Практическое занятие № 1

Сравнительный анализ методологий проектирования

Общие положения

На сегодняшний день существует множество разнообразных методологий построения процесса разработки ПО, и у каждой из них есть свои плюсы и минусы, области применения, в которых определенные из них наиболее эффективны. Все эти методологии преследуют своей первой целью улучшение производственного процесса, который позволил бы наиболее эффективно и качественно производить программные продукты.

Модели (или методологии) процессов разработки программного продукта принято классифицировать по «весу» – количеству формализованных процессов (большинство процессов или только основные) и детальности их регламентации. Чем больше процессов документировано, чем более детально они описаны, тем больше «вес» модели.

Модель процессов MSF

Microsoft Solutions Framework (MSF) – это гибкая и достаточно легковесная модель, построенная на основе итеративной разработки.

Модель процессов MSF описывает общие подходы к разработке и внедрению ПО. Благодаря своей гибкости она может быть применена при разработке весьма широкого круга проектов. Эта модель сочетает в себе свойства двух стандартных моделей: каскадной и спиральной. Модель процессов MSF охватывает весь жизненный цикл создания решения, начиная с его отправной точки и заканчивая внедрением.

Модель процессов MSF опирается на следующие базовые принципы:

- подход, основанный на фазах и вехах;
- итеративный подход;
- интегрированный подход к созданию и внедрению решений.

Модель процессов включает такие основные фазы процесса разработки как:

- выработка концепции;
- планирование;
- разработка;
- стабилизация;
- внедрение.

Процесс MSF ориентирован на «вехи» (milestones) – ключевые точки проекта, характеризующие достижение в его рамках какого-либо существенного (промежуточного либо конечного) результата, причем этот результат может быть оценен и проанализирован. Критерии оценки результата должны быть сформулированы еще до начала проекта.

Для декомпозиции больших этапов работы может быть введено также большое количество промежуточных вех.

Модель процессов MSF учитывает постоянные изменения проектных требований. Она исходит из того, что разработка решения должна состоять из коротких циклов, реализующих поступательное движение от начальных версий системы к ее окончательному виду.

В рамках MSF программный код, документация, дизайн, планы и другие рабочие материалы создаются, как правило, итеративными методами. MSF рекомендует начинать разработку решения с построения, тестирования и внедрения его базовой функциональности. Затем к решению добавляются все новые и новые возможности, т. е. с каждой новой версией эволюционирует функциональность решения. Для малых проектов может быть достаточным выпуск одной версии.

Итеративный подход к процессу разработки требует использования гибкого способа ведения документации. Документация должна изменяться по мере реализации проекта вместе с изменениями требований к конечному продукту. В рамках MSF предлагается ряд шаблонов стандартных документов для каждой стадии разработки продукта, которые могут быть использованы для планирования и контроля процесса разработки.

Решение не представляет ценности, пока оно не внедрено. Именно по этой причине модель процессов MSF содержит весь жизненный цикл создания решения, включая его внедрение, вплоть до момента, когда решение начинает давать отдачу.

ТЕХНОЛОГИЯ XP

Технология XP (англ. Extreme Programming) базируется на спиральной модели жизненного цикла. Авторами технологии являются К. Бек, У. Каннингем, М. Фаулер. Название технологии связано со стремлением авторов поднять существующие методы разработки ИС (и программного обеспечения в целом) на новый, «экстремальный» уровень.

Для оценки проектов с точки зрения применимости XP применяются два показателя – критичность и масштаб. Критичность определяется последствиями, вызываемыми дефектами ПО, и может иметь один из четырех уровней:

1. C – дефекты вызывают потерю удобства.
2. D – дефекты вызывают потерю возместимых ресурсов.
3. E – дефекты вызывают потерю невозместимых ресурсов.
4. L – дефекты могут создавать угрозу для человеческой жизни.

Масштаб определяется количеством разработчиков, участвующих в проекте:

- от 1 до 6 человек – малый масштаб;
- от 6 до 20 человек – средний масштаб;
- свыше 20 человек – большой масштаб.

По оценке специалиста по разработке ПО А. Коберна, XP применима в проектах малого и среднего масштаба с низкой критичностью (C или D).

Разработка в XP ведется небольшими трехнедельными итерациями, в течение которых уточняются и реализуются требования к системе. При разработке применяется *рефакторинг* – методика улучшения кода без изменения его функциональности. Программный код в процессе работы над проектом неоднократно переделывается, в том числе и на поздних стадиях проекта.

