Оптимизация процесса разработки:
объединение подходов к разработке на основе моделей и на основе требований

Энди Гёрд (Andy Guard), менеджер по развитию рынка, IBM Rational Software

Пол Урбан (Paul Urban), менеджер по продукту, IBM Rational Software

Оглавление

На заметку

Возрастающая конкуренция вынуждает компании как можно быстрее приступать к разработке, что не позволяет впоследствии гибко реагировать на изменяющиеся требования заказчиков.

Введение

Современный чрезвычайно конкурентный рынок разработки систем и программного обеспечения характеризуется не только жесточайшей, как никогда ранее, конкуренцией, но и экспоненциальным ростом сложности проектов и клиентских ожиданий от них. При таком давлении быть первыми на рынке становится все сложнее, а цена задержки выхода продуктов на рынок может измеряться сотнями тысяч долларов. Поэтому неудивительно, что менеджеры ищут решение для увеличения продуктивности разработки, позволяющее одновременно гарантировать то, что конечные результаты будут соответствовать требованиям заказчика.

Несмотря на то, что всем интуитивно понятна необходимость привязки разработки к требованиям, на практике управлять изменяющимися требованиями оказывается весьма непросто. Неэффективное управление требованиями приводит к плохому взаимодействию между заказчиком, менеджером проекта и разработчиками. В результате в системе появляются ошибки, которые не будут обнаружены вплоть до окончательной стадии разработки, когда внесение изменений в код наиболее трудоемко и дорого. В процессе написания кода команда разработчиков часто упускает из виду, что их главная цель - это создать продукт, отвечающий потребностям заказчика, и забывает о том, что привязка разработки к требованиям - это не пустая трата времени. Более того, привязка к требованиям является лучшим способом для достижения успеха проекта (заключающегося в своевременной поставке продукта, который удовлетворяет ожиданиям заказчика), позволяющим эффективно и гибко управлять неизбежными изменениями требований, возникающими со стороны заказчика по ходу продвижения проекта.

Важность привязки разработки к требованиям хорошо иллюстрируется простым примером из повседневной жизни, имеющим место при заполнении налоговой декларации. Практически в каждой стране правительство требует, чтобы население заполняло налоговые декларации. В налоговом кодексе сформулированы требования, которые определяют суммы взимаемых налогов, и этими требованиями руководствуются налогоплательщики. Они хотят платить ровно столько, сколько им положено, ни больше и ни меньше. Это позволяет избежать неприятных последствий в виде аудиторских проверок, штрафов и, возможно, даже тюремного заключения. Очевидно, что ответственность за заполнение налоговой декларации достаточно высока и является тем примером из реальной жизни, в котором успешность конечного результата определяется внешними требованиями. Похожая ситуация складывается и с успешностью проекта разработки, содержащего более 500 000 строк кода. В современных условиях системные

На заметку

Для того, чтобы процесс разработки был более гибким, разработчики и другие заинтересованные лица проекта должны более эффективно общаться друг с другом, обмениваясь информацией об изменениях и необходимых доработках.

инженеры и разработчики ПО имеют возможность использовать широкий спектр мощных инструментальных средств, которые позволяют им справляться с трудностями по управлению сложными проектами, содержащими множество требований. В этом им помогают интегрированные решения для управления требованиями и моделирования.

Многие разработчики встраиваемых систем и приложений реального времени уже пришли к осознанию того, что использование методологии разработки на основе моделей (MDD – Model-Driven Development), базирующейся на гибком унифицированном языке моделирования (UML) и языке моделирования систем (SysML), является лучших способом получения конкурентных преимуществ на рынке. Разработчики также признают, что использование полноценных решений для поддержки процесса разработки на основе требований (RDD – Requirement-Driven Development) является лучшим способом, позволяющим гарантировать соответствие разрабатываемых системных требований, моделей и программного кода требованиям заказчика. Несмотря на это, до недавнего времени было не так просто организовать процесс разработки, в котором подходы MDD и RDD были бы тесно интегрированы и усиливали возможности друг друга в одной простой для использования и интуитивно понятной последовательности действий. Далее будет показано, каким образом разработчики, использующие интегрированное решение для поддержки MDD и RDD, могут быстрее разрабатывать качественный программный код и быть уверенными, что их проект продвигается к завершению даже в условиях постоянно изменяющихся требований заказчика. Преимуществом рассматриваемого подхода является улучшение взаимодействия между командой разработчиков и заинтересованными лицами проекта, что позволяет системным инженерам управлять сложностью создаваемых систем и их изменениями на основе единого, легкого для понимания формата. Это, в свою очередь, позволяет разрабатывать более качественные продукты за меньшее время.

