Оглавление

**Элементы оглавления не найдены.**

**1.Введение**

Известно, что DSDM используют больше всего в Великобритании, однако с этой методологией следует ознакомиться, так как ее инфраструктура обладает всеми чертами, характерными традиционным методологиям, в то время как основные принципы соответствуют новым гибким методологиям. Мне кажется, что те материалы о DSDM, которые мне удалось изучить, говорят о чрезмерной ориентированности этой методологии на процесс и любви к формальностям.

При написании работы я для себя поставила следующие задачи:

1.Дать общие понятия о методологиях создания программных продуктов и места рассматриваемой методологии в общей классификации.

2.Рассмотреть историю возникновения методологии: предпосылки появления рассматриваемой методологии, основные публикации.

3. Изучить содержание методологии: подробное описание этапов, правил применения, результатов использования, достоинств и недостатков рассматриваемой методологии.

4. Сделать выводы о достоинствах и недостатках рассмотренной методологии.

“Толковый словарь русского языка” С.И. Ожегова определяет понятие “Методология” как “Принципы и способы организации теоретической и практической деятельности. Совокупность методов, применяемых в какой-либо науке”.

Применительно к разработке программного обеспечения это определение можно переформулировать так: “Методология – это принципы и способы организации деятельности проектной группы для создания программного продукта”.

Разберем формулировку на элементы.Во-первых, “программный продукт”. Именно продукт является конечной целью в любой методологии. Отметим также, что в последнее время вместо термина “программный продукт” все чаще используют термин “решение”(“solution”), а компании-разработчики вместо фразы “мы поставляем программные продукты” все чаще говорят “мы поставляем (готовые) решения”.

Во-вторых, “проектная группа”, “команда разработчиков” или просто “команда”. В любом случае за этим скрывается коллектив людей,непосредственно занятых созданием “готового решения”. Именно люди являются точкой приложения любой методологии, поскольку, как уже сказано, в организации деятельности людей и состоит основное назначение методологий.

Методология превращает создание программного продукта в упорядоченный процесс, с помощью которого можно сделать работу программиста более прогнозируемой и эффективной. Для этого создается детальное описание процесса создания системы, особое место в котором занимает планирование

Такие методологии существуют уже давно. Нельзя сказать, что они очень уж эффективны. Чаще всего их обвиняют в бюрократизме - чтобы следовать такой методологии, нужно выполнять так много различных предписаний, что замедляется весь темп работ. Именно поэтому их называют тяжеловесными методологиями, или, согласно термину Джима Хайсмита, - монументальными (методология императивного программирования, методология объектно-ориентированного программирования, методология функционального программирования, методология логического программирования, методология программирования в ограничениях).

За последние годы в противовес этим методологиям появилась группа новых, которые раньше было принято называть облегченными (lightweight). Теперь для них используют другой термин - гибкие (agile) методологии (XP (Extreme Programming, семейство методологий Crystal Алистэра Коуберна,Open Source,Адаптивная разработка (ASD) по Джиму Хайсмиту,SCRUM,Feature Driven Development,Dynamic System Development Method (DSDM)).

Привлекательность новых методологий для многих заключается в отсутствии бюрократизма, присущего монументальным методологиям. Новые методологии представляют собой попытку достичь необходимого компромисса между слишком перегруженным процессом разработки и полным его отсутствием.

По сравнению с монументальными методологиями, в гибких смещены все основные акценты. Самое очевидное различие - меньшая ориентация на документацию, что выражается в меньшем ее объеме для каждой конкретной задачи. Правильнее в данном случае будет говорить об ориентированности на код, то есть основная предпосылка состоит в том, что ключевая часть документации - это исходный код.

