Гибкая методология разработки программного обеспечения

Microsoft® Visual Studio® Team System 2008

Microsoft® Solutions Framework

Гибкая методология разработки программного обеспечения

Microsoft® Solutions Framework

Гибкая методология разработки программного обеспечения

Данное руководство посвящено методологии разработки программного обеспечения Microsoft Solutions Framework (MSF) for Agile Software Development корпорации Mictosoft®. Эта методология описывает подход и организацию работы при создании программных продуктов и, в отличие от более масштабной и абстрактной системы Microsoft Solutions Framework, сразу готова к применению. Руководство предназначено для менеджеров проектов, а также для разработчиков, аналитиков, тестеров, архитекторов и, возможно, других участников команды разработчиков.
Руководство состоит из введения и двух глав.

Подготовлено к печати издательством «Русская Редакция»
123290, Москва, Шелепихинская наб., д. 32
тел.: (495) 256-6691, тел./факс: (495) 256-7145
e-mail: info@rusedit.com, http://www.rusedit.com

РУССКАЯ РЕДАКЦИЯ

Оглавление

Введение

Что узнает читатель этой книги?

В данной книге представлен перевод описания методологии Microsoft Solutions Framework (MSF) for Agile Software Development, которая входит в поставку Microsoft Team Foundation Server.

Эта методология описывает подход и организацию работы при создании программных продуктов и, в отличие, например, от более масштабной и абстрактной системы Microsoft Solutions Framework, готова к применению.

Для кого предназначена эта книга?

Книга предназначена в первую очередь для менеджеров проектов, а также для разработчиков, аналитиков, тестеров, архитекторов и, возможно, других участников команды, которые соотносятся с перечнем ролей описанного в книге процесса.

Как организован материал в этой книге?

В главах ***Общие определения***, ***Аспекты*** и ***Принципы*** представлено основное описание процесса и подход к его реализации. Мы рекомендуем внимательно изучить материал, представленный в этих главах.

В главе ***Начало работы*** перечислены первые шаги, которые следует предпринять при начале проекта.

В главе ***Роли*** дано описание ролей и работ, в которые они вовлечены. В зависимости от планируемой роли в проекте изучение материала в книге можно сузить до действий и операций, специфичных только для этой роли.

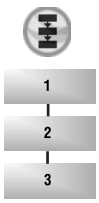
Обзор методологии

Общие определения

Роли

В гибкой методологии MSF разработки ПО все лица, участвующие в производстве, использовании и сопровождении продукта, обладают равными полномочиями. Участники команды имеют разные роли, связанные с их функциями, при этом ни одна из ролей не считается важнее другой: такое деление гарантирует реализацию качественного решения. Члены команды могут выступать в одной или нескольких ролях.

Действия и операции



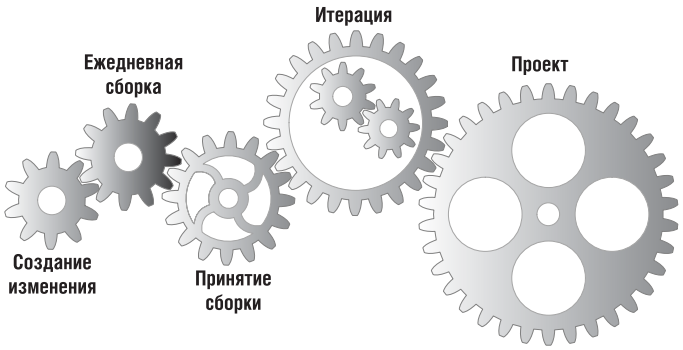
Роли выполняют *операции* (activity), которые могут быть сгруппированы в *действия* (workstream). Другими словами, действие — это набор операций.

Операции приводят к возникновению конечных продуктов и могут требовать для своего выполнения конечные продукты как результаты предыдущих операций.

Конечные продукты

Конечные продукты (deliverables) — это документы, электронные таблицы, проектные планы, исходные тексты программ и другие результаты операций.

Циклы и итерации



С помощью *циклов* описывается периодичность выполнения различных действий, а также частота выпуска и обновления конечных продуктов. Выполнение проектов и входящих в них задач производится циклично.

Тесная интеграция гибкой методологии MSF разработки ПО с Visual Studio Team System обеспечивает ускоренную *итеративную разработку* с постоянным уточнением деталей и совершенствованием конечного продукта. Определение требований к продукту, разработка и тестирование — это перекрывающиеся между собой повторяющиеся действия, ведущие к постепенному завершению проекта.

Вклад разных итераций в завершение проекта различен. Например, короткие итерации позволяют снизить погрешность предварительных оценок и предоставляют оперативные сводки о точности проектных планов. В результате каждой итерации должна появляться стабильно работающая часть системы.

Качество

Качество работ является одним из основополагающих принципов в гибкой методологии MSF разработки ПО. Все участники проекта должны осознавать важность качественной работы и руководствоваться этим принципом при проектировании, реализации, тестировании и внедрении системы. Особое внимание следует уделять качеству в таких областях, как безопасность, производительность и интерфейс пользователя.

Треки

Треки служат для группировки операций, завершающихся контрольными точками, то есть имеющих определенный бюджет и график. Треки не всегда связаны с задачами или работами, выполняемыми в циклах. Треки могут накладываться друг на друга, а между операциями, выполняемыми в разных треках, происходит постоянный обмен информацией.

Дисциплины

Дисциплины — это относящиеся к проекту в целом треки, в которых задействованы все участники команды.

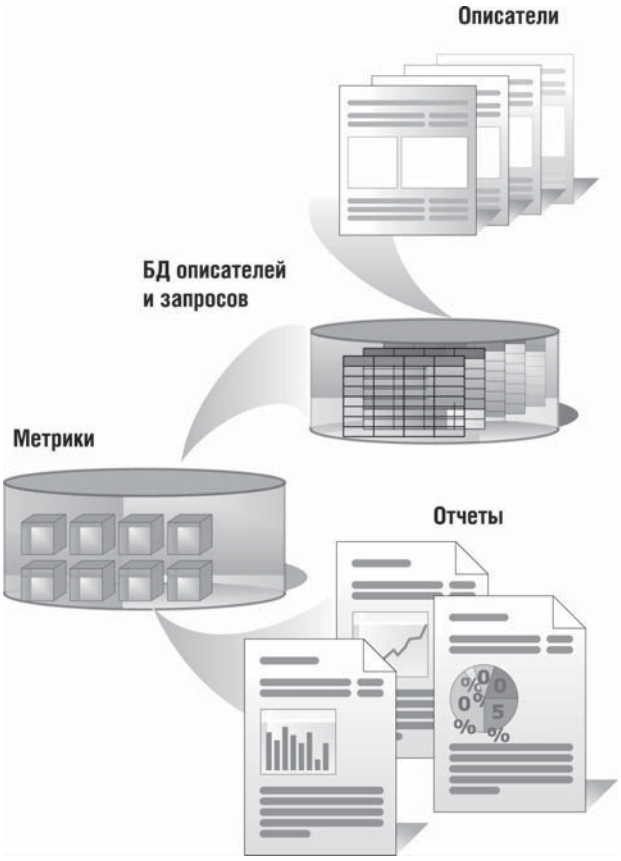
Руководство

Руководство (governance) — это контроль бюджета и графика выполнения проекта в привязке к текущим результатам. В гибкой методологии MSF разработки ПО определены пять контрольных точек руководства, в каждой из которых дается ответ на определенный вопрос. Не следует путать треки ру-

ководства с упорядочиванием работ посредством их группирования в итерации с циклами внутри.

Описатели

Описатель (work item) — это запись в базе данных, которая используется в Visual Studio Team Foundation для отслеживания назначения определенной операции или действия исполнителям, а также для мониторинга состояния этого задания. В гибкой методологии MSF разработки ПО определены пять типов описателей для назначения и отслеживания работ. Данные, хранящиеся в общей базе данных описателей и в хранилище показателей, позволяют в реальном времени отслеживать состояние проекта.



Описатели характеризуются *состоянием* (state), например состояния «новый дефект» и «закрытый дефект». При *переходе* (transition) из одного состояния в другое требуется *основание* (reason), например от состояния «новый дефект» сразу к состоянию «закрытый дефект» на основании того, что это дубликат уже существующего дефекта.

Аспекты

Методология MSF — больше чем набор правил, которым вы должны слепо следовать. Она призвана привить культуру, способствующую реализации успешных проектов. *Аспект* — это совокупность показателей, определяющая, как отдельный участник проекта оценивает ту или иную ситуацию и реагирует на нее. Участники проекта от его начала до завершения должны принимать во внимание аспекты при принятии решений, расстановке приоритетов, представлении своих интересов и при взаимодействии с другими участниками команды и прочими заинтересованными лицами. В MSF представлено девять аспектов.

Понимание практической ценности

Первый аспект акцентирует внимание участников проекта на его практических результатах. Каждая программная система создается с конкретной целью, например для решения некоторой проблемы в бизнесе. При пос-троении программных систем, особенно крупных, они испытывают на себе множество воздействий, часть из которых требует в ответ приложения значительных усилий. В конечном же счете программный проект оценивается по его полезности для бизнеса. Каждый участник должен всегда помнить о практических результатах и должен быть нацелен на их достижение.

Представление интересов

С программными проектами обычно связаны многие заинтересованные лица со своими интересами, например, заказчикам нужна хорошо работающая система. Можно также представить неодушевленные «заинтересованные лица», например саму разрабатываемую систему. Система может быть рассчитана на работу в течение тридцати лет, а может использоваться только тридцать минут. Системы, рассчитанные на длительное использование, разрабатываются дольше. Каждая из заинтересованных сторон (будь то люди или нет) в проекте должна быть представлена защитником своих интересов. Каждому участнику проекта важно понимать, чьи интересы он представляет. Следует помнить, что модель представления интересов — это совокупность ограничений, поэтому для получения оптимальных результатов зачастую неизбежны компромиссы.

Гордость за участие в проекте

Чувство гордости за участие в проекте — важный фактор для получения качественного продукта. Из чувства гордости вырастает мотивация и ответственность за выполнение проекта. Разработка ПО сродни искусству, и качественные системы — любимые детища их создателей. Комфортные условия труда, доверие, возможности роста, мотивированные, профессиональные люди — все это составляющие, необходимые для создания качественного

ПО. Поддержание чувства гордости за участие в проекте — задача и участников проекта, и организации. Применяйте определение общих затрат в проекте по принципу «от частного к общему», дайте проекту кодовое имя и четко идентифицируйте команду проекта — это поможет поддерживать чувство гордости. В MSF представители всех ролей ответственны за создание продукта самого высокого качества.