Для того чтобы эти переделки не привели к неработоспособности системы, используется методика TDD (*Test-Driven Development* – разработка через тестирование). Технология XP предполагает написание автоматических тестов – специальных программ, написанных для тестирования других программ. Ручная прогонка тестов здесь невозможна, т. к. количество тестов слишком велико. Тесты пишутся еще до того, как начинается создание системы, поэтому риск получения неработоспособной версии из-за постоянно вносимых изменений существенно снижается – любую новую версию тут же можно протестировать.

В процессе создания системы применяются такие характерные для XP методы, как парное программирование, непрерывная интеграция, упрощенное проектирование.

Парное программирование предполагает, что программы создается парами программистов, работающих за одним компьютером. Один из них пишет непосредственно текст программы, другой оценивает его работу, благодаря чему становится возможной постоянная проверка программного кода. В течение работы над проектом пары не фиксируются: это делается с той целью, чтобы каждый программист в команде имел хорошее представление обо всей системе. Повышение эффективности при работе парой программистов подтверждено специальными исследованиями.

Непрерывная интеграция (сборка) системы позволяет поддерживать ее целостность в течение всего процесса разработки. В традиционных методиках интеграция выполняется в самом конце работы над продуктом, когда все составные части разрабатываемой системы полностью готовы. Интеграционные проблемы обладают способностью накапливаться и наслаиваться друг на друга, что может даже привести к провалу проекта. В XP интеграция системы выполняется несколько раз в день, после того, как все модули прошли положенные для них тесты. Это позволяет выявить проблемы интеграции на возможно более ранней стадии разработки и заблаговременно принять необходимые шаги к их преодолению.

Упрощенное проектирование применяется в XP из-за того, что в процессе работы требования к системе могут неоднократно меняться, что снижает ценность проекта, выполненного целиком в самом начале разработки. Для XP характерно непрерывное проектирование,

выполняемое в течение всего времени работы над проектом. Проектирование должно выполняться небольшими этапами, с учетом постоянно изменяющихся требований. В каждый момент времени следует использовать наиболее простые решения, которые подходят для решения текущей задачи, и менять его по мере того, как условия задачи меняются. Согласно К. Беку, упрощенное проектирование обеспечивает корректное выполнение всех тестов, не порождает дублирующего кода, включает наименьшее количество классов и методов, ясно выражает цель программиста.

Помимо перечисленного, все члены команды в ходе работы должны соблюдать *общие требования стандартов программирования*, что существенно облегчает рефакторинг и снижает риски проекта, связанные с текучкой кадров. В идеале соблюдение стандартов программирования должно полностью исключить индивидуальные черты стиля разработки – программный продукт должен выглядеть как результат работы одного человека.

Коллективное владение означает, что каждый член команды несет ответственность за весь исходный код. Каждый вправе вносить изменения в любой участок программы. Сопутствующие риски от вносимых изменений устраняются мощной системой тестирования. Однако это не порождает безответственности, поскольку существует требование, согласно которому каждый программист должен сам исправить сделанные им ошибки. Важное преимущество коллективного владения кодом состоит в том, что оно ускоряет процесс разработки, поскольку при необходимости любой программист может оперативно внести изменения в любую часть кода.

Существенным считается и наличие *метафоры системы* – простой аналогии, понятной всем участникам проекта, которая с достаточной точностью описывает функционирование и внутреннюю структуру ИС.

Каждая итерация обычно длится две-три недели и выглядит как программный проект в миниатюре, включая все этапы жизненного цикла системы:

- планирование;
- анализ требований;
- проектирование;
- программирование;
- тестирование;
- документирование.

Хотя отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, подразумевается, что гибкий программный проект готов к работе в конце каждой итерации. По окончании каждой итерации команда выполняет переоценку приоритетов разработки.

МЕТОДОЛОГИЯ RUP

Rational Unified Process (RUP) – методология разработки программного обеспечения, созданная компанией Rational Software, с 2003 года входящей в корпорацию IBM. Методология RUP основана на спиральной модели жизненного цикла ИС. В качестве языка моделирования в RUP используется язык Unified Modelling Language (UML).

Наибольшее внимание RUP уделяет начальным стадиям разработки проекта – анализу и моделированию, – что призвано снизить риски проекта за счет раннего обнаружения возможных ошибок. Последовательный выпуск версий организован таким образом, чтобы наиболее существенные риски устранялись в первую очередь.