Проблема управления требованиями

В одном из номеров журнала “CIO Magazine” было отмечено, что, по мнению аналитиков, более 71% всех провалов проектов по разработке программного обеспечения происходит по причине плохого управления требованиями. Эта причина провалов выделяется как главная и превалирует над причинами, связанными с использованием неправильных технологий, несоблюдением сроков и даже сменой руководства.

На заметку

Текстовые требования не позволяют эффективно доносить информацию до разработчиков, подвержены контекстным ошибкам и неправильному пониманию.

Почему это происходит, несмотря на наличие мощных технологических решений, используемых при разработке? Одна из причин состоит в том, что заинтересованные лица или аналитики часто не имеют ясного понимания требований. Иногда требования плохо формулируются, не проверяются соответствующими тестами в процессе контроля качества. Зачастую разработчики просто упускают некоторые требования из вида или понимают их неправильно. Или, что случается чаще, требования пользователей изменяются, а влияние этих изменений на другие требования, проектные решения и процесс тестирования полностью не учитывается. Вне зависимости от причины, эффект для проекта может быть разрушительным: провал приводит к финансовым потерям, негативно сказывается на репутации компании и выпускаемого продукта.

Соответствие требованиям как ключ к успеху

Есть основания полагать, что перечисленные выше проблемы связаны с механизмами работы человеческого мозга. Многие люди склонны к визуальному мышлению, и для них наиболее эффективный способ понять какую-либо концепцию - это представить её визуально. Несмотря на это, общепринятая практика по определению требований состоит в их представлении в текстовом виде, что оставляет множество «лазеек» для ошибок в интерпретации и понимании контекста.

Пример набора текстовых требований

2 Функциональные требования

2.1 Включение двигателя

2.1.1 Движение автомобиля
2.1.1.1 Движение вперед
Автомобиль должен позволять двигаться вперед со скоростью в диапазоне от 0 до 200 км/ч при движении по стандартной ровной дороге и скорости ветра 0 км/ч, развивая при этом мощность 180 л.с.

2.1.1.2 Движение назад
Автомобиль должен позволять двигаться назад с максимальной скоростью 20 км/ч при движении по стандартной ровной дороге и скорости ветра 0 км/ч, развивая при этом мощность 180 л.с.

На заметку

Визуальное моделирование эффективно при описании назначения системы, но не подходит для описания детальных требований.

2.1.2 Разгон автомобиля
Автомобиль должен разгоняться с места до 100 км/ч за 10 сек. при движении по стандартной ровной дороге и скорости ветра 0 км/ч.
Автомобиль должен разгоняться со скорости 100 км/ч до 150 км/ч с ускорением 5 км/с при движении по стандартной ровной дороге и скорости ветра 0 км/ч.
Автомобиль должен разгоняться со скорости 150 км/ч до 200 км/ч, с ускорением 3 км/с при движении по стандартной ровной дороге и скорости ветра 0 км/ч.

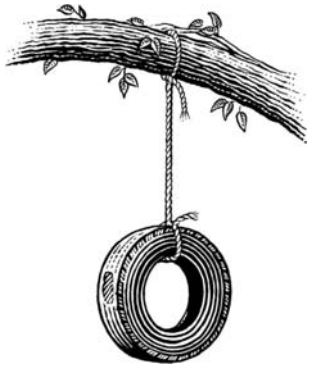
2.2 Управление автомобилем

2.2.1 Включение зажигания автомобиля
В автомобиле должна быть предусмотрена возможность идентифицировать людей, имеющих право включать зажигание и управлять автомобилем.