Своевременное выполнение своих обязанностей

Многие задачи при разработке ПО связаны между собой. Бывают моменты, когда мы не можем выполнить свою работу самостоятельно, без участия коллег. Для эффективной работы нужно своевременно решать задачи, от которых зависит деятельность других участников проекта, и держать их в курсе текущего состояния дел. Вы должны быть надежным и заслуживающим доверия партнером для своих коллег.

Видение системы в целом

При работе над конкретной частью системы есть риск «не увидеть за деревьями леса». Важно не только представлять свой отрезок работы, но и понимать, как ваша деятельность отражается на конечном продукте. Надо уделять больше внимания итогу, а не процессу. Это не значит, что процесс плох или не важен: процесс не самоцель, а средство достижения конечного результата. Нужно четко понимать, зачем вы выполняете то или иное действие и как результаты вашей работы интегрируются в общее решение. Думайте о том, что нужно для реализации системы, не вдаваясь в мелкие детали. Регулярно — не только во время плановых встреч — общайтесь с другими участниками команды.

Равнозначность участников

Аспект равнозначности участников команды означает их равноправие и равноценность, независимо от их ролей и представляемых ими интересов. Он может быть обеспечен за счет отсутствия ограничений в коммуникациях и прозрачности проекта. Этот аспект также предполагает взаимоуважение и заботу о каждом члене коллектива. Атмосфера взаимоуважения — необходимое условие максимальной эффективности в работе любой команды: она обеспечивает высокую коллективную ответственность, эффективные коммуникации и командный дух. Для успешной работы в команде равнозначных участников представители всех ролей должны заботиться о качестве создаваемого продукта, действовать как представители интересов заказчика и понимать решаемую бизнес-проблему.

Хозяйский подход

К ресурсам проекта, корпоративным и вычислительным ресурсам нужно относиться по-хозяйски. Такой подход нужно применять во всем — от ис-

пользования эффективных методов управления проектом до оптимизации системных ресурсов. Дальновидным шагом будет снабжение описателей ошибок подробнейшими данными, которые помогут кому-то в дальнейшем. Также полезно давать точную оценку трудозатрат на выполнение задач разработки и тестирования. Существующие ресурсы должны использоваться везде, где это возможно.

Непрерывное обучение

Готовность к обучению включает в себя постоянное самосовершенствование участника команды путем накопления знаний и обмена ими с другими. Надо учиться на чужом опыте: не повторять ошибок и применять успешные подходы. В графике проекта нужно предусмотреть время для обучения членов команды, анализа текущего состояния дел и проделанной работы. Кроме того, важной индивидуальной чертой каждого участника команды должно быть стремление к обмену знаниями с другими.

Приверженность качеству

Аспект приверженности качеству предполагает такой подход к планированию и реализации решения, который учитывает все потребности конечного пользователя. При этом качество в таких областях, как производительность или безопасность, должно учитываться на всех этапах разработки. Без такого подхода к обеспечению качества систему, как правило, создают, делая неявные допущения о ее поведении. В результате функционирование системы не удовлетворяет заказчика. Для обеспечения качества нужно видеть систему в целом и не полагаться на неявные допущения, а руководствоваться явно сформулированными требованиями к качеству.

Принципы

Microsoft Solutions Framework (MSF) for Agile Software Development — это методология построения .NET-приложений и другого объектно-ориентированного ПО. В ее основе лежит гибкий, управляемый, адаптируемый к контексту проекта процесс разработки. Непосредственно в гибкую методологию MSF разработки ПО включены нормы работы с требованиями к качеству в таких областях, как производительность и безопасность. В данной методологии также учитываются конкретные условия для реализации каждого проекта. При таком подходе создается адаптивный процесс, обеспечивающий преодоление ограничений большинства гибких процессов разработки ПО и достижение целей, установленных концепцией проекта.

Партнерство с заказчиками

В модели команды MSF, основанной на представлении интересов, ключевое внимание уделяется пониманию потребностей заказчика и участию представителей заказчика в реализации проекта. Главный приоритет для любой профессиональной команды — действовать так, чтобы клиенты были довольны результатами. Ориентироваться на клиента — значит понимать его проблемы. Разобравшись в решаемой проблеме клиента, надо вовлечь его в работу в том объеме, который соответствует его ожиданиям. На всех фазах проекта нужно поддерживать открытое, активное, регулярное общение с заказчиком. Это важно потому, что зачастую только заказчик видит разницу между действительными и мнимыми проблемами бизнеса.

Единая точка зрения

В MSF настойчиво рекомендуется выработать единую точку зрения на подходы к реализации решения. Общий взгляд всех участников команды гарантирует, что они одинаково понимают, каков будет результат их работы; они сплачиваются вокруг единой цели и одинаково трактуют потребности заказчика. Совместная работа единомышленников всегда эффективнее, поскольку решения принимаются не произвольно, а основываясь на общем видении проблемы. Без единой точки зрения участники команды могут иметь противоречивые представления о целях работы, а достигнуть нужных результатов в этом случае сложнее. Даже после того как результат получен, не все участники могут согласиться с тем, что он оказался успешен. Понимание достоинств выработанного решения и умение их сформулировать зачастую является ключевым фактором успеха.

Инкрементная выдача результатов

Ничто так не завоевывает доверие заказчика, как частая выдача результатов. Очень выгодно постоянно иметь «практически готовый» продукт. Реагирование на потребности заказчика регулярной выдачей небольших работоспособных дополнений наглядно демонстрирует прогресс разработки. При частой выдаче результатов для заказчика существует гарантия работоспособности команды и развития процесса и инфраструктуры. При этом риски, ошибки и упущенные требования выявляются на ранних стадиях. Инкрементный подход подтверждает правильность проектных решений и обеспечивает их корректировку благодаря эффективной обратной связи.

Для частой выдачи результатов работа должна быть разбита на небольшие фрагменты, результаты должны выдаваться точно по графику, а в случае нескольких вариантов решения должны предоставляться они все, а не один успешно выбранный.

Планируйте, выполняйте планы, оценивайте прогресс и темп работы команды на основе инкрементной выдачи результатов — и вы увеличите

рентабельность. Минимизируйте деятельность, не приносящую понятных заказчику конкретных результатов. Применяйте итерации для поддержания ритма выдачи тех результатов, которые ваш клиент способен оценить. Внимательно оценивайте эффективность передачи работ от одного участника команды другому. Разработчики должны постоянно проверять создаваемый продукт, а ваша компания — испытывать новые версии.

Инвестиции в качество

В успешной команде каждый участник должен чувствовать свою ответственность за разрабатываемый продукт и быть представителем интересов заказчика, заботясь о качестве на протяжении всего жизненного цикла разработки. Качество должно учитываться в планах и графиках. Используйте ассигнования на исправление дефектов (запланированные итерации для устранения неисправностей) для снижения общих затрат на ошибки. Таким образом вы несколько снизите темпы разработки, обеспечив резерв времени в последующих итерациях для уменьшения числа дефектов.

Широкие полномочия участников проекта

Команда работает эффективно, когда каждому участнику предоставлены все необходимые полномочия для выполнения его обязанностей и он уверен, что если его работа зависит от коллег, она будет выполнена. В свою очередь, заказчик вправе считать, что команда выполнит свои обязательства, и может строить свои планы, исходя из этого предположения. В случае возможной задержки или изменения функций необходимо своевременно уведомить об этом клиента.

Модель команды

Модель команды MSF описывает подход компании Microsoft к структуризации участников команды и их действий, приводящих к успеху проекта. Фундаментальными принципами модели команды MSF являются:

- команда — это группа равнозначных сотрудников с четкой подотчетностью, разделяемой ответственностью и свободным общением;
- защита интересов каждого ключевого представителя, вовлеченного в проект, голос которого может повлиять на успех;
- необходимая гибкость в масштабировании команды.

Четкая подотчетность

Модель команды MSF основывается на предпосылке, что цели всех участников равноценны, сегменты деятельности в рамках проекта уникальны и при этом нет единого представителя всех разнообразных целей. При таком подходе в команде равнозначных участников целесообразно совмещать четкую подотчетность перед заинтересованными сторонами с коллективной ответственностью за конечный результат. Каждый участник подотчетен коллективу (и организации, которая за ним стоит) за достижение целей, установленных для его роли. Другими словами, представитель каждой роли обязан отчитываться за свой вклад в конечный результат. Ответственность же в команде равнозначных участников распределяется равномерно. На то есть причины: во-первых, невозможно выделить результат работы отдельного участника из общего решения, и, во-вторых, команда работает эффективнее, когда каждый ее участник, исполняющий любую роль, видит картину в целом. Такая взаимозависимость членов коллектива должна стимулировать их интерес к областям, за которые они не подотчетны, обеспечивая полноценное использование всего спектра знаний, компетенции и опыта команды.

Достижения проекта относятся ко всем участникам: они разделяют всеобщее уважение и заслуженное вознаграждение в случае удачных решений. Вместе с этим, каждый участник проекта должен задуматься над повышением своего профессионального уровня, извлекая уроки из менее удачных проектов.

Учет любого опыта

Каждый проект, среда и команда уникальны, следовательно любой проект и всякая его итерация — это потенциальный источник дополнительного опыта. Однако нельзя узнать ничего нового без обратной связи и открытости в коллективе. Если не заботиться об участниках команды, не давать им почувствовать себя раскрепощенными, обратная связь будет ограниченной и неэффективной. Если же обеспечить людям психологический комфорт, каждый участник и команда в целом будут нацелены на самосовершенствование, углубление своих знаний и обмен ими с коллегами. Как уже отмечалось, в графике проекта нужно учесть время на обучение, в том числе на основе преды-

дущего и чужого опыта. Подробный анализ сделанного в доброжелательной атмосфере — ключевой принцип MSF, обеспечивающий эффективные коммуникации между членами команды.

Свободное общение

Исторически во многих организациях и проектах применялся принцип необходимого знания, в соответствии с которым сотрудники получали доступ только к тем сведениям, которые были нужны для выполнения конкретных функций. Зачастую такой подход приводит к недоразумениям и снижает шансы команды на достижение успеха.

В MSF предлагается открытый и честный обмен информацией как внутри команды, так и с ключевыми заинтересованными сторонами вне ее. Свободный обмен информацией не только сокращает риск возникновения недоразумений и неоправданных затрат, но и обеспечивает максимальный вклад всех участников команды в снижение существующей в проекте неопределенности.

