Обоснованный выбор модели жизненного цикла как фактор повышения качества разработки информационных систем

The Proved Choice of Life Cycle Model as the Factor of Quality Improvement of Information Systems Development

Обґрунтований вибір моделі життєвого циклу як фактор підвищення якості розробки інформаційних систем

Л. К. Бобров, Н. Н. Савиных, Г. Л. Бабченко
Новосибирский государственный университет экономики и управления, Новосибирск, Россия

Leonid Bobrov, Nataliya Savinykh, and Galina Babchenko
Novosibirsk State University of Economics and Management, Novosibirsk, Russia

Л. К. Бобров, Н. Н. Савиних, Г. Л. Бабченко
Новосибірський державний університет економіки та управління, Новосибірськ, Росія

Дается краткая характеристика наиболее распространенных моделей жизненного цикла, используемых при разработке информационных систем. Обсуждается влияние выбора модели жизненного цикла на качество разработки информационных систем. Приводится структурная схема информационной системы, ориентированной на решение задачи обоснованного выбора модели жизненного цикла.

The article is about the most popular life cycle models that are used for information systems development. It is demonstrated strong connection between model of life cycle and quality of information systems development. It is also given a diagram of the information system focused on the well-grounded choice of the life cycle model.

Подано стислу характеристику найбільш розповсюджених моделей життєвого циклу, що використовуються при розробці інформаційних систем. Обговорюється вплив вибору моделі життєвого циклу на якість розробки інформаційних систем. Наведено структурну схему інформаційної системи, орієнтованої на вирішення завдання обґрунтованого вибору моделі життєвого циклу.

Инженерный подход к рассмотрению жизненного цикла, регламентируется международным стандартом ISO/IEC 12207 [1], который определяет состав и последовательность основных (базовых) процессов жизненного цикла программного обеспечения. К этим процессам отнесены заказ, поставка, разработка и сопровождение. Выделены процессы, поддерживающие жизненный цикл, а именно: документирование, управление конфигурацией, обеспечение качества, проверка соответствия, ревизия, решение проблем. К организационным процессам жизненного цикла отнесены процессы управления, поддержки инфраструктуры, совершенствования и обучения. В то же время стандарт не регламентирует реализацию процессов жизненного цикла, его предписания не касаются взаимосвязи перечисленных процессов и их последовательности. На практике наибольшее распространение получили модели жизненного цикла программного обеспечения, рассматриваемые ниже.

Каскадная модель жизненного цикла

Эта модель реализует принцип однократного выполнения каждого вида деятельности в виде заранее ограниченных и однозначно упорядоченных во времени этапов. В типовой каскадной модели [2] все этапы процесса разработки выполняются последовательно и каждый этап завешается выпуском полного комплекта документации на созданный программный продукт. Согласно данной модели каждый последующий этап начинается лишь тогда, когда полностью завершается и документируется предыдущий. Таким образом, каскадная модель представляет собой формальный метод разработки «сверху вниз», который долгие годы успешно использовался при создании автоматизированных систем.

Понимание сильных сторон и недостатков каскадной модели улучшает оценочный анализ других, зачастую более эффективных моделей жизненного цикла, основанных на данной модели. Каскадная модель не рассчитана на динамические изменения в требованиях к создаваемому программному обеспечению, и поэтому ее лучше использовать тогда, когда требования и их реализация максимально четко определены и понятны. Например, если разработчик имеет большой позитивный опыт создания определенного рода систем, и создает еще одну аналогичную систему, основанную на имеющихся разработках. Эффективно использовать каскадную модель можно и при создании новой версии ранее разработанного продукта, когда вносимые изменения вполне определены и управляемы. В качестве идеального примера использования каскадной модели можно рассматривать перенос уже существующего продукта на новую платформу.

Попытки улучшения каскадной модели привели к возникновению ее модифицированных, менее жестких версий. Таковой является итерационная модель, называемая также поэтапной моделью с промежуточным контролем. Это модель разработки, предусматривающая наличие обратных связей между этапами. Здесь на каждом из этапов осуществляется проверка и корректировка разрабатываемой информационной системы, благодаря чему существенно снижается трудоемкость отладки системы по сравнению с традиционной каскадной моделью. Если в ходе промежуточной проверки на каком-либо из этапов обнаружилась ошибка, допущенная на более ранней стадии разработки, то работы этапа, повлекшего ошибку, проводятся повторно. При этом анализируются причины ошибки и корректируются, по необходимости, исходные данные этапа или перечень проводимых работ. Однако итерационная модель может оказаться неэффективной, если в ходе работ над системой могут измениться начальные требования к ней.