2.2.2 Управление скоростью
Для управления скоростью в автомобиле должен использоваться ножной механизм. Скорость должна свободно регулироваться в диапазоне от нуля до максимальной. Скорость автомобиля должна контролироваться средствами автоматики.

2.2.3 Торможение автомобиля
При скорости 10 км/ч автомобиль должен останавливаться за 2 сек.
При скорости 30 км/ч автомобиль должен останавливаться за 6 сек.
При скорости 100 км/ч автомобиль должен останавливаться за 30 сек.
При скорости 200 км/ч автомобиль должен останавливаться за 45 сек.

Попробуйте представить себе, как эту же информацию можно было отобразить визуально! Изображение может заменить тысячу слов и предоставить больше информации в более доступном и ясном виде. Хотя и у этого способа имеются недостатки, что можно видеть на рисунке ниже.



Изображение верёвочных качелей заменяет тысячу слов, но не отображает тысячу требований к ним

Несмотря на то, что изображение наглядно отображает цели проектирования для случая верёвочных качелей, оно не может заменить список требований. Например, если в Вашем распоряжении имеется только изображение, Вы ничего не сможете сказать о следующем:

- *Качели должны выдерживать вес до 80 кг.*
- *На качелях должны помещаться двое маленьких детей.*
- *Качели должны быть подвешены не ниже 0,5 м над землей.*
- *Качели не должны иметь возможности раскачиваться больше, чем на 180 градусов.*
- *Упругость веревки для качелей должна равняться…*
- *Качели должны удовлетворять следующим нормам безопасности…*
- *Качели должны удовлетворять следующим дополнительным требованиям…*

На заметку

Продукты IBM Rational Rhapsody и IBM Rational DOORS предоставляют интегрированное решение для поддержки процесса проектирования на основе подходов MDD и RDD, реализующее широкие возможности для визуальных коммуникаций и установления связей с детальными требованиями.

Объединение подходов MDD и RDD

К счастью, системные инженеры и разработчики ПО могут использовать мощные инструментальные средства для поддержки и MDD, и RDD. А в случае с IBM Rational® Rhapsody® и IBM Rational® DOORS® они получают в свое распоряжение тесно интегрированное решение, позволяющее управлять обеими сторонами процесса. Используя интегрированное решение для поддержки MDD и RDD, руководители проектов могут организовать процесс разработки, позволяющий гарантировать, что разработка и требования связаны друг с другом понятным и гибким образом. Это достигается за счет использования среды моделирования для установления связей между уровнями требований и моделей. Например, проектная спецификация может быть связана с моделью ПО, системные требования - с функциональной моделью и моделью архитектуры системы, а требования заинтересованных лиц проекта - с моделью целей (или вариантов использования). Все эти уровни привязаны к назначению системы, являющемуся верхним уровнем в спецификации требований. Благодаря тому, что уровни требований и моделей пересекаются друг с другом, никакой потери требований в процессе разработки не происходит.

Проверка системы на предмет соответствия требованиям становится чрезвычайно трудной, когда производится для большого и сложного проекта, в котором требования часто изменяются. Однако использование интегрированных решений для поддержки MDD и RDD позволяет легко справиться с этой непростой задачей. Поскольку требования пользователей связываются с требованиями к системе, подсистемам и программному обеспечению, то элемент модели, на основе которого создается программный компонент, может быть легко поставлен в соответствие исходному требованию. Этот процесс определяет ясный путь от требований к коду и далее к процессу тестирования, в котором тестовые сценарии могут быть явным образом связаны с любым элементом модели, и позволяет реализовать проверку элементов модели на предмет соответствия требованиям.

Возможность трассировки от требований к коду позволяет проводить проверку их соответствия друг другу. В результате такой проверки заинтересованные лица могут легко убедиться, что результирующая система соответствует требованиям и не содержит элементов, не вытекающих из этих требований. Но самое важное заключается в том, что каждый уровень требований может быть проверен посредством моделирования. Это позволяет на каждом уровне разработки создавать код, соответствующий требованиям. Данный процесс предоставляет возможности для быстрого и полного анализа влияния изменений в требованиях, позволяя понять все последствия этих изменений еще до того, как они будут сделаны, а также гарантирует, что одобренное изменение реализовано полностью.

