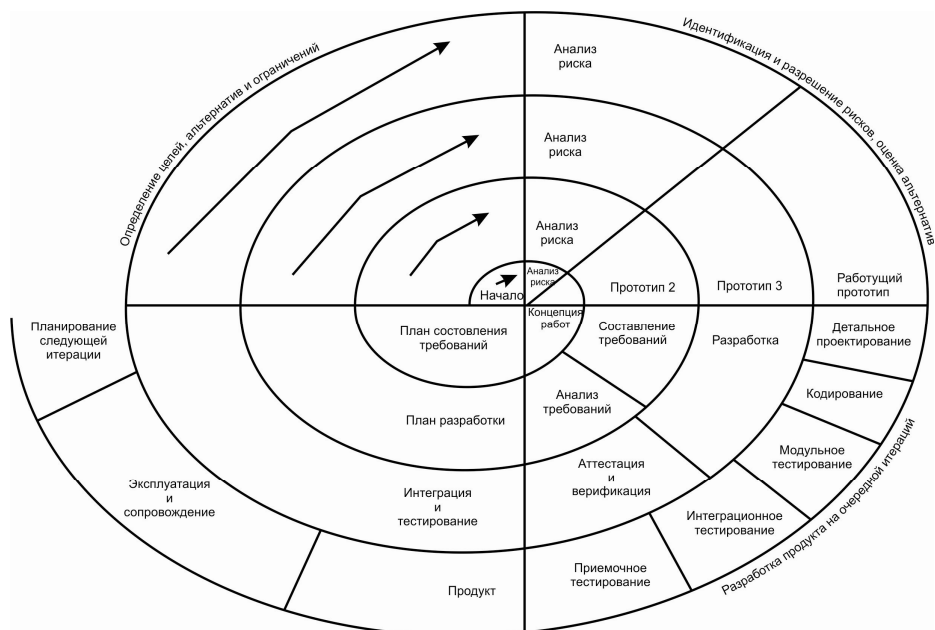


Рис. 6. Спиральная модель

Спиральная модель обладает следующими достоинствами: заказчик принимает активное участие при разработке программного обеспечения; заказчик может увидеть разрабатываемое программное обеспечение на ранних стадиях; спиральная модель объединяет преимущества каскадной и многопроходной модели. К недостаткам данной модели можно отнести следующие: план работы составляется на основе статистических данных, полученных в предыдущих проектах и из личного опыта разработчика; спиральная модель может продолжаться бесконечно, в результате предложений заказчика, которые могут породить новую спираль, а, следовательно, увеличить трудоёмкость, стоимость и время разработки программного обеспечения.

3. Выводы

В ходе анализа стандарта ISO/IEC 12207[5] и моделей жизненного цикла разработки программного обеспечения можно сделать вывод о том, что в стандарте не предлагаются конкретные модели жизненного цикла и методы разработки программного обеспечения для корпоративных систем технологической подготовки производства. Положения стандарта являются общими для любых моделей жизненного цикла методов и технологий разработки, что не позволяет использовать конкретные модели жизненного цикла для решения задач оценки трудоёмкости, стоимости и риска разработки программного обеспечения для корпоративных информационных систем технологической подготовки производства с учетом их специфики и сложности научно-технической реализации на раннем этапе жизненного цикла.

Список литературы: 1. ISO/IEC 12207:1995 Standard for Information Technology–Software life cycle processes – Description. 2. Технологии разработки программного обеспечения: Учебник / С. Орлов. – СПб.: Питер, 2002. – 464 с. 3. Ерик Дж. Брауде Технология разработки программного обеспечения. – СПб.: Питер, 2004. – 655 с. 4. Благодатских В.А. Стандартизация разработки программных средств: Учеб. пособие / В.А. Благодатских, В.А. Волнин, К.Ф. Посакалов; Под ред. О.С. Разумова. – М.: Финансы и статистика, 2005. – 288 с: ил. 5. Невлюдов И.Ш., Андрусевич А.А. Евсеев В.В. Анализ жизненного цикла разработки

УДК 681.3.06

А.В. ПРОХОРОВ, канд. техн. наук, доцент, НАУ
им Н.Е. Жуковского «Харьковский авиационный институт»
Ю.Н. СТРАШЕНКО, аспирант, НАУ им Н.Е. Жуковского
«Харьковский авиационный институт»

ВЗАИМОДЕЙСТВИЕ АГЕНТОВ ИМИТАЦИОННОЙ МОДЕЛИ ПРИ РЕШЕНИИ ЗАДАЧ УПРАВЛЕНИЯ ФИНАНСОВЫМИ РЕСУРСАМИ БАНКА

Представлены общие принципы взаимодействия агентов имитационной модели. Предлагается решение основных задач управления финансовыми ресурсами коммерческого банка и формальное описание онтологических моделей знаний агентов для системы моделирования. Ключевые слова: агент, протоколы взаимодействия, трансфертное ценообразование, онтология.

Дано загальні принципи взаємодій агентів імітаційної моделі. Пропонується вирішення головних задач управління фінансовими ресурсами комерційного банку, формальний опис онтологічних моделей знань агентів для системи моделювання. Ключові слова: агент, протоколи взаємодій, трансфертне ціноутворення, онтологія.

The article represents general principles of agent's interrelation in imitation model. It was suggested making decision of bankroll management tasks. It was described dummy ontological knowledge models of agents for modeling system. Key words: agent, protocols of interrelation, transfer pricing, ontology.

Введение

Развитие современной банковской системы во многом обусловлено векторами развития мировой банковской сферы. Международные стандарты банковской деятельности, известные как система Базель II, переход на которые намечен для Украины в долгосрочном периоде до 2016г., смещают акценты из математического расчета показателей/коэффициентов в направлении оценки качества управления, что потребует от банков предельной автоматизации аналитических процедур и создания новых моделей, позволяющих как в глобальном масштабе, так и в каждом дочернем банке или компании финансового сектора видеть изменение качества активов [1].

В этих условиях актуальной является задача моделирования процессов управления финансовыми ресурсами банка и принятия решений в очень сложной и динамичной среде на основе согласования интересов различных участников банковского процесса.

1. Анализ последних исследований и публикаций

Наиболее актуальным и перспективным направлением исследования в настоящее время является создание систем имитационного моделирования на основе мультиагентного подхода. Рассмотренные особенности, связанные с моделированием процессов управления финансовыми ресурсами банка, хорошо