Впрочем, я думаю, что не это является главным отличием гибких методологий от монументальных. Отсутствие документации - это следствие куда более существенных различий:

1. Гибкие методологии адаптивны, а не предсказуемы. Для тяжеловесных методологий необходимо детальное планирование большого объема разработок, и такой подход работает - однако до тех пор, пока не начнутся изменения. Следовательно, для этих методологий сопротивляться всяким изменениям совершенно естественно. Гибкие же методологии, напротив, изменения приветствуют. В отличие от тяжеловесных, они были задуманы как процессы, которые адаптируют изменения и только выигрывают от них, даже в том случае, когда изменения происходят в них самих.
2. Гибкие методологии ориентированы на человека, а не на процесс. В них ясно заявлено о необходимости учитывать в работе природные качества человеческой натуры, а не действовать им наперекор. Кроме этого в гибких методологиях особо подчеркивается, что работа по созданию программных продуктов должна приносить удовольствие.

**2.История возникновения методологии.**

В начале 1990-х в индустрии информационных технологий стал распространятся новый термин - быстрая разработка приложений (Rapid Application Development, RAD).. На рынок начали выходить новые инструменты для создания приложений, например PowerBuilder. Они позволили разработчикам проще делиться планируемыми разработками с покупателями - появилось прототипирование и началось разрушение классических, последовательностных (каскадных) методов разработки.

Тем не менее, новое движение RAD было очень неструктурированным: не существовало согласованного описания этого метода и у многих организаций были созданы собственные описания и подходы к нему. Множество крупных корпораций были заинтересованы в перспективах, предоставляемых методом, но они также были обеспокоены тем, чтобы не понизился уровень качества их продукции в конечном результате.

Консорциум DSDM был образован в 1994 году в Лондоне. Были представлены крупные организации, такие как British Airways, American Express, Oracle and Logica.

На этом собрании было решено, что Дженнифер Степлтон, тогда представлявшая компанию Logica, разработает архитектуру комлексного, ориентированного на пользователя метода с хорошим контролем качества для итеративной и инкрементной разработки.Несмотря на то, что многие члены Консорциума были прямыми конкурентами, они свободно делились тем, как они решают возникающие проблемы, описание метода становилось всё более и более полным.Дженифер Стэйплтон (Jennifer Stapleton) написала книгу “DSDM, Dynamic Systems Development Method: The Method in Practice”, где можно найти краткий обзор этой методологии.

Версия 1 была сформирована в декабре 1994 года и опубликована в феврале 1995 года. Результатом стал универсальный метод, охватывающий людей, процессы и инструменты. Он сформировался на основе опыта организаций, различных по роду своей деятельности и размерам.

Метод разработки динамических систем (Dynamic Systems Development Method, DSDM) - это главным образом методика разработки программного обеспечения, основанная на концепции быстрой разработки приложений (Rapid Application Development, RAD).

В 2007 году DSDM стал основным подходом к управлению проектом и разработки приложений. DSDM - это итеративный и инкрементный подход, который придаёт особое значение продолжительному участию в процессе пользователя/потребителя.

Цель метода - сдать готовый проект вовремя и уложиться в бюджет, но в то же время регулируя изменения требований к проекту во время его разработки. DSDM входит в семейство гибкой методологии разработки программного обеспечения, а также разработок не входящих в сферу информационных технологий.

Самая последняя версия DSDM называется DSDM Atern. Название Atern - это сокращение от Arctic Tern (пер. Полярная крачка). Полярная крачка - птица, которая может путешествовать на большие расстояния. Она символизирует множество аспектов метода, например определение приоритета или кооперирование, которые являются естественным способом ведения рабочего процесса.

Предыдущая версия DSDM (выпущенная в мае 2003 года), которая всё ещё действует и широко используется, - это DSDM 4.2, являющаяся немного расширенной версией DSDM 4. Расширенная версия содержит руководство по тому, как использовать DSDM совместно с Экстремальным программированием (Extreme Programming).

**3.Содержание методологии**

**3 .1.Обзор структуры методологии**

Вся система основывается на девяти принципах, которые обсуждаются более подробно позже, но это полезно, чтобы перечислить их здесь. Первые четыре определяют основы, на которых строится DSDM и остальные пять обеспечивают принципы, которыми руководствуется данная структура.

1. Активное участие пользователя является обязательным условием.

2. Команды DSDM должны быть уполномочены принимать решения.

3. Основное внимание уделяется частые поставки продукции.

4. Приспособленность для деловых целей является важнейшим критерием для принятия результатов.

5. Итеративное и инкрементное развитие необходимо, чтобы сойтись на точном бизнес-решении.