Чаще всего каскадные модели используются при выполнении больших проектов, в которых задействовано несколько больших команд разработчиков.

V-образная модель жизненного цикла

V-образная модель является разновидностью каскадной модели и имеет такую же последовательную структуру. Как и в каскадной модели, здесь каждая последующая фаза начинается по окончании предыдущей фазы, однако данная модель базируется на комплексном подходе к определению фаз процесса разработки. В этой модели особое значение придается действиям, направленным на верификацию и аттестацию продукта. Она предполагает, что тестирование продукта обсуждается и планируется на ранних этапах жизненного цикла разработки. План приемочных испытаний разрабатывается на этапе планирования, а программа комплексного испытания системы - на фазах анализа и разработки проекта. В данной модели подчеркнуты взаимосвязи, существующие между аналитическими фазами и фазами проектирования, которые предшествуют кодированию, после которого следуют фазы тестирования.

Фазы V-образной модели [3]:

- планирования проекта и требований – здесь определяются системные требования и то, как будут распределены ресурсы с целью их соответствия поставленным требованиям;
- анализ требований к продукту и его спецификации - анализ существующих требований, завершаемый созданием полной спецификации программной системы;
- разработка архитектурного проекта на высшем уровне - определяет, каким образом функции программного обеспечения должны выполняться при реализации проекта;
- детализированная разработка проекта – здесь определяются и документально обосновываются алгоритмы для каждой задачи, поставленной на фазе проектирования архитектуры;
- разработка программного кода – на данной фазе выполняется преобразование алгоритмов, определенных на этапе детализованного проектирования, в готовое программное обеспечение;
- модульное тестирование заключается в проверке каждого программного модуля на наличие ошибок;
- интеграция и тестирование состоит в установке взаимосвязей между группами ранее поэлементно испытанных модулей с целью подтверждение того, что эти группы работают

также корректно, как и модули, испытанные независимо на этапе поэлементного тестирования;

- системное и приемочное тестирование предполагает проверку функционирования программной системы в целом в аппаратной среде, определенной спецификацией требований;
- производство, эксплуатация и сопровождение – фаза, на которой программное обеспечение вводится в эксплуатацию и осуществляется его модернизация и внесение поправок;
- приемочные испытания - тестирование функциональных возможностей системы на соответствие исходным требованиям.

С целью преодоления недостатков V-образной модели ее можно модифицировать, включив итерационные циклы, необходимые для возможного внесения изменений в требования по завершении фазы анализа.

Преимущества V-образной модели наиболее отчетливо проявляются в проектах, где вся информация о требованиях к продукту известна заранее, методы реализации и технологии отработаны, персонал всесторонне подготовлен и имеет необходимый опыт.

Модель прототипирования жизненного цикла

На практике при проектировании информационных систем важнейшую роль играет принятие окончательного, однозначного и верного решения о том, что за систему необходимо построить, какие задачи и как именно она должна решать. В приведенных выше моделях эта задача решается чисто аналитически и результаты ее решения в виде детальных технических требований закрепляются документально (например, на уровне технического задания). При этом возможны ситуации, когда заказчик и разработчик могут трактовать отдельные положения технического задания по своему. Поэтому сформулированные требования часто бывают неполными и неточными, и избежать этого можно только путем максимального вовлечения заказчика в процесс создания системы.

В основе модели прототипирования лежит идея построения экспериментальной (прототипной) системы, на модификациях которой итеративно и быстро отрабатываются требования к проектируемому продукту. Такая модель обычно называется структурной эволюционной моделью быстрого прототипирования.

Под эволюционным ускоренным прототипом понимают легко создаваемую, расширяемую и модифицируемую рабочую модель предполагаемой системы, которая дает достаточно полное представление о ключевых компонентах и функциях системы до ее непосредственной реализации.

Заказчики (конечные пользователи) системы анализируют ускоренный прототип и выдают свои замечания до тех пор, пока прототип не будет удовлетворять всем их требованиям. После завершения процесса определения требований (путем разработки ускоренных прототипов) формируется детальный проект системы, а на основе прототипа создается конечный рабочий продукт.