Гибкость и готовность к переменам

Чем большую отдачу для бизнеса от внедрения новых технологий хотят получить организации, тем больше они должны быть готовы вкладывать в новые области. Нельзя рассчитывать на успех, не осваивая новые территории. В MSF предполагается, что все вокруг непрерывно меняется, и оградить проект от этих перемен невозможно. Применение модели проектной группы MSF гарантирует участие всех ролей и их вовлеченность в процесс принятия решений, обусловленных происходящими переменами. Эта модель поощряет *гибкость* (agility) при работе в меняющейся среде. Участие всех ролей в процессе принятия решений обеспечивает рассмотрение вопросов с учетом полного спектра точек зрения.

Начало работы

После создания проекта на базе гибкой методологии MSF разработки ПО в Team Foundation Server следует изучить последовательность дальнейших действий для организации работы над проектом и найти необходимые дополнительные сведения, которые позволят оптимально использовать процесс MSF.

Первые задачи

При создании проекта формируются описатели задач, которые позволяют распределить задания и сразу приступить к работе. Эти задачи вы можете просмотреть в электронной таблице Project Checklist.xls на портале проекта или с помощью Team Explorer (Проводника команды). Project Checklist.xls

находится в папке Project Management (Управление проектом) портала проекта или в папке Documents (Документы) при просмотре в Team Explorer. Эти задачи вы также можете увидеть, выполнив запрос Project Checklist (Контрольный список проекта) в Team Explorer.

Условия завершения

В контрольном списке проекта содержатся все описатели, помеченные как содержащие условия завершения для данной итерации. Например, в описателе задачи есть поле Exit Criteria (Условия завершения). Задачи, у которых в данном поле содержится значение Yes (Да), представлены в контрольном списке. Контрольный список проекта используется для отслеживания всех критически важных работ, которые должны быть выполнены для успешного завершения итерации.

Начальные задачи

Начальные задачи в контрольном списке проекта — это задания, которые должны быть завершены до начала первой итерации. В первую очередь идут подготовительные действия по информированию участников проекта о его начале, а заканчивается список задачами планирования первой итерации.

Актуализация данных проекта

Обычно проектные работы начинаются до создания проекта команды на Team Foundation Server. Если у вас уже есть концепция проекта, какие-либо требования, а также назначенные или требующие отслеживания задачи, вам нужно сразу обновить проектные данные.

Конечные продукты

Просмотрите все шаблоны конечных продуктов в папке Documents в Team Explorer или на портале проекта. Если у вас уже есть конечные продукты или сопутствующая информация, вы можете актуализировать проект команды, загрузив на его портал внешние документы или обновив существующие документы MSF на портале проекта.

Описатели

В MSF используются описатели дефектов, рисков, задач, сценариев и требований к качеству. Если у вас уже есть описатели, применимые в данном проекте, импортируйте их. Например, если вы уже определили какие-либо риски, задокументируйте их в виде описателей рисков с помощью Team Explorer.

Начало действий

Работа над проектом начинается с того, что бизнес-аналитик формулирует концепцию проекта. Этот этап является частью выполнения действия «Формулирование концепции проекта» (Capture Project Vision). Кроме того, менеджер проекта начинает планировать первую итерацию в рамках выполняемого им действия «Планирование итерации» (Plan an Iteration). После запуска этих двух действий, естественным образом начинаются и другие действия, в которые вовлечены участники проекта с другими ролями.

Типы описателей

В MSF используется пять типов описателей. В описателях каждого типа содержатся данные для формирования различных отчетов. Чтобы отчеты отвечали своему назначению и выполнение работ отслеживалось адекватно, участники проекта должны корректно использовать типы описателей.

Запросы

Состояние работ над проектом можно узнать с помощью запросов, выдаваемых из папки Work Items (Описатели) в Team Explorer. Для получения дополнительных сведений о запросах щелкните вкладку Index (Указатель), а затем — Queries (Запросы).

Отчеты

В начале работы над проектом нет значимых показателей для создания полезных отчетов, но с каждым днем появляется все больше данных, по которым можно судить о прогрессе проекта. Прежде всего, следует разобраться с отчетом Remaining Work (Оставшаяся работа). В первой и последующих итерациях этот отчет позволит вам следить за ходом проекта. Для получения дополнительных сведений об отчетах щелкните вкладку Index (Указатель), а затем — Reports (Отчеты).

Методология

Роли

Бизнес-аналитик

В рамках командной модели MSF (MSF Team Model) *бизнес-аналитик* участвует в управлении продуктом. Основная задача бизнес-аналитика — разобраться в возможностях системы, относящихся к бизнесу и раскрыть их команде. Он работает с пользователями и другими заинтересованными лицами, чтобы понимать их потребности и задачи, трансформировать их в кон-

кретные определения, сценарии и требования к качеству, которые команда разработчиков будет использовать для построения приложения. Кроме того, бизнес-аналитик определяет ожидания от функциональных возможностей системы и управляет ими. В проекте он представляет пользователей и участвует в управлении продуктом в том смысле, что постоянно отслеживает интересы пользователей и заказчиков проекта. И, наконец, бизнес-аналитики отвечают за обеспечение взаимодействия между разработчиками и пользователями. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Сотрудник» (Contributor). Это позволит ему выполнять все функции, необходимые в рамках его деятельности, например, такие как создание и изменение документов, описателей и конечных продуктов.

Действия и операции роли

- Действия:
  - формулировка концепции проекта;
  - разработка требований к качеству;
  - создание сценария;
  - планирование итерации.
- Операции:
  - написание концепции;
  - определение собирательных образов;
  - уточнение собирательных образов;
  - определение требований к качеству путем «мозгового штурма»;
  - создание моментальных снимков деятельности;
  - определение приоритетов в списке требований к качеству;
  - написание требований к качеству;
  - определение требований безопасности;
  - работа над сценариями методом «мозгового штурма»;
  - определение приоритетов в списке сценариев;
  - оценка задачи по разработке базы данных;
  - выполнение ретроспективного анализа;
  - оценка сценария;
  - оценка требования к качеству;
  - составление графика сценария;
  - составление графика реализации требований к качеству;
  - обзор целей.

Менеджер проекта

В рамках командной модели MSF *менеджер проекта* участвует в управлении продуктом. Основная задача менеджера проекта — добиваться выполнения поставленных перед командой задач в соответствии с графиком и в рамках бюджета. На менеджере проекта лежат обязательства по планированию и составлению графика работ, включающие разработку проекта и планов итераций, отслеживание состояния дел и составление отчетов, а также определению рисков и выработке мер по их уменьшению. Менеджер проекта также проводит консультации с бизнес-аналитиками по планированию сценариев и выработке требований к качеству для каждой итерации, консультируется с архитекторами и разработчиками для оценки объемов работ, советуется с тестировщиками, чтобы спланировать тестирование, и, кроме того, содействует взаимодействию участников команды. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Администратор проекта» (Project Administrator). Это позволит ему выполнять такие функции, как создание нового проекта, формирование новой команды и добавление в нее участников.

Действия и операции роли

- Действия:
  - контроль итерации;
  - планирование итерации;
  - ведение проекта.
- Операции:
  - оценка задачи по разработке базы данных;
  - мониторинг итерации;
  - снижение риска;
  - выполнение ретроспективного анализа;
  - определение продолжительности итерации;
  - оценка сценария;
  - оценка требования к качеству;
  - разбиение сценариев на задачи;
  - разбиение требования к качеству на задачи;
  - составление графика сценария;
  - составление графика реализации требований к качеству;
  - оценка хода выполнения;
  - оценка пороговых значений показателей тестов;
  - определение риска;

- обзор целей;
- классификация дефектов.

Архитектор

В рамках командной модели MSF *архитектор* отвечает за архитектуру проекта. Его основная задача — обеспечить успех проекта путем разработки основных принципов приложения, которые включают в себя как организационную конфигурацию системы, так и физическую структуру ее развертывания. При этом архитектор должен стремиться к снижению сложности путем разделения системы на понятные и простые части. Архитектура приложения чрезвычайно важна, поскольку она не просто устанавливает этапы построения системы, а определяет, будет ли приложение обладать свойствами, присущими успешным проектам. К ним относятся: удобство использования, надежность, практичность сопровождения, производительность и безопасность, а также возможности модификации в случае изменения требований. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Сотрудник» (Contributor). Это позволит ему выполнять все функции, необходимые в рамках его деятельности, такие как создание и изменение документов, описателей и конечных продуктов.

Действия и операции роли

- Действия:
  - разработка архитектуры решения;
  - планирование итерации.
- Операции:
  - разделение системы на подсистемы;
  - определение интерфейсов;
  - разработка модели угроз;
  - разработка модели производительности;
  - создание архитектурной модели;
  - создание архитектуры инфраструктуры;
  - определение требований безопасности;
  - разбиение сценариев на задачи;
  - разбиение требования к качеству на задачи.

Разработчик

В рамках командной модели MSF *разработчик* выполняет разработку приложения. Его основная задача — реализовать приложение согласно спецификациям и в установленные сроки. Разработчик также помогает уточнять физический дизайн, оценивать время и усилия для выполнения конкретных

элементов, выполняет реализацию функций или руководит ею, подготавливает продукт к внедрению и является экспертом команды в технологических областях. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Сотрудник» (Contributor). Это позволит ему выполнять все функции, необходимые в рамках его деятельности, такие как создание и изменение документов, описателей и конечных продуктов.

Действия и операции роли

- Действия:
  - сборка продукта;
  - устранение дефекта;
  - реализация задачи по разработке;
  - планирование итерации.
- Операции:
  - разделение системы на подсистемы;
  - определение интерфейсов;
  - разработка модели производительности;
  - запуск сборки;
  - проверка выпуска;
  - исправление сборки;
  - приемка сборки;
  - воспроизведение дефекта;
  - определение причины возникновения дефекта;
  - переназначение дефекта;
  - выбор стратегии устранения дефекта;
  - создание или изменение теста модуля;
  - выполнение теста модуля;
  - рефакторинг кода;
  - обзор кода;
  - интеграция изменений;
  - оценка задачи по разработке;
  - кодирование;
  - анализ кода;
  - оценка задачи по разработке базы данных;
  - выполнение ретроспективного анализа;
  - определение продолжительности итерации;
  - оценка сценария;

- оценка требования к качеству;
- разбиение сценариев на задачи;
- разбиение требования к качеству на задачи;
- создание заметок о выпуске;
- развертывание продукта.