6. Все изменения в ходе развития обратимы.

7. Требования скомпонованные на высоком уровне.

8. Тестирование интегрировано в жизненный цикл.

9. Сотрудничества и подход на основе сотрудничества между всеми заинтересованными сторонами.

Итеративный и инкрементный процесс воплощен в пятом принципе состоит из пяти фаз развития. (Есть два развития стадий: предпроектная подготовка, которая гарантирует, что проект создан на прочной основе, и после проекта, который включает в себя учета поставленного решения оперативных.) Первые два этапа развития являются последовательными: обоснование оценки пригодности системы к подходу и предусмотреть первоначальный вид расходов, и т. д., далее следуют бизнес-исследования, которые состоят из бизнеса и технических основ всего проекта. После бизнес исследования, первым итерационным этапом является функциональная модель итерации, в которой бизнес анализируется более детально. Анализ поддерживается благодаря эволюционному прототипированию функциональных возможностей внутри архитектуры системы, которая также определяется на высоком уровне в бизнес-исследованиях. Когда площадь функциональности достаточно хорошо понята, дизайн и итерации построения проектируют компонент системы достаточно высокого качества, который годен для перехода в стадию реализации. Реализация охватывает не только движение системы в производственную среду, но и обучение пользователей. В конце реализации, приращение рассматривается и принимаются бизнес-решения о том, какая дальнейшая работа (если есть) должно быть реализована в последующих инкрементах.

Никакой процесс не обходится без людей, чтобы принять его. Первый принцип гласит, что пользователи решения должны принимать активное участие в процессе развития: их обычный вход и обратная связь являются важными в структуре методологии. DSDM определяет роли для людей, вовлеченных в проект DSDM. Они включают в себя как пользователей, так и состав разработчиков. Например, одна из ролей пользователя- это Visionary (Фантазер). Обычно,это лицо, ответственное за получение проекта в силу своего видения ИТ-поддержки в сфере бизнеса. Ключевая его роль заключается в том, что он является техническим координатором, который в основном системный архитектор и хранитель технического представления . Сочетание этих двух ролей обеспечивает безопасность консультирования и защиту технической базы проекта, но существует еще много ролей, определенных в обеих областях специализации.

Цель DSDM поставить системы в сроки, что невозможно с помощью подхода водопада. Последствия этого таковы, что рабочие процессы должны управляться по-другому и метод, используемый в этих процессах должнен быть отточен вниз, чтобы максимально снизить накладные расходы. Основным инструментом регулирования работа является timebox. Timebox в DSDM-это короткий промежуток времени (несколько дней или несколько недель) в проекте, когда что-то производится для определенных целей качества, что отвечает третий, четвертый, и восьмой принципы. Принимая продукт на основе просмотра, а не деятельностный взгляд на процесс, DSDM позволяет управлению быть направленным на то, что производится, а не на способ производства. Это позволяет гибкий подход к методам, используемых в системе.

Применение шестого принципа “обратимые изменения” означает, что все, что производится, должно быть достаточно хорошо контролируемым, чтобы вернуться в известное состояние, когда продукт оказывается неверным.

Так DSDM контролировать стиль разработки, который часто рассматривается как способ получения неудобной в сопровождении системы. Он держит четко ориентированных на удовлетворение потребностей бизнеса, а не его отношение к ним. Применение пользователя DSDM сосредотачивается на итеративном и инкрементальном подходах, обеспечивающих множество преимуществ. Как было доказано на тысячах проектов, они включают в себя:

1. большие шансы пользователя стать владельцем системы;
2. меньший риск формирования неправильной системы;
3. большее соответствие системы бизнес-требованиям пользователей;
4. лучшее обучения пользователей;
5. легкое внедрение системы

**3 .2 .Роли**

Методология DSDM определяет набор стандартных ролей. Каждая роль имеет свою собственную зону ответственности. Каждому члену команды сопоставляется одна из ролей. Ниже перечислены основные роли, определяемые методологией DSDM.

1. Менеджер проекта

Менеджер проекта (Project Manager) обеспечивает общее руководство проектом.