На заметку

IBM Rational DOORS отслеживает изменения между различными уровнями требований и моделей, упрощая анализ влияния изменений как сверху, так и снизу.

Rational. DOORS

Управление изменениями

Управление изменениями является важной возможностью, которая помогает увеличить эффективность совместного процесса разработки на основе MDD и RDD. Когда какое-либо требование изменяется, DOORS позволяет отслеживать изменения (так называемые «подозрительные связи») до нижних уровней требований. DOORS автоматически помечает изменения в требованиях специальными маркерами, чтобы модель могла быть изменена для разрешения возникшего конфликта с требованиями. DOORS также позволяет определить, на какие элементы в проекте влияет данное изменение. Благодаря этому команда разработчиков может оценить степень влияния каждого изменения и принять обоснованное решение о том, следует ли его вносить, основываясь на информации о том, какое влияние окажет его внесение на весь проект разработки.

Синхронизация моделей и кода позволяет команде разработчиков и заинтересованным лицам проекта быть уверенными, что изменения будут выполнены на всех важных уровнях разработки, что существенно облегчает взаимодействие между ними и дает существенные преимущества. Объединение процессов MDD и RDD способствует улучшению коммуникации, повышению качества выпускаемых продуктов и степени удовлетворенности заказчика, позволяя визуализировать решение, производить его раннюю оценку на предмет соответствия требованиям и улучшать общее понимание проекта и требований.

Используя данный подход, разработчики могут быстро создавать итеративные прототипы системы, что позволяет им уйти от использования морально устаревшего подхода «делаем-переделываем». Общеизвестно, что устранение ошибок на стадиях сбора требований и проектирования обходится намного дешевле, однако решения, способные предоставлять эту возможность, ранее отсутствовали на практике.

Благодаря интеграции подходов MDD и RDD, инженеры и разработчики ПО могут проверять соответствие проектных решений требованиям уже на ранних стадиях процесса разработки, обнаруживая и исправляя ошибки намного раньше, когда их легче и дешевле всего исправить.

Демонстрация прототипов и результатов моделирования заказчикам и всем заинтересованным лицам проекта дает ключ к пониманию того, действительно ли разработка идет в правильном направлении.

В результате разработчики могут быстро реагировать на изменения в требованиях, определяя элементы проекта, на которые они влияют, и получают возможность более аккуратно оценивать необходимые усилия для их реализации.

На заметку

Интегрированный процесс на основе MDD и RDD позволяет улучшить соответствие полученных результатов первоначальным требованиям и повысить продуктивность разработки, способствуя реализации успешных проектов и большей удовлетворенности заказчиков.

Заключение

После рассмотрения процесса можно выделить список рекомендаций.

- *Трассировать требования между различными уровнями на протяжении всего жизненного цикла разработки.*
- *Определять формальные сценарии на основе требований с целью последующей оценки результатов разработки.*
- *Оценивать состояние проекта с участием заинтересованных лиц, используя раннее создание прототипов системы.*
- *Использовать моделирование для того, чтобы убедиться в корректности решения, что позволяет:*
  – предотвращать ошибки,
  – избегать дорогостоящих переработок,
  – повышать качество.
- *Создавать работающие прототипы с поддержкой графических интерфейсов. Это идеальное средство для улучшения коммуникаций, необходимых для получения обратных связей и обмена информацией.*

Совокупный эффект от следования этим рекомендациям чрезвычайно велик и в значительной мере способствует успеху проекта. Моделирование позволяет рано обнаруживать и устранять проблемы, оценивать решение на предмет его соответствия требованиям. Возможности по трассировке между требованиями и моделями позволяют анализировать влияние изменений в требованиях и соответствие моделей требованиям. Возможности по моделированию в процессе проектирования служат лучшему пониманию и декомпозиции требований, при этом, требования определяют границы проекта для проектировщиков и разработчиков ПО. Улучшение коммуникаций способствует более быстрому процессу разработки, в котором результаты в точности соответствуют требованиям, что, в свою очередь, приводит к реализации успешных проектов и большей удовлетворенности заказчиков.