Тестировщик

В рамках командной модели MSF *тестировщик* выполняет задачи тестирования продукта. Основной задачей тестировщика является выявление проблем в продукте, которые могут неблагоприятно повлиять на его качество. Тестировщик обязан понимать контекст проекта и помогать остальным членам команды понимать решения, основанные на этом контексте. Ключевая цель тестировщика — поиск серьезных дефектов в продукте путем его тестирования и последующее их описание. Каждый найденный дефект тестировщик должен сопроводить точным описанием вредного воздействия и предложить способ обойти дефект, чтобы уменьшить это воздействие. Он должен как можно проще описать дефект и последовательность, в которой его можно воспроизвести.

Тестировщик участвует в деятельности команды по определению стандартов качества продукта. Цель тестирования — убедиться, что известные функции работают правильно, и выявить возможные проблемы продукта. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Сотрудник» (Contributor). Это позволит ему выполнять все функ-ции, необходимые в рамках его деятельности, такие как создание и изменение документов, описателей и конечных продуктов.

Действия и операции роли

- Действия:
  - планирование итерации;
  - закрытие дефекта;
  - тестирование требования к качеству;
  - проверка сценария.
- Операции:
  - выполнение ретроспективного анализа;
  - разбиение сценариев на задачи;
  - разбиение требования к качеству на задачи;
  - проверка исправления;
  - закрытие дефекта;
  - определение подхода к тестированию;

- создание теста производительности;
- создание теста безопасности;
- создание стрессового теста;
- создание нагрузочного теста;
- выбор и запуск тестового задания;
- документирование дефекта;
- оценка пороговых значений показателей тестов.

Релиз-менеджер

В рамках командной модели MSF *релиз-менеджер* отвечает за операции по выпуску продукта. Основная цель релиз-менеджера — обеспечение выпуска готового продукта. Он координирует выпуск продукта со специалистами по эксплуатации, создает план выпуска продукта и сертифицирует подготовленные к выпуску версии для поставки или развертывания. Человек, назначенный на эту роль, должен быть добавлен в группу доступа «Администратор проекта» (Project Administrator). Это позволит ему выполнять такие свои функции, как создание нового проекта, формирование новой команды и добавление в нее участников.

Действия и операции роли

- Действия:
  - выпуск продукта.
- Операции:
  - исполнение плана выпуска;
  - создание заметок о выпуске;
  - развертывание продукта.

Администратор баз данных

Основная задача *администратора баз данных* в контексте разработки базы данных — поддержка создания проектов баз данных, а также внесение изменений проекта в рабочую базу данных. В дополнение к этому он должен выполнять традиционные задачи, такие как ежедневное администрирование и поддержка серверов баз данных.

Действия и операции роли

- Действия:
  - создание проекта базы данных;
  - развертывание проекта базы данных.

- Операции:
  - создание проекта базы данных;
  - импорт существующей базы данных;
  - установка параметров сборки и развертывания;
  - изменение сгенерированных сценариев;
  - проверка проекта базы данных;
  - размещение проекта базы данных в службе управления исходным кодом;
  - синхронизация проекта базы данных;
  - проверка сборки;
  - выполнение тестирования модулей базы данных;
  - анализ изменений;
  - создание резервной копии рабочей базы данных;
  - установка базы данных на тестовом сервере;
  - установка базы данных.

Разработчик баз данных

Основная задача *разработчика баз данных* — выполнение комплекса задач по разработке базы данных в установленные сроки. Кроме того, он отвечает за оценку стоимости, контроль над реализацией функций и помощь остальным участникам команды по вопросам, связанным с базами данных. Разработчик баз данных совместно с администратором баз данных и прикладными разработчиками участвует в итеративном жизненном цикле разработки базы данных.

Действия и операции роли

- Действия:
  - реализация задачи по разработке базы данных;
  - планирование итерации.
- Операции:
  - оценка задачи по разработке базы данных;
  - обновление локальной среды проекта;
  - выполнение теста модуля базы данных;
  - рефакторинг базы данных;
  - определение продолжительности итерации;
  - оценка сценария;
  - оценка требования к качеству;

- разбиение сценариев на задачи;
- разбиение требования к качеству на задачи.

Действия

Контроль итерации

Все выполняющиеся итерации необходимо контролировать. Время, затрачиваемое на решение проблем, влияет на общий темп проекта. При выдаче запросов поиска проблем зарегистрировать заблокированные описатели позволяет флаг «Issue». Запросы поиска проблем должны выполняться хотя бы один раз в начале рабочего дня. По завершении итерации необходим анализ сделанного для определения путей совершенствования процесса и рабочей среды. Если итерация начинается с собрания, на котором она планируется, заканчиваться она должна собранием, закрывающим итерацию, на котором проводится ретроспективный анализ проделанной работы.

Операции:

Мониторинг итерации	Обзор приоритетов в описателе
Снижение риска	Уменьшение вероятности риска
Выполнение ретроспективного анализа	Организация ретроспективного анализа

Операция: Мониторинг итерации

Понимание текущего состояния итерации очень важно для ее успешного завершения. Один из важнейших моментов в оценке хода итерации — анализ описателей высокоприоритетных сценариев, требований к качеству и ошибок в данной итерации. Актуализируйте план итерации, чтобы увидеть незакрытые задачи, связанные с описателями перечисленных типов. Определите завершившиеся задачи и распределите приоритеты так, чтобы сначала заканчивались наиболее важные части итерации. Задача менеджера проекта состоит в том, чтобы распознать потенциальные проблемы, приводящие к остановке работы, а также распознать узкие места, снижающие темп выполнения работы, и обеспечить непрерывное выполнение проекта.

Обзор приоритетов в описателе	Если в процессе итерации изменяются приоритеты описателей сценариев или требований к качеству, менеджер проекта отвечает за то, чтобы организовать короткое совещание с участием бизнес-аналитиков, архитекторов, разработчиков и тестировщиков для обсуждения изменений. Приоритеты могут меняться по многим причинам: из-за возникновения технических вопросов, изменения приоритетов заказчика или зависимостей
Решение проблем	Ежедневно выполняйте запросы поиска проблем и проверяйте, не заблокированы ли те или иные задачи разработки или тестирования. При возникновении проблем привлекайте к их решению других участников команды

Операция: Снижение риска

Для снижения риска необходимо предпринять шаги с целью уменьшения его вероятности или смягчения его воздействия. Проактивное управление рисками эффективней реактивного. Чтобы проактивное управление рисками было рентабельным, в первую очередь нужно уделять внимание наиболее рискованным моментам, но тратить на их устранение не больше, чем они могут принести убытков.

Уменьшение вероятности	Предложите варианты снижения вероятности возникновения неблагоприятных событий
Смягчение воздействия	Если вероятность возникновения неблагоприятной ситуации остается высокой, предложите способы смягчения вредного воздействия
Выработка плана на случай чрезвычайных обстоятельств	Если риск все же остается неприемлемо высоким, пересмотрите все сделанные допущения и требования к качеству, особенно относящиеся к эксплуатационной среде, доступности и специальным возможностям
Применение решений	Реализуйте выбранное решение, обеспечивающее уменьшение вероятности и смягчение воздействия неблагоприятных обстоятельств

Операция: Выполнение ретроспективного анализа

Обратная связь является важной составляющей любого гибкого процесса разработки ПО. Поскольку всякая проектная группа — как и каждая проблема, с которой она сталкивается — уникальна, создание среды, отвечающей всем потребностям ее участников, является ключевым фактором в увеличении про-

изводительности их труда. Ретроспектива позволяет команде разработчиков проанализировать свои правильные и ошибочные действия после завершения очередной итерации. Главное здесь — увидеть возможные пути совершенствования процесса и соответствующим образом перестроиться. Чтобы извлечь максимальную пользу от ретроспективы, соответствующее совещание должно длиться около двух часов. Однако если члены команды видят, что все идет гладко, встречу можно сократить. Последняя итерация заканчивается ретроспективным анализом проекта.

Организация совещания для ретроспективы	Пригласите участников команды на совещание. Оно должно проводиться после завершения итерации с участием всех членов коллектива. Совещание должно быть относительно коротким, но достаточным для того, чтобы сделать необходимые выводы (обычно около двух часов). Окончательный ретроспективный анализ (в конце проекта) может занять больше времени, вплоть до нескольких дней для масштабных проектов. Это завершающее собрание целесообразно организовывать таким образом, чтобы оно не мешало повседневной деятельности
Проведение совещания	Установите четкие правила участия сотрудников в совещаниях. Используйте список из двух столбцов: «плюсов» и «минусов» завершенной итерации
Учитывайте результаты анализа в следующих итерациях	Там, где это возможно, используйте результаты проделанного ретроспективного анализа для корректировки планов следующих итераций

Планирование итерации

При планировании итерации важно заранее определить правильный баланс между сценариями, требованиями к качеству и ассигнованиями на исправление ошибок. За каждую итерацию можно выполнить ограниченный объем работы. В текущую итерацию попадают сценарии и требования к качеству, имеющие наибольшую практическую ценность. Первоначальный план итерации создается на основе *элементов сценариев* (scenario entries) и требований к качеству, имеющих пока приблизительные оценки. Затем элементы, попавшие в план итерации, передаются бизнес-аналитику для написания сценариев. После этого разработчики и тестировщики разбивают сценарии на задачи. Эти задачи распределяются между разработчиками, и делается более точная оценка затрат, которая используется для равномерного распределения нагрузки между разработчиками. Завершение выработки плана итерации происходит на собрании участвующих в итерации бизнес-аналитиков, менеджеров проекта и разработчиков.

Операции:

Определение продолжительности итерации	При планировании отталкивайтесь от даты окончания проекта
Оценка сценария	Определите подмножество из списка сценариев
Оценка требования к качеству	Определите подмножество из списка требований к качеству
Планирование сценария	Создайте набросок плана итерации
Планирование требования к качеству	Создайте набросок плана итерации
Планирование ресурсов на исправление дефектов	Зарезервируйте ресурсы на устранение дефектов
Разбиение сценария на задачи	Соберите команду
Разбиение требования к качеству на задачи	Соберите команду

Операция: Определение продолжительности итерации

Итерация — это набор задач, которые должны быть выполнены в течение фиксированного промежутка времени. Общее время, необходимое для завершения задач, называется продолжительностью итерации. График проекта разбит на последовательность итераций и задач, которые, в свою очередь, имеют свой график. Определение продолжительности итерации включает в себя рассмотрение всех ключевых факторов, включая дату поставки продукта (если она определена), размер сценариев и их общее время. Обычно продолжительность итерации определяется в неделях, однако возможны и меньшие единицы времени. Продолжительность итераций определяется в начале проекта, а затем на основе этих данных определяется общая продолжительность проекта.