Методология RUP широко использует так называемые *прецеденты*, или сценарии использования – описание последовательностей действий, которые может осуществлять система, взаимодействуя с внешними действующими факторами. Прецеденты создаются при помощи UML и включают варианты как правильных, так и ошибочных последовательностей (исключений). Прецеденты служат для документирования требований заказчика к проектируемой информационной системе. Прецедент описывает целостный фрагмент поведения системы в виде последовательности сообщений, которыми система обменивается с действующими лицами.

Другие существенные черты методологии RUP:

- заранее предусматриваются изменения в требованиях и проектных решениях в течение всего процесса разработки,
- постоянное обеспечение качества на всех этапах разработки ИС,

 компонентная архитектура используется начиная с ранних стадий проекта.

Процессы и стадии RUP

RUP использует итеративную модель разработки. В конце каждой итерации (продолжительностью в несколько недель) команда разработчиков должна получить функционирующую версию конечного продукта, позволяющую достичь запланированных на данную итерацию целей. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта.

Полный жизненный цикл разработки ИС состоит из четырех фаз, описываемых ниже. Каждая фаза может включать в себя одну или несколько итераций процесса создания системы.

Начальная стадия

В фазе начальной стадии:

 формируется единая точка зрения на проект у заказчиков и разработчиков, а также определяются границы проекта,

 создается экономическое обоснование разработки,

 определяются основные требования, ограничения и функциональность системы,

 создается базовая версия модели прецедентов,

 оцениваются риски.

Уточнение

В фазе «Уточнение» производится анализ исходных данных для проектирования и выбор архитектуры ИС. Фаза включает в себя:

 анализ требований заказчика, включая детальное описание для большинства прецедентов,

 проектирование архитектуры ИС, спецификация функций и пользовательского интерфейса,

 планирование работ по проекту и всех необходимых ресурсов.

Построение

В фазе «Построение» происходит итеративная реализация требуемых функций ИС. Это основная фаза проектирования и создания программного кода. Фаза завершается выпуском бета-версии системы.

Внедрение

В фазе «Внедрение» финальная версия системы внедряется у ее заказчика. Фаза включает проведение испытаний системы, обучение пользователей, а также оценку качества ИС. В том случае, если качество системы не соответствует требованиям заказчика, фаза «Внедрение» выполняется повторно. Выполнение всех требований и достижение целей проекта означает завершение полного цикла разработки.

МЕТОД DSDM

Метод разработки динамических систем (англ. Dynamic Systems Development Method, DSDM) основан на концепции быстрой разработки приложений (RAD). Метод DSDM – это итеративный и инкрементный подход разработки программного обеспечения, который придает особое значение продолжительному участию в процессе заказчика системы.

Метод DSDM был разработан в Великобритании в 1990-х Консорциумом DSDM. Консорциум DSDM – это ассоциация разработчиков и экспертов в области программного обеспечения, созданная с целью использования лучшего практического опыта участников ассоциации. Все, кто распространяет DSDM, должны быть членами этого некоммерческого консорциума.

Последняя версия DSDM называется DSDM Atern. Предыдущая версия DSDM 4.2, выпущенная в мае 2003 года, все еще действует. Расширенная версия содержит руководство по тому, как использовать DSDM совместно с XP (eXtreme Programming).

Цель метода DSDM – соблюдение сроков и бюджета проекта при допущении изменений в требованиях к системе во время ее разработки. Как представитель RAD-технологии DSDM фокусируется на проектах информационных систем, характеризующихся сжатыми сроками и

бюджетами. DSDM входит в семейство гибкой методологии разработки программного обеспечения, а также может применяться для разработок, не входящих в сферу информационных технологий.

Существует возможность включения в DSDM частей других методик, таких как Rational Unified Process (RUP) или XP. Другой гибкий метод, похожий на DSDM по процессу и концепции, – Scrum.

DSDM содержит указания на типичные ошибки проектов информационных систем, такие как превышение бюджета, несоблюдение сроков сдачи, недостаточное вовлечение пользователей и менеджеров организации-заказчика в работу над проектом.

Жизненный цикл проекта

Согласно DSDM, жизненный цикл ИС состоит из трех последовательных стадий:

- предпроектной стадии;
- стадии проекта;
- постпроектной стадии.

Предпроектная стадия

На этой стадии определяются риски проекта, происходит выделение средств и определение проектной команды. Решение задач на этой стадии поможет избежать проблем на более поздних стадиях проекта.