Преимущества использования структурной эволюционной модели быстрого прототипирования проявляются наиболее отчетливо при разработке таких информационных систем, для которых нет аналогов, либо, например, когда:

- предъявляемые требования не известны заранее, неудачно сформулированы или набор требований не фиксирован;
- целесообразно некоторые компоненты прототипировать, а некоторые разрабатывать традиционными методами (напр., в комбинации с каскадной моделью, когда на начальном этапе проекта используется прототипирование, а на последнем – каскадная модель);
- существует реальный риск создания системы, которая не имеет никакой ценности для заказчика;
- алгоритмы или интерфейсы усложнены.

Модель быстрой разработки приложений (Rapid Application Development)

Модель RAD, предложенная фирмой IBM, предусматривает активную работу пользователя (заказчика) на всех стадиях жизненного цикла от определения требований до ввода системы в эксплуатацию. Продуктивность этой работы и существенное сокращение длительности разработки обеспечиваются благодаря использованию специальных инструментальных средств, реализующих быструю разработку приложений[1]. Процесс разработки заключается в итеративной генерации прототипов, анализ которых осуществляется совместно с заказчиком, причем на длительность процесса разработки накладывается жесткое ограничение[2].

Этап планирования требований предусматривает совместное проведение структурного анализа решаемых системой задач и формулирование на этой основе набора требований к проектируемой системе[3].

Этап создания пользовательского описания предполагает совместную разработку приложений[4] для максимально полного сбора пользовательской информации, причем используются инструментальные средства, обеспечивающие быструю разработку приложений.

Этап конструирования включает реализацию системы, ее тестирование и поставку заказчику в оговоренные сроки.

Ввод системы в эксплуатацию предусматривает установку системы, проведение заказчиком приемочных испытаний и обучение пользователей.

При использовании данной модели максимум трудозатрат пользователя приходится на этап разработки пользовательского описания продукта.

Модель быстрой разработки приложений наилучшим образом может быть использована при создании систем информационного назначения, масштабируемых систем и систем, легко поддающихся моделированию, а также в случаях, когда требования к проектируемой системе хорошо известны. Хорошие результаты могут быть получены в небольших системах, предназначенных для концептуальной проверки поставленных задач и при невысокой степени технических рисков.

Инкрементная модель жизненного цикла

При использовании инкрементной модели, называемой также запланированным усовершенствованием продукта, разработка осуществляется как пошаговый процесс последовательного приближения системы к ее итоговому виду. В данной модели уменьшение затрат, понесенных до момента создания полной версии, достигается за счет поэтапного наращивания функциональных возможностей или производительности системы. При этом каждая последующая версия системы обогащает предыдущую определенными функциональными возможностями до тех пор, пока не будут реализованы все требования к системе. Ускорению процесса разработки способствует применяемый принцип компоновки из стандартных блоков.

Инкрементная модель построена по принципу каскадной модели с перекрытиями и представляет собой комбинацию элементов последовательной линейной модели и модели прототипирования, благодаря чему появляется возможность реализации и анализа функциональных возможностей системы на ранних стадиях разработки.

Это требует наличия полного набора требований, которые могут быть сформированы либо в результате последовательной реализации небольших локальных проектов, либо путем формулирования общих целей и их последующего уточнения.

Реализации инкрементов предшествует планирование, анализ, разработка проекта разрабатываемой системы и определение инкрементов.

Каждый инкремент включает полный цикл работ от детализации проекта до эксплуатации и сопровождения продукта. Процесс выполнения инкрементов завершается при удовлетворении требований заказчика к проектируемой системе.

---

[1] Например, Visual C++, Oracle Designer, Visual Basic 6, и др.
[2] Как правило не более 60 дней.
[3] Этот метод называют методом совместного планирования требований (Joint Requirements Planning, JRP).
[4] Joint Application Design, JAD.

Инкрементная модель хорошо применима в следующих ситуациях:

- если необходимо оперативно поставить на рынок информационный продукт, обладающий ограниченным набором основных функциональных свойств, и при этом основные требования к продукту не могут быть сформулированы сразу, поскольку их выявление возможно только через определенный период времени;
- когда необходимо избежать резких переходов пользователя на принципиально новую технологию выполнения работ и необходима постепенная адаптация пользователей к новым условиям:
- если проект можно выполнять в течение достаточно длительного периода времени (как правило, один год), и при этом желательно распределить риски по отдельным этапам выполнения проекта (инкрементам);
- когда получение результативных данных возможно через регулярные промежутки времени, и это укладывается в отдельные инкременты.