Планирование от даты окончания	Если для проекта задана дата поставки конечного результата проекта, то она определяет продолжительность итераций. Создайте план проекта или с помощью календаря выясните, сколько времени может быть отведено на разработку
Проверка размера сценария	После окончания планирования первой итерации проверьте, не разбиваются ли сценарии на более мелкие из-за ее продолжительности. Увеличьте продолжительность итерации, если это необходимо, для того, чтобы она соответствовала размеру сценария

Проверка затрат на интеграцию	Интеграция выполняется на протяжении всей итерации по окончании каждой задачи разработки. Учитываются все особые требования или затраты на интеграцию
Анализ продолжительности итерации	Проводите анализ продолжительности итерации совместно с проектной группой. Оптимальная продолжительность позволяет выполнить небольшое количество сценариев или требований к качеству. Каждая итерация должна заканчиваться выпуском внутренней или внешней версии с новыми функциональными возможностями, исправленными ошибками, реализованными сценариями или требованиями к качеству

Операция: Оценка сценария

Оценка сценария выполняется для создания общей картины, которая помогает понять, сколько усилий потребуется для реализации сценария. Подобные оценки позволяют бизнес-аналитику расставлять приоритеты и регулировать внешние ожидания. Менеджер проекта совместно с разработчиками выполняет оценки для наиболее приоритетных сценариев из списка.

Определите подмножество из списка сценариев	Откройте список сценариев на портале проекта. Для каждого сценария с максимальным приоритетом назначьте разработчика или группу разработчиков, которые смогут выполнить оценку
Оценка реализации	Совместно с разработчиками оцените *приблизительный порядок величины* (rough order of magnitude, ROM) сложности каждого сценария. Для многих проектов может быть применено следующее правило. Если необходимо выполнить не более шести заданий, требующих 1–2 дня, в этом поле указывается значение 1. Если необходимо выполнить от 6 до 12 заданий, требующих 1–2 дня, в этом поле указывается значение 2. Если трудозатраты больше, в данном поле указывается значение 3 и рассматривается возможность разбиения сценария на части. Важно соблюдать примерные соотношения оценок, т. е. сценарий с порядком сложности 2 должен быть приблизительно в два раза больше сценария с оценкой сложности 1

Определение особых требований	Совместно с разработчиками определите риски, которые могут задержать или помешать успешному завершению работ по реализации сценария, такие как недостаток знаний или зависимость от приложений сторонних поставщиков. Создайте описатель для каждого выявленного риска

Операция: Оценка требований к качеству

Оценка требований к качеству служит для определения усилий, необходимых для их реализации. Подобные оценки позволяют бизнес-аналитику устанавливать приоритеты и регулировать внешние ожидания. Менеджер проекта совместно с разработчиками выбирает из списка наиболее приоритетные требования к качеству и выполняет для них оценки.

Выбор требований к качеству из списка	Откройте список требований к качеству на портале проекта. Для каждого сценария с максимальным приоритетом назначьте разработчика или группу разработчиков, которые смогут выполнить оценку
Выполнение оценки реализации	Совместно с разработчиками оцените *приблизительный порядок величины* сложности каждого требования к качеству. Для многих проектов может быть применено следующее правило. Если необходимо выполнить не более шести заданий, требующих 1–2 дня, в этом поле указывается значение 1. Если необходимо выполнить от 6 до 12 заданий, требующих 1–2 дня, в этом поле указывается значение 2. Если трудозатраты больше, в данном поле указывается значение 3 и рассматривается возможность разбиения на части задачи реализации требования к качеству. Важно соблюдать примерные соотношения оценок, т. е. сценарий с порядком сложности 2 должен быть приблизительно в два раза больше сценария с оценкой сложности 1
Определение ограничений	Совместно с разработчиками определите риски, которые могут помешать успешному и своевременному завершению работ по реализации требования, такие как недо статок знаний или зависимость от ПО сторонних поставщиков. Создайте описатель риска и прикрепите его к соответствующему требованию к качеству для отображения зависимости

Операция: Разбиение сценариев на задачи

Сценарий перед реализацией нужно разбить на задачи. Такое разбиение позволяет точнее выполнить оценку сложности и распределить задачи между разработчиками. Некоторые сценарии могут затрагивать несколько подсистем приложения. В этом случае для каждой затрагиваемой области создается отдельная задача. Ключевым моментом в успешном разбиении сценария является подключение к этому процессу разработчиков и тестировщиков, а также поощрение их выбирать себе задачи самостоятельно. Все задачи должны быть распределены.

Соберите команду	Соберите архитекторов, разработчиков и тестировщиков, которые будут работать над сценарием, либо как-то влиять на него. Менеджер проекта организует короткое собрание или дискуссию по электронной почте и фиксирует задачи
Определение архитектуры и задач разработки	Начиная с того места, где новый сценарий отличается от уже реализованных сценариев, проследите его до конца. Обращайте внимание на видимые пользователю элементы, изменения в инфраструктуре или структуре данных, а также на моменты, которые влияют на всю архитектуру. Определите изменения в компонентах системы, которые произойдут в результате реализации планируемого сценария
Выбор архитектуры и задач разработки	Пусть архитекторы и разработчики сами выберут себе задачи
Назначение ответственного разработчика сценария	Передайте сценарий одному из разработчиков, работающих над задачами сценария. Он будет отвечать за сквозное тестирование интеграции этих задач
Создание тестовых задач	Отредактируйте документ, описывающий тестирование для данной итерации, указав в нем тестовые данные и заполнив другие разделы. Создайте задачи для разработки необходимых тестов проверки сценария
Выбор тестовых задач	Пусть тестировщики сами выберут себе задачи

Операция: Разбиение требования к качеству на задачи

Перед реализацией каждое требование к качеству должно быть разбито на задачи. Такое разбиение позволяет точнее выполнить оценку сложности и способствует лучшему пониманию разработчиками предстоящей работы. В рамках реализации требований к качеству обычно нужно выполнить не-

сколько задач. Ключевым моментом в успешном разбиении требования к качеству является подключение к этому процессу разработчиков и тестировщиков, а также поощрение их выбирать себе задачи для себя самостоятельно. Все задачи должны быть распределены.

Соберите команду	Соберите архитекторов, разработчиков и специалистов по тестированию, которые будут работать над требованием к качеству, либо как-то влиять на него. Менеджер проекта организует короткое собрание или дискуссию по электронной почте и фиксирует задачи
Определение архитектуры и задач разработки	По ссылкам найдите сценарии, имеющие отношение к требованию к качеству, и проведите их обзор. Требования к качеству различаются по области своего влияния. Часть из них влияет только на один сценарий, в то время как другие могут оказывать воздействие сразу на несколько сценариев. Если будут обнаружены сценарии, пропущенные бизнес-аналитиком, добавьте их к соответствующему требованию к качеству
Выбор архитектуры и задач разработки	Доверьте архитекторам и разработчикам самим выбрать себе задачи
Назначение ответственного разработчика требования к качеству	Передайте требование к качеству разработчику, которому принадлежат задачи по соответствующему сценарию. Он будет отвечать за сквозное тестирование интеграции этих задач
Создание тестовых задач	Отредактируйте документ, описывающий тестирование для данной итерации, указав в нем тестовые данные и заполнив другие разделы. Создайте набор задач для разработки необходимых тестов
Выбор тестовых задач	Пусть тестировщики сами выберут себе задачи

Операция: Планирование ресурсов на исправление дефектов

Составление графика выделения ресурсов на исправление дефектов предполагает исправление дефектов в порядке, определяемом их важностью. Обычно ресурсы на исправление дефектов выделяются всей группе разработчиков, но могут ограничиваться одним или несколькими разработчиками из команды. Ассигнования на исправление дефектов берутся из общего бюджета разработки.

Планирование ассигнований на исправление дефектов	Из общего бюджета разработки выделите часть, необходимую на исправление дефектов (единица измерения — абстрактный человеко-день). Если план итерации выполняется в Microsoft Excel, обозначьте эти ассигнования как резерв бюджета. Если план итерации выполняется в Microsoft Project, то ассигнования могут быть запланированы условно
Учет зависимостей и других факторов	Если сценарии или требования к качеству спланированы с учетом ассигнований на исправление дефектов, убедитесь, что ключевые разработчики, имеющие большое количество неисправленных ошибок, обладают достаточным резервом времени для исправления ошибок с самым высоким приоритетом. Проверьте еще раз предстоящие задачи, чтобы убедиться в наличии резерва. Переназначьте задачи или дефекты для обеспечения исправления дефектов
Проведение собрания, посвященного началу итерации	Завершите разработку плана итерации проведением собрания, на котором будет изложена информация о предстоящей итерации. На собрании должны присутствовать все участники команды, чтобы еще раз проверить задачи итерации

Операция: Составление графика сценария

Сценарий разрабатывается и проходит первоначальное тестирование в рамках определенной итерации. В плане итерации отражено текущее понимание того, что в ней должно быть выполнено. План должен быть завершен до начала итерации. Первоначальный план создается на основе оценок, а затем уточняется по мере того, как сценарии и требования к качеству разбиваются на задачи. С каждой задачей связаны оценка трудоемкости и ответственные разработчики. Если позволяет бюджет итерации, для завершения ее планирования может быть проведено посвященное этому собрание. Оно позволяет всем заинтересованным участникам команды собраться вместе для обсуждения возникших вопросов и завершения плана итерации.

Создание плана начальной итерации	Совместно с бизнес-аналитиком разработайте план начальной итерации
Уточнение плана начальной итерации	Когда сценарий написан и задачи по разработке включены в план итерации, общее количество запланированных работ не должно превышать средние возможности команды разработчиков

Проведение совещания, посвященного началу итерации	Завершите разработку плана итерации проведением собрания, на котором будет изложена информация о предстоящей итерации. На собрании должны присутствовать все участники команды, чтобы еще раз проверить задачи итерации

Операция: Составление графика реализации требований к качеству

Для реализации требований к качеству должен быть составлен график работ. Требования к качеству описывают нефункциональные требования, такие как, например, уровень производительности. С некоторыми требованиями не связаны задачи. Однако требования могут быть связаны со сценариями, в которых они реализуются. Таким образом, требования к качеству должны быть запланированы одновременно с соответствующими сценариями, чтобы обеспечить выполнение задач, связанных с их описателями.