2. Провидец

Провидец (Visionary) является движущей силой проекта. Провидец следит за соответствием проекта коммерческим целям и задачам. 3. Провидцем часто является топ-менеджер, который инициировал проект. Данная роль предполагает возможность частичной занятости.

3. Чемпион проекта

Чемпион проекта (Project Champion или Executive Sponsor) обладает возможностями и обязанностями по распоряжению ресурсами и фондами,которые необходимы данному проекту. Чемпион проекта несет ответственность за принятие любых решений, связанных с проектом. Данная роль предполагает возможность частичной занятости.

4. Лидер команды

Лидер команды (Team Leader) руководит командой разработчиков и обеспечивает эффективность ее работы.

5. Технический координатор

Технический координатор (Technical Co-ordinator) отвечает за разработку архитектуры продукта. Технический координатор также отвечает общее техническое состояние проекта.

6. Разработчик

Разработчик(Developer) участвует в анализе требований,моделировании и проектировании. Очевидно, что основной обязанностью разработчика является программирование.

7.Тестировщик

Тестировщик (Tester) отвечает техническое тестирование продукта.

8.Представительный пользователь

Представительный пользователь (Ambassador User) представляет пользователей продукта. Представительный пользователь отвечает за то, чтобы разработчики вовремя получали обратную связь со стороны пользователей.

9. Пользователь-консультант

Пользователем-консультантом (Advisor User) может быть любой пользователь, представляющий значительную точку зрения на продукт. Пользователь-консультант привносит в проект знание по некоторому Аспекту использования разрабатываемого продукта. Данная предполагает возможность частичной занятости.

10. Секретарь

Секретарь (Scribe) отвечает за протоколирование всех соглашений и решений, принятых во время семинаров.

11. Посредник

Посредник (Facilitator) отвечает за проведение семинаров. Посредник также отвечает за эффективность коммуникации между всеми членами команды.

**3 .1.Разделение на команды**

Методология DSDM рекомендует создавать команды небольшого размера, до шести человек (без учета руководящих персон вроде провидца и чемпиона проекта). При этом над одним проектом может работать несколько команд. Известны примеры проектов, выполненных по методологии DSDM,в которых участвовало до 150 человек. Команды могут отвечать как за реализацию некоторого набора функциональности (например, отдельная команда может отвечать за реализацию системы базы данных), так и за некоторые виды деятельности (например, в проекте могут быть две команды,отвечающие за разработку, и одна команда, отвечающая за тестирование).

**4. Заключение**

Метод разработки динамических систем (DSDM) основан на концепции быстрой разработки приложений (RAD). Представляет собой итеративный и инкрементный подход, в котором важную роль играет участие пользователя. К достоинствам можно отнести:

* принципы методологии DSDM хорошо сочетаются с основными принципами манифеста гибких технологий разработки программного обеспечения;
* строгий контроль на протяжении жизненного цикла разработки;
* в методологии DSDM высоко ценится индивидуальность всех заинтересованных сторон;
* создание работающего программного обеспечение, способного удовлетворять потребности бизнеса, является в методологии DSDM основным критерием успеха проекта;
* сотрудничество и кооперация всех заинтересованных сторон является в методологии DSDM одним из ключевых способов достижения успеха в проекте;
* методология DSDM также изначально предполагает возможность изменения требований к продукту на любой стадии проекта;
* более того, методология DSDM предполагает тесную интеграцию с другими гибкими методологиями. Например, методология экстремального программирования может быть тесно интегрирована с методологией DSDM в рамках одного проекта. В результате чего появляется возможность использовать все основные преимущества экстремального программирования в области проектирования и реализации системы совместно с основными преимуществами методологии DSDM в области управления требованиями и проектом в целом.

К недостаткам можно отнести:

* неприменимость DSDM к проектам, в которых важна безопасность данных;
* документация является сложной и трудоемкой.

Методология DSDM представляется как  зрелая методология гибкой разработки программного обеспечения. Многие другие гибкие методологии сфокусированы на программировании, в то время как  
методология DSDM сфокусирована на организации процесса производства  
программного обеспечения.

**СПИСОК ЛИТЕРАТУРЫ**