Стадия проекта

Это самая детально разработанная стадия DSDM. Она состоит из пяти этапов, которые формируют итеративный, инкрементный подход к разработке информационных систем:

1. Исследование реализуемости.
2. Исследование экономической целесообразности.
3. Создание функциональной модели.
4. Проектирование и разработка.
5. Этап реализации.

Первые два этапа выполняются последовательно и дополняют друг друга. После их завершения происходит итеративная и инкрементная разработка системы на основе этапов 3–5, выполняемых циклически, вплоть до выпуска готового продукта.

Постпроектная стадия

На этой стадии обеспечивается внедрение и эксплуатация системы. Это достигается за счет поддержания проекта, его улучшения и исправления ошибок согласно принципам DSDM. Поддержка проекта осуществляется как продолжение разработки, основанной на итеративной и инкрементной природе DSDM. Вместо того чтобы закончить проект за один цикл, обычно возвращаются к предыдущим стадиям или этапам, чтобы улучшить продукт.

МЕТОД SCRUM

Методология Scrum устанавливает правила управления процессом разработки и позволяет использовать уже существующие практики кодирования, корректируя требования или внося тактические изменения. Использование этой методологии дает возможность выявлять и устранять отклонения от желаемого результата на более ранних этапах разработки программного продукта.

В 1986 японские специалисты Hirotaka Takeuchi и Ikujiro Nonaka опубликовали сообщение о новом подходе к разработке новых сервисов и продуктов (не обязательно программных). Основу подхода составляла сплоченная работа небольшой универсальной команды, которая разрабатывает проект на всех фазах. Приводилась аналогия из регби, где вся команда двигается к воротам противника как единое целое, передавая (пасуя) мяч своим игрокам как вперед, так и назад. Джеф Сазерленд (Jeff Sutherland) использовал эту работу при создании методологии для корпорации Easel в 1993 году, которую по ана-логии и назвал Scrum (термин из регби, означающий – схватка), а Кен Швабер (Ken Schwaber) формализовал процесс для использования в индустрии разработки ПО, представив в 1995 г. описание этого проекта на конференции Object-Oriented Programming Systems, Languages & Applications – одной из самых авторитетных конференций в области программирования. С тех пор метод активно используется в индустрии и многократно описан в литературе. Scrum активно используется также в России. Метод Scrum позволяет гибко разрабатывать проекты небольшими командами (5-9 человек) в ситуации изменяющихся

требований. Девиз Scrum – «анализируй и адаптируй»: анализируй то, что получил, адаптируй то, что есть, к реальной ситуации, а потом анализируй снова. Чем меньше формализма, тем более гибко и эффективно можно работать, – это основной принцип данной методологии. Scrum представляет собой простой каркас, который можно использовать для организации команды и достижения результата более продуктивно и с более высоким качеством за счет анализа сделанной работы и корректировки направления развития между итерациями. Методология позволяет команде выбрать задачи, которые должны быть выполнены, учитывая бизнес-приоритеты и технические возможности, а также решить, как их эффективно реализовать. Это позволяет создать условия, при которых команда работает с удовольствием и максимально продуктивно. Scrum фокусируется на постоянном определении приоритетных задач, основываясь на бизнес-целях, что увеличивает полезность и доходность проекта на его ранних стадиях. Так как при инициации проекта его доходность определить почти невозможно, Scrum предлагает концентрироваться на качестве разработки и к концу каждой итерации иметь промежуточный продукт, который можно использовать, пусть и с минимальными возможностями. Например, результатом итерации может быть каркас сайта, который можно показать на презентации. Методология Scrum ориентирована на то, чтобы оперативно приспосабливаться к изменениям в требованиях, что позволяет команде быстро адаптировать продукт к нуждам заказчика. Такая адаптация достигается за счет получения обратной связи по результатам итерации: имея после каждой итерации продукт, который уже можно использовать, показывать и обсуждать, легче собирать информацию и делать правильные корректировки и изменять приоритеты требований. Например, если каркас сайта показать потенциальным пользователям, то появится много вопросов, на основании которых можно скорректировать то, что уже написано или еще не реализовано, понять что более важно пользователю.

Схема метода Scrum представлена на рисунке:

Выполнение работы

№ 1. Изучите методологии проектирования ИС

№ 2. Дайте характеристику каждой методологии:

1) Назначение;
2) Модель жизненного цикла;
3) Фазы и стадии методологии

№ 3. Перечислите преимущества и недостатки разработки ПО по «весу», заполнив таблицу:

Вес модели	*Преимущества*	*Недостатки*
Тяжелые		
Легкие		