Создание плана начальной итерации	Совместно с бизнес-аналитиком разработайте план начальной итерации. Получите последнюю версию списка требований к качеству с расставленными приоритетами и оцененными трудозатратами. Совместно с бизнес-аналитиком выберите те из требований, которые укладываются в план итерации с учетом средней производительности команды, показанной в предыдущей итерации, и количества сценариев и ассигнований на исправление дефектов, запланированных на данную итерацию
Уточнение плана начальной итерации	Когда задачи по разработке будут включены в план итерации, убедитесь, что общее количество запланированных работ не превышает средние возможности команды разработчиков
Проведение собрания, посвященного началу итерации	Завершите разработку плана итерации проведением собрания, на котором будет изложена информация о предстоящей итерации. На собрании должны присутствовать все участники команды, чтобы еще раз проверить задачи итерации

Разработка архитектуры решения

Хорошая архитектура — это ясная и простая внутренняя структура большинства элементов приложения. Простая архитектура понижает сложность приложения. Архитектура может определять такие структурные элементы, которые позволяют проще модифицировать приложение в случае изменения

требований, и благодаря которым участки приложения могут разрабатываться независимо. Кроме того, в хорошей архитектуре используются преимущества от разбиения по уровням для увеличения надежности и уменьшения времени вывода приложения на рынок. Если имеются технологические риски, для их уменьшения можно применять разработку архитектурных моделей, что помогает лучшему пониманию системы. И, наконец, безопасность и производительность — это вопросы архитектуры. Работа над ними должна проводиться с учетом всей системы.

Операции:

Разделение системы на подсистемы	Выберите архитектурные шаблоны
Определение интерфейсов	Разработайте интерфейсы
Разработка модели угроз	Создайте обзор приложения
Разработка модели производительности	Проведите обзор требований к качеству
Создание архитектурной модели	Изучите риски
Создание архитектуры инфраструктуры	Разработайте архитектуру инфраструктуры

Операция: Разделение системы на подсистемы

Приложения, являющиеся частью распределенной системы, содержат логически связанные фрагменты кода и совместно реализуют поведение целой системы. Разделение системы на модули дает массу преимуществ. Оно позволяет снизить общую сложность, инкапсулировать функции, увеличить потенциальные возможности повторного использования подсистем и создать логические единицы для разработки. При этом отпадает необходимость реализовывать всю систему сразу.

Для создания гибкой архитектуры в MSF применяется *создание временного прототипа* (shadowing). *Приложение-прототип* (shadow) позволяет применить принцип нисходящего проектирования в любой итерации. В начале итерации, когда существует лишь проект, прототип опережает рабочий программный код. В этот период проектные конструкции не синхронизованы с кодом. В рамках прототипа выполняются все изменения архитектуры или дизайна, которые необходимы для предохранения основного кода от превращения его в «дымоход», «спагетти-код» и прочих проблем, связанных с плохим проектированием. По мере реализации элементов *начального предваряющего прототипа* (leading shadow), архитектура все больше

соответствует рабочему коду. Те части системы, которые были спроектированы, но не были реализованы, теперь становятся действительными. Когда архитектура соответствует рабочему коду, прототип называется *завершающим* (trailing shadow). Это сумма конструкций из всех предыдущих итераций.

Чтобы избежать чрезмерной детализации архитектуры, рекомендуется сконцентрироваться на уровне компонентов и устанавливаемых элементов. Добавьте приложения, необходимые для поддержки функциональных возможностей, предусмотренных в запланированных сценариях, требованиях к качеству и в намеченных для исправления дефектах. В начале итерации добавьте задачи по переработке кода, чтобы он соответствовал измененной архитектуре. Если необходимо, проверьте новую структуру при помощи архитектурных моделе.

Выбор шаблона архитектуры	Если предполагается развертывание системы в рамках существующей инфраструктуры, изучите логическую диаграмму центра обработки данных для того, чтобы понять технологию развертывания. Если такой диаграммы не существует, создайте ее
Создание приложений	В рамках диаграммы приложения создайте предваряющий прототип или его эквивалент для выбранной топологии системы. Создайте диаграмму системы на основе приложений-прототипов и добавьте прокси для всех нереализованных точек входа
Проверка развертывания	После того как диаграмма приложения стала соответствовать архитектуре в данной итерации и определена целевая инфраструктура, пора проверить развертывание. На диаграмме приложений выберите определенный вариант установки и отобразите новые приложения на соответствующие им логические серверы
Создание задач, реализующих изменения в архитектуре	Создайте новые задачи, в рамках которых будет реализован предваряющий прототип и выполнены необходимые переработки существующих функциональных возможностей в новое приложение

Операция: Определение интерфейсов

Разработку системы можно распараллелить, если до начала разработки определены все интерфейсы. Зачастую в начале итерации имеется еще недостаточно деталей, что мешает полному пониманию, необходимому для разработки элементов интерфейса. Поэтому разработку интерфейсов стоит откладывать до того

момента, пока она не станет абсолютно необходимой: ведь даже после определения интерфейсов могут происходить изменения в проекте.

Проработка интерфейсов помогает при интеграции систем, а также при изменении подходов в реализации задач разработки. В диаграмме приложений следует отображать внешние интерфейсы системы, а также внутренние интерфейсы высшего уровня.

Проектирование интерфейса	Определите интерфейсы для сценария или требования к качеству. Проверьте требования по взаимодействию с внешними системами
Определение интерфейсов веб-сервисов	Выберите конечную точку веб-сервиса на диаграмме приложения. Определите операцию и снабдите ее необходимой информацией, такой как имя, возвращаемый тип и параметры. Убедитесь, что в свойствах правильно настроены язык и путь к генерируемым файлам. Сгенерируйте заглушку для нового веб-сервиса
Создание интерфейсов классов	Совместно с разработчиками определите методы классов, которые будут обеспечивать интерфейсы между разработчиками. Применяя диаграмму классов, проверьте согласованность классов и новых методов

Операция: Разработка модели угроз

В модели угроз документируются известные угрозы безопасности, и описывается, как на них следует реагировать. Моделирование угроз является частью структурного подхода, позволяющего определить опасности, с которыми вероятнее всего столкнется система, а также степень их воздействия. В рамках целостной модели угрозы рассматриваются в порядке убывания представляемой опасности, а также рассматриваются необходимые меры противодействия им. Модель угроз следует создавать как можно раньше, а потом, по мере развития архитектуры, уточнять ее. Совместно с бизнес-аналитиком проведите обзор угроз и разработайте требования к безопасности.

Создание обзора приложения	Определите сценарии, относящиеся к задачам обеспечения безопасности. Внимательно изучите каждый сценарий на предмет возможности отступления от бизнес-правил. Постарайтесь найти уязвимости в сценариях, относящиеся к задачам обеспечения безопасности

Разбиение приложения	Определите границы доверия в рамках вашего приложения. На логической диаграмме центра обработки данных создайте новую зону, которая будет отражать эти границы. Для каждой подсистемы определите, являются ли надежными входящие потоки данных и пользовательский ввод. Если нет, то решите, как они должны аутентифицироваться и авторизоваться
Определение угроз	Для каждого ресурса или функции определите затрагивающие их реализованные сценарии. На их примере обсудите, как система должна использоваться, а как — нет
Определение уязвимостей	Убедитесь, что все части системы были рассмотрены и что все известные типы угроз были проанализированы

Операция: Разработка модели производительности

Моделирование производительности — это процесс, помогающий определить потенциальные проблемы с производительностью в приложении и найти решение для этих проблем. Моделирование производительности проводится на основе требования к качеству, которое, в свою очередь, разбито на задачи. Каждая задача имеет свой бюджет, предназначенный для решения вопросов производительности во время реализации.

Обзор требований к качеству	Определите сценарии, связанные с требованиями к производительности
Определение объема работы	По списку требований к производительности определите объем работ для приложения
Определение необходимого уровня производительности	Используя оценки объема работ и список требований к качеству, определите необходимый уровень производительности для каждого ключевого сценария. Он включает такие характеристики, как время отклика, пропускную способность и используемые ресурсы
Определение бюджета для решения вопросов производительности	Определите объем ресурсов, влияющих на производительность, который позволит добиться необходимой производительности. Примерами таких ресурсов являются время исполнения и пропускная способность сети
Распределение бюджета	Распределите ресурсы между технологическими операциями для каждого сценария

Оценка бюджета	Найдите бюджетные ассигнования, для которых существует риск нехватки для достижения необходимой производительности
Проверка модели	Выявите сценарии, на которые не выделены бюджетные ассигнования

Операция: Создание архитектурной модели

Создание архитектурной модели может значительно снизить имеющиеся риски. Важно как можно раньше обратить внимание на риски в проекте и, тем самым, учесть их при принятии стратегических и архитектурных решений, пока изменения основных компонентов архитектуры еще не требуют больших затрат. Создание ранних прототипов снижает общие риски и неопределенности в проекте. Это, в свою очередь, позволяет более точно выполнять планирование и оценки в последующих итерациях. Модели могут быть временными и отбрасываться, как только они отвечают на поставленные вопросы, или могут быть основой для архитектуры приложения.

Изучение рисков	Определите элементы, которые могут помочь в определении риска или принятии архитектурного решения
Планирование	Определите необходимый вид модели
Построение и работа модели	Постройте модель. Сосредоточьтесь на решаемой проблеме. Убедитесь, что модель должным образом описывает исследуемый вопрос

Операция: Создание архитектуры инфраструктуры

Архитектура инфраструктуры определяет окружение, в котором будет работать приложение. Хорошо разработанная архитектура инфраструктуры гарантирует установку приложения в соответствии с требованиями к качеству. Начинайте разработку архитектуры инфраструктуры, как только появляется представление о будущем приложении. При необходимости изменяйте разработку по мере того, как улучшается понимание подсистем в рамках общей архитектуры. Для построения архитектуры инфраструктуры используйте логическую диаграмму центра обработки данных.

Создание архитектуры инфраструктуры	Создайте логическую диаграмму центра обработки данных. Работайте с операциями, если они применимы, для лучшего понимания рабочего окружения. В логическом центре обработки данных создайте логические серверы, отражающие физическое окружение. Проверьте получившуюся диаграмму

Сопоставление архитектуры решения	Отобразите участки диаграммы приложения на логическую диаграмму центра обработки данных. Проверьте правильность этого отображения. Проверьте диаграмму приложения при помощи логической диаграммы центра обработки данных
Корректировка архитектуры решения	По мере определения новых подсистем выявляйте необходимые изменения в архитектуре инфраструктуры

Формулирование концепции проекта

Перед запуском проекта необходимо четко сформулировать его концепцию. Формулирование и донесение центральной концепции является самым важным элементом в поддержании направленности проекта. В течение жизни проекта его концепция может меняться под воздействием внеш-них или внутренних факторов. Если подобные изменения произошли, то важно сразу изменить проект согласно новой концепции. Концепция дает представление о будущих пользователях. Способы использования и задачи этих пользователей должны быть выявлены при помощи *собирательных образов* (personas). Концепция дает понять, привязан ли проект ко времени или к реализуемому функционалу. Сама концепция и связанная с ней деятельность создают надежный фундамент, на котором будет строиться весь проект.