Спиральная модель жизненного цикла

Эта модель, предложенная Барри Боэмом и опубликованная в журнале IEEE Computer в 1988 году, ориентирована на разрешение проблем, связанных с внесением изменений в требования заказчика. Спиральная модель жизненного цикла относится к моделям итерационного типа, где на каждом витке спирали прорабатываются и конкретизируются все аспекты системы и в результате получается окончательный вариант, отвечающий всем требованиям заказчика. Этапы, представленные на каждом витке спирали, идентичны и подобны этапам, принятым в каскадной модели.

В модели выделено четыре квадранта:

1. определение целей, альтернативных вариантов и ограничений;
2. оценка альтернативных вариантов, идентификация и разрешение рисков;
3. разработка продукта следующего уровня;
4. планирование следующей фазы.

В процессе разработки повышенное внимание уделяется анализу и проектированию, причем принимаемые решения апробируются путем создания прототипов. На завершающем шаге каждого витка спирали создается версия, на которой уточняются все требуемые условия и характеристики проекта и планируется следующая итерация. Таким образом, путем последовательных приближений приходят к окончательному варианту, в результате реализации которого появляется конечный продукт.

Спиральная модель наилучшим образом подходит для случаев, когда:

- проект имеет среднюю или высокую степень риска;
- пользователи не уверены в своих потребностях или требования слишком сложные;
- предполагается создание новой серии продуктов или когда в процессе разработки используются новые технологии;
- проект сложен, объемен, требуется большой объем вычислений;
- организация обладает навыками, позволяющими эффективно адаптировать модель;
- невозможно заранее выделить все необходимые для проекта средства, и отсутствует надежная финансовая поддержка.

Выбор модели жизненного цикла

Как было показано, каждая из моделей жизненного цикла имеет свои достоинства, недостатки и области применения. Поэтому в каждом конкретном случае выбирать модель жизненного цикла следует весьма осмотрительно и желательно с участием группы разработчиков, которой поручено выполнение проекта. Выбор модели жизненного цикла разработки можно осуществлять исходя из результатов анализа следующих характеристик [3]:

- группы разработчиков проекта;
- предъявляемых к проектируемой системе требований;
- коллектива предполагаемых пользователей (заказчиков);
- вероятных рисков и типа проекта.

Успех проекта в решающей степени определяется человеческим фактором. Поэтому формирование группы разработчиков (или ядра группы разработчиков) желательно осуществить еще на «нулевом» этапе проекта исходя из профессиональных и личных качеств каждого члена группы. При этом выбор модели жизненного цикла разработки во многом будет зависеть от того, каковы характеристики группы разработчиков.

Применимость той или иной модели жизненного цикла существенно зависит от характера требований, предъявляемых к проектируемой системе (табл. 1[5]).

Таблица 1

Применимость различных моделей ЖЦ в зависимости от характеристик набора требований к проектируемой системе

Характеристика	Модель					
	Каскадная	**V-образная**	**Прототипная**	**Спиральная**	**RAD**	**Инкрементная**
Являются ли требования легко определимыми и/или хорошо известными?	Да	Да	Нет	Нет	Да	Нет
Могут ли быть требования заранее определены?	Да	Да	Нет	Нет	Да	Да
Часто ли будут изменяться требования?	Нет	Нет	Да	Да	Нет	Нет
Нужно ли демонстрировать требования с целью их определения?	Нет	Нет	Да	Да	Да	Нет
Требуется ли для демонстрации возможностей проверка концепции?	Нет	Нет	Да	Да	Нет	Нет
Будут ли требования отражать сложность системы?	Нет	Нет	Да	Да	Нет	Да
Отражают ли требования на раннем этапе функциональные свойства системы?	Нет	Нет	Да	Да	Да	Да

Например, если требования не могут быть заранее определены, а в ходе работ будут часто изменяться, то наиболее подходящими являются модель прототипирования и спиральная модель.

Поскольку при использовании ряда моделей успех реализации проекта зависит от степени слаженности работы единой команды разработчиков и пользователей, то уже на начальных этапах работы, выбирая модель жизненного цикла, следует получить полное представление о характеристиках коллектива пользователей как комплексном факторе, влияющем на выбор модели.

Важнейшую роль при выборе модели жизненного цикла играют тип предполагаемого проекта и риски, связанные с выполнением проекта.

Из приведенных рассуждений следует, процедура выбора модели жизненного цикла должна осуществляться на основе рассмотрения не отдельных критериев, а их комплекса. При этом существенную роль будут играть особенности реальной ситуации.