Операции:

Написание концепции	Обобщите всю известную о проекте информацию
Определение собирательных образов	Определите группы пользователей
Уточнение собирательных образов	Определите характер отличий между собирательным образом и пользователем

Операция: Написание концепции

Концепция — это короткая и точная формулировка цели создания новой системы или улучшения существующей. Она дает команде представление о проекте и его движущих факторах, а также содержит обоснование разработки системы. В концепции должно присутствовать высокоуровневое описание пользователей системы. Концепция должна также содержать описание потребностей пользователей и способов их удовлетворения разрабатываемым приложением. Наконец, концепция определяет, привязан ли выпуск системы к срокам или к функциональным возможностям. Концепция должна быть на-

писана языком, понятным будущим пользователям. Написание сжатой и предметной концепции является сложной задачей.

Обобщение исходных данных проекта	Напишите один-два абзаца о контексте проекта. В них должна быть описана предпосылка создания новой системы или улучшения существующей
Объяснение движущих факторов	Опишите основные требования или выделенный интервал времени, которые управляют выпуском продукта. Будьте кратки
Определение пользователей	Определите тех пользователей, которые получат наибольшую пользу от системы
Определение основных достоинств	Опишите предполагаемые достоинства новой или улучшенной системы

Операция: Определение собирательных образов

Собирательный образ — это вымышленный персонаж, представляющий группу пользователей. Собирательные образы используются для изучения потребностей целого сегмента пользователей путем рассмотрения характеристик одного вымышленного персонажа. Для определения собирательного образа изучите сегмент пользователей, взаимодействующих с системой, соберите их опыт, умения и цели, которые они разделяют. На основе этих данных создайте вымышленный персонаж, представляющий данный сегмент. Собирательный образ является также инструментом, с помощью которого можно получить информацию о пользователях, упомянутых в концепции. Собирательные образы используются при создании сценариев и при проведении исследовательских тестирований.

Определение ролей	Среди пользователей, определенных в концепции, выберите группу пользователей, взаимодействующих с системой
Создание собирательного образа	Для создания собирательного образа соберите реальные данные. Используйте данные, полученные в исследованиях удобства использования и при посещениях клиентов, в работе с фокус-группами и при маркетинговых исследованиях, чтобы убедиться, что собирательный образ представляет реальных пользователей

Операция: Уточнение собирательных образов

Собрания и обзоры часто помогают выявить новые детали о способах использования и уровне знаний. Используйте такие собрания для проверки

собирательных образов. Периодически уточняйте их, чтобы они соответствовали текущей целевой аудитории системы. Если в собирательных образах происходят изменения, сообщайте об этом заказчику.

Определение отличий	Выявляйте сходства и различия в уровне знаний, способах использования, взаимодействии и задачах, имеющиеся между собирательным образом и реальным пользователем. Если нашлись такие отличия, разберитесь, является ли пользователь тем, кого описывает собирательный образ
Обновление и передача изменений	Добавьте новые собирательные образы и измените существующие

Разработка требований к качеству

Требования к качеству используются для описания нефункциональных требований или ограничений на функциональные возможности системы. Требования к качеству должны быть точными и не быть субъективными. Они выявляются, им присваиваются приоритеты и — если они запланированы на текущую итерацию — документируются.

Определение требований к качеству путем «мозгового штурма»	Определите цели требований к качеству
Создание моментальных снимков деятельности собирательного образа	Определите характерный промежуток времени действий собирательного образа
Определение приоритетов в списке требований к качеству	Определите важность каждого требования
Написание требований к качеству	Задокументируйте требования к качеству
Определение требований безопасности	Задокументируйте требования к системе безопасности

Операция: Определение требований к качеству путем мозгового штурма

Проведите коллективное обсуждение требований к качеству, проанализируйте каждый сценарий и определите, какие взаимодействия в них должны сопровождаться требованиями к качеству. Во время обсуждения выявите аспекты системы, для которых в сценариях были пропущены необходимые требова-

ния к качеству. Список требований к качеству содержит нефункциональные требования к приложению, он же является списком ограничений функциональных возможностей системы. Бизнес-аналитик каждый раз переоценивает и корректирует список требований к качеству, когда в процессе тестирования выявляются новые требования или когда появляются изменения в проекте.

Определение необходимых уровней качества	В папке требований *Проводника команды* (Team Explorer) откройте список требований к качеству. Он связан с проектом. Если необходимо, импортируйте описатели требований к качеству, которые были созданы непосредственно в Проводнике команды
Определение требований к качеству	Проанализируйте сценарии на предмет соответствия желаемым стандартам качества всех моментов, связанных с взаимодействием пользователя и системы

Операция: Создание моментальных снимков деятельности

Моментальные снимки деятельности — это необязательные средства, которые используются совместно с моделью собирательного образа. При написании сценариев моментальные снимки помогают конкретизировать некоторые детали. Моментальный снимок деятельности — это описание одного дня или короткого периода в жизни собирательного образа. При этом всегда описывается текущее состояние, без применения предлагаемой технологии, что помогает продемонстрировать, чем новая система или продукт будут полезны для пользователя. Работая с конкретными примерами деятельности, проще оценить пользу от каждого сценария и назначить им на основании этого приоритеты.

Создание прототипа рабочего дня собирательного образа	Для каждого собирательного образа выделите один или несколько периодов времени в их жизни
Поиск технологических возможностей	Изучите каждый день собирательного образа и выявите места, где применение продукта оказывается полезным пользователю за счет решения каких-либо его задач или расширения возможностей

Операция: Определение приоритетов в списке требований к качеству

Приоритеты в списке требований к качеству определяются, исходя из их важности и оказываемого влияния на пользователя. Требования, которые наиболее важны

пользователю, должны реализовываться в первую очередь. Список требований к качеству, упорядоченных по приоритетам, включается в план итерации. Тот, в свою очередь, используется при планировании разработки для повышения эффективности использования ресурсов. При добавлении или удалении требований, а также при изменении потребностей пользователей следует заново определить приоритеты списка требований к качеству.

Определение общего приоритета	Каждому сценарию в списке назначьте общий приоритет. В поле приоритета списка сценариев впишите соответствующее числовое значение
Описание приоритетных сценариев	Используя шаблон описания сценария, контурно обрисуйте наиболее приоритетные сценарии, снабдив их кратким описанием, но, в то же время, достаточно подробным, чтобы разработчики на его основании смогли сделать оценки
Передача запросов для выполнения оценок	Пошлите разработчикам запрос на выполнение общих оценок для самых приоритетных требований
Разбиение требований или изменение приоритетов	Если цена какого-либо требования к качеству превышает бюджет итерации, попробуйте разбить это требование

Операция: Написание требований к качеству

Требования к качеству используются для описания нефункциональных требований или ограничений на функциональные возможности системы. Требования к качеству должны быть точными и не быть субъективными. Их описание должно предоставлять достаточно информации разработчикам, чтобы те могли спроектировать и запрограммировать функциональные возможности, удовлетворяющие этим требованиям.

Документирование требования к качеству	Опишите требование к качеству в поле описания его описателя. При разработке требований к качеству используйте следующий языковой шаблон: «контекст», «воздействие», «ответ». Требование к качеству должно быть таким, чтобы его можно было протестировать. Проектирование оставьте архитекторам и разработчикам
Прикрепление вспомогательной информации	К требованию к качеству прикрепите связанные с ним сценарии, если таковые имеются. Если требование к качеству затрагивает все сценарии или большинство из них, просто укажите это вместо того, чтобы прикреплять их все

Операция: Определение требований безопасности

Требования к системе безопасности определяют уровень, до которого система будет защищать себя и ресурсы. Это могут быть конфиденциальные данные, требования закона или нематериальные активы, такие как репутация компании, торговые секреты или интеллектуальная собственность. Требования к системе безопасности должны быть конкретны, а для защищаемых ресурсов должна быть возможность проверки защиты. Способ защиты описывать не нужно. Требования к системе безопасности обычно формулируются предложениями, начинающимися с глагола, например: «Предотвратить доступ неавторизованных пользователей к учетным записям клиентов». Защищаемые ресурсы должны быть четко определены.

Документирование требований к системе безопасности	Опишите требования к системе безопасности в поле описания описателя требования к качеству. Убедитесь, что требования к системе безопасности обращены на конкретные ресурсы и могут быть протестированы. Проектирование оставьте архитекторам и разработчикам
Прикрепление вспомогательной информации	Прикрепите ссылки на законодательные акты, если они имеются. Прикрепите все сценарии, относящиеся к задачам обеспечения безопасности. Если требование к системе безопасности затрагивает все сценарии или большинство из них, просто укажите это вместо того, чтобы прикреплять их все

Создание сценария

В сценариях формулируются функциональные задачи системы. Для определения этих задач внимательно изучите все потребности собирательных образов системы. Эти задачи могут быть сначала просто выписаны в виде списка, а позднее описаны в виде сценариев. Старайтесь не допускать сценариев, описывающих неудачные или неоптимальные пути поставленных задач. Сценарии могут создаваться во время «мозгового штурма», при помощи моментальных снимков или пробных тестов. Все они добавляются в список сценариев. Когда сценарии назначаются на следующую итерацию, они документируются и им присваиваются приоритеты. Создание сценариев заканчивается, когда составлен график реализации всех сценариев в итерациях, либо после создания архитектурного прототипа.

Операции:

Работа над сценариями методом «мозгового штурма»	Определите задачи системы

Создание моментальных снимков деятельности	Выясните типичные способы использования системы
Определение приоритетов в списке сценариев	Определите приоритеты для каждого сценария
Создание описания сценария	Выберите подходящий обобщенный образ
Раскадровка сценария	Подберите инструмент для раскадровки

Операция: Работа над сценариями методом «мозгового штурма»

Первоначально список сценариев состоит из элементов сценариев. Их используют для определения целей приложения. Элемент сценария состоит из имени и краткого описания, определяющего уникальность элемента. Элементы становятся описателями после синхронизации сценариев. Проведите коллективное обсуждение и добавьте сценарии в список сценариев, проанализируйте задачи приложения и то, как оно будет использоваться каждым собирательным образом. Исходный список сценариев создается в первой итерации, а затем каждый раз, когда меняются требования или выявляются новые, список сценариев следует пересматривать и корректировать.