Для адекватной оценки данной ситуации возникает необходимость в привлечении экспертов и последующей обработке полученных экспертных оценок, и, соответственно, представляется актуальной задача создания соответствующей информационной системы, которая бы в комплексе охватывала вопросы применимости различных моделей жизненного цикла в конкретных ситуациях, охарактеризованных экспертами. Работа этой информационной системы должна включать обоснование выбора критериев, подготовку опросных листов, выбор моделей и алгоритмов обра-

[5] Аналогичные таблицы строятся для характеристик типа проекта, группы разработчиков, вероятных рисков и т.п.

ботки экспертных данных, преобразование качественных характеристик в количественные, и т.п. Возможная структурная схема предлагаемой информационной системы приведена на рис. 1.

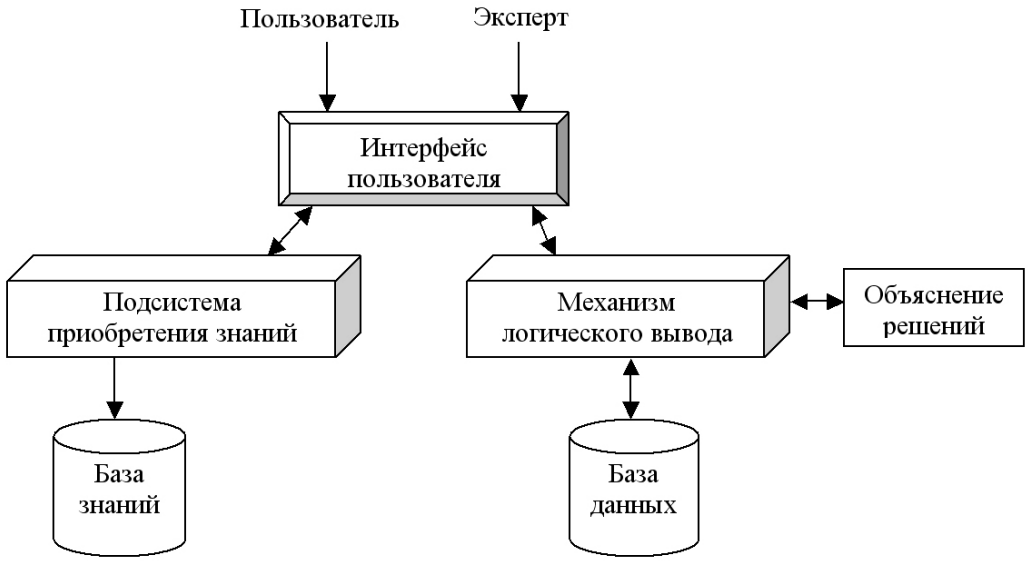

Рис. 1. Структурная схема информационной системы

База данных аккумулирует информацию, касающуюся экспертных оценок и характеристик выбранных групп критериев.

В основу системы представления знаний может быть положена продукционная модель, которая в силу своей простоты получила наиболее широкое распространение. В этой модели знания представляются в виде совокупности правил типа «ЕСЛИ-ТО». Системы обработки знаний, использующие такое представление, получили название продукционных систем. В состав экспертных систем продукционного типа входят база правил (знаний), рабочая память и интерпретатор правил (решатель), реализующий определенный механизм логического вывода.

Пополнение базы знаний осуществляется через подсистему приобретения знаний, а механизм логического вывода обеспечивает как выдачу рекомендаций по выбору той или иной модели жизненного цикла, так и построение цепочек обратного логического вывода с целью объяснения логики принятия того или иного решения.

В настоящее время такая система находится в стадии разработки. По мнению авторов, ее реализация позволит осуществлять обоснованный выбор модели жизненного цикла, наиболее адекватно отвечающей реальной ситуации.

По завершении выбора наиболее подходящей модели жизненного цикла необходимо привести данную модель в соответствие с потребностями проекта, т.е. адаптировать модель к конкретным условиям (этот этап работ также называют подгонкой модели жизненного цикла). На этом этапе осуществляется анализ предполагаемых видов работ (разработка, тестирование, сопровождение, и др.), определяются фазы жизненного цикла и операционный состав каждой фазы, внутренние и внешние производимые продукты и их содержимое, устанавливается степень соответствия самого проекта и его жизненного цикла внутренней и внешней нормативной базе, а также дается оценка предполагаемой эффективности выбранной и адаптированной модели жизненного цикла.

Литература

1. ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств.
2. Вендров А.М. Обзор средств проектирования информационных систем [Электронный ресурс] // http://www.citforum.ru/database/kbd96/42.shtml.
3. Шафер Дональд, Ф., Фатрелл Роберт, Т., Шафер Линда, И. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1136 с.