Определение задач системы	В папке требований Проводника команды откройте список сценариев. Список сценариев связан с проектом. Импортируйте все сценарии, созданные при помощи Проводника команды
Формулировка сценариев	Выберите цель и рассмотрите разные способы, которыми собирательный образ может достичь ее или потерпеть в этом неудачу. Для способа, которым собирательный образ пытается достичь цели, выберите описательное имя. Затем добавьте элемент сценария с этим именем в список сценариев. При определении концепции проекта следует убедиться, что представление о будущем приложения охватывает все сценарии. При изменении представления следует корректировать концепцию

Операция: Определение приоритетов в списке сценариев

При определении приоритета сценария в списке в расчет принимаются как его значимость для пользователей, так и для приложения в целом. Приоритеты определяют очередность реализации сценариев. Сам процесс опреде-

ления приоритетов помогает выявить наиболее важные и ценные сценарии, которые будут реализованы в ближайшей итерации.

Определение общего приоритета	Каждому сценарию в списке назначьте общий приоритет. В поле приоритета списка сценариев впишите соответствующее числовое значение
Описание приоритетных сценариев	Используя шаблон описания сценария, контурно обрисуйте наиболее приоритетные сценарии, снабдив их кратким описанием, но, в то же время, достаточно подробным, чтобы разработчики на его основании смогли сделать оценки
Передача запросов для выполнения оценок	Пошлите менеджеру проекта и разработчикам запрос на выполнение общих оценок для самых приоритетных требований
Разбиение требований или изменение приоритетов	Если цена какого-либо сценария превышает бюджет итерации, попробуйте разбить это требование

Операция: Формулировка описания сценария

Описание сценария из списка пишется в процессе создания начального плана итерации, т. е. когда она становится кандидатом на реализацию в ближайшем сценарии. Уровень детализации должен быть достаточным для передачи потребностей собирательного образа разработчикам и для создания контрольных примеров. Большие сценарии могут быть разбиты на более мелкие, укладывающиеся в рамки одной итерации. После создания описания оно публикуется на портале проекта, а будущим разработчикам и тестировщикам рассылаются сообщения, чтобы они просмотрели их.

Выбор подходящего собирательного образа	Из списка сценариев выберите сценарий, вошедший в ближайшую итерацию или важный с точки зрения архитектуры. Откройте шаблон описания сценария в Microsoft Word и сохраните документ, дав ему имя сценария, чтобы отличать его от остальных, уже написанных сценариев
Формулировка описания сценария	Пишите сценарий в разделе описания соответствующего документа. С самого начала описывайте каждое действие, выполняемое собирательным образом в процессе достижения поставленной цели. Действия, уже описанные в других сценариях, записывайте схематично, а в тех местах, где сценарий отличается от остальных, будьте наиболее подробны

Разбиение сценариев	Если детальная оценка (составленная как сумма задач разработки) превышает длину итерации, то такой сценарий является кандидатом на разбиение. Получившиеся в результате разбиения подсценарии в сумме должны соответствовать исходному сценарию. Для каждого нового сценария следует определить характерные отличия от других. На основе шаблона сценария создайте хотя бы два документа, описывающие сценарии

Операция: Раскадровка сценария

Раскадровка улучшает сценарий и снабжает его информацией о пользовательском интерфейсе. Она исключительно полезна, когда пользовательский интерфейс играет важную роль в удовлетворении потребностей внешних пользователей. Раскадровка также помогает подогнать проект под требования разных организаций. Она может включать различные графические элементы, такие как снимки экранов, макеты приложений, диа-граммы последовательности экранов. Она также может включать краткое описание поведения в системе собирательных образов. Графическое представление сценариев значительно улучшает взаимопонимание между конечными пользователями и разработчиками. Если раскадровки созданы для каждого отдельного сценария, то их можно объединить в общий обзор приложения.

Выбор сценариев	Отберите сценарии, требующие раскадровки
Создание снимков экрана	Для раскадровки хорошо подходят такие инструменты, как Microsoft Visio и Microsoft PowerPoint. Создайте экраны, которые должен видеть собирательный персонаж. Схематично нарисуйте пользовательский интерфейс, требуемый сценарием. Совместно с командой разработчиков убедитесь, что полученный пользовательский интерфейс не противоречит выбранной метафоре
Создание диаграммы последовательности экранов	Прямоугольниками изображайте экраны, а стрелками — их последовательность

Ведение проекта

Ведение проекта — это постоянный мониторинг и документирование его состояния. Чтобы обеспечить постоянный контроль над проектом, допускается внесение необходимых изменений. Необходимо отслеживать как ход

проекта в целом, так и выполнение отдельных итераций. Надо своевременно выявить дефекты, определить риски и присвоить им приоритеты.

Операции:

Обзор целей	Установите минимальный приемлемый уровень
Оценка хода выполнения	Поддерживайте график проекта в актуальном состоянии
Оценка пороговых значений показателей тестов	Проанализируйте отчеты о выполненных тестах
Классификация дефектов	Создайте список дефектов, подлежащих классификации
Определение риска	Определите требования к качеству, связанные с большими рисками

Операция: Оценка хода выполнения

Для успешной реализации проекта важно отслеживать и оценивать ход его выполнения. Правильное планирование и управление проектом зависит от возможности оценивать его текущее состояние. Кроме того, надо отслеживать организационные, технические и проектные риски. Проект можно привязать ко времени или к реализуемому функционалу. В привязанных ко времени проектах крайние сроки зачастую определяются событиями на рынке, положением конкурентов, существующими нормами, финансовыми факторами и другими причинами, связанными с бизнесом. Даже проекты с фиксированным функционалом могут иметь желательную дату реализации. В проектах, не фиксированных по времени, правильный прогноз даты завершения благотворно скажется на уровне продаж, маркетинговых показателях, на внедрении, обучении и эксплуатационных факторах.

Оценка списка обязательных работ	Если есть предполагаемая дата реализации проекта, определите число необходимых итераций на основании длительности начальной итерации. Используйте данные из *отчета об общем темпе проекта* (velocity report) для вычисления количества человеко-дней для итерации. Произведение двух этих значений даст объем оставшихся трудозатрат, выраженный в человеко-днях
Проверка зависимостей	Проверьте зависимости в графике и убедитесь, что они соблюдаются

Анализ осуществимости	При необходимости оцените число итераций, необходимых для реализации планируемых функций, и определите, соответствует ли ход работ целям, определенным в концепции проекта

Операция: Оценка пороговых значений показателей тестов

В документе с описанием подходов к тестированию указаны пороговые значения для показателей тестов. Необходимо периодически анализировать продвижение проекта в соответствии с этими показателями и при необходимости их корректировать. Эффективность тестов, число неисправностей, охват кода, подробные отчеты о выполненных тестах — все это поможет контролировать выполнение проекта. Затем, при необходимости, можно скорректировать курс.

Анализ отчетов о выполненных тестах	Отчеты о выполненных тестах помогут понять текущее положение дел, если сравнить результаты с пороговыми значениями
Определение задач и рисков тестирования	Совместно со специалистами по тестированию создайте и распределите задачи тестирования

Операция: Определение риска

Своевременно необнаруженный и неконтролируемый риск может отрицательно сказаться на выполнении проекта. Риски имеют место в различных областях. Например, в одном проекте риск может быть связан с эргономикой, поскольку эта область принципиально важна для заказчика, а в другом проекте риски могут быть связаны с производительностью. Важно четко определить и учесть наиболее опасные риски в первых же итерациях проекта. По мере появления дополнительных сведений о сценариях и требованиях к качеству в каждой последующей итерации уделите время поиску новых возможных рисков.

Определение требований к качеству, связанных с риском	Проанализируйте требования к эргономике. Определите, является ли интерфейс пользователя ключевым фактором оценки приложения заказчиком
Проверка сценариев	Проанализируйте сценарии на предмет потенциальных рисков

Вопросы интеграции	Какие библиотеки сторонних поставщиков могут использоваться приложением? Проверены ли они? Если нет, выявите неизвестные характеристики библиотек, которые необходимо проверить
Формулирование рисков и расстановка их приоритетов	Если в рассмотренных областях обнаружены угрозы для успешной реализации проекта, создайте описатель риска. Сформулируйте потенциальные проблемы, связанные с данным риском, и задокументируйте их в описателе

Операция: Обзор целей

После всех итераций, кроме нулевой, продукт должен быть в стабильном состоянии и готовым к поставке. Установите показатель *минимального приемлемого уровня* (minimum acceptance level), чтобы в дальнейшем сравнивать реализованные сценарии и требования к качеству с этим уровнем. Минимальный приемлемый уровень должен постоянно переоцениваться с учетом изменений потребностей заказчика, состояния рынка и т. д. Минимальный приемлемый уровень обновляется после каждой итерации.

Установка минимального приемлемого уровня	Оцените ожидания, которые связывают с разрабатываемым продуктом заказчики, и рынок в целом. Примите в расчет конкурентов, существующие аналоги и потребности бизнеса
Отчет о текущем состоянии	Сформируйте отчет об оставшихся работах для оценки хода выполнения проекта

Операция: Классификация дефектов

Классификация (triage) — процесс анализа вновь обнаруженных или заново обрабатываемых дефектов, назначения им приоритетов и привязки к определенным итерациям. Классификацию проводит менеджер проекта с участием других участников проектной группы.

Создание отчета о дефектах	Выдайте запрос системе отслеживания дефектов о вновь обнаруженных и заново обрабатываемых дефектах
Анализ дефектов	Ознакомьтесь с каждым обнаруженным дефектом, привлекая разработчиков, тестировщиков и менеджера проекта. Определите важность исправления каждого из них до конца проекта и текущей итерации

Присвоение приоритета и привязка к итерации	Каждому дефекту присвойте приоритет и определите, в какой итерации его необходимо исправить, чтобы его обработку можно было учесть в графике проекта
Воспроизведение непонятных дефектов	Если отчет о дефекте невразумителен или нет уверенности в истинном действии дефекта, группа управления программой должна попробовать воссоздать дефект, чтобы лучше понять его реальное влияние и возможности повторного появления

Сборка продукта

В процессе сборки создается единый продукт, включающий существующий код и все новые пакеты изменений. Однако вносимые изменения должны быть корректны. Пакет изменений должен быть скомпилирован и должны быть выполнены компоновочные тесты. Добавляемый пакет изменений не должен провоцировать проблемы — проверка этого факта является частью процесса приемки сборки. К каждой сборке прилагается набор комментариев, поясняющих вносимые изменения.

Операции:

Запуск сборки	Минимизируйте зависимости
Проверка сборки	Проверьте основные функции
Исправление сборки	Выделите ошибки компиляции
Приемка сборки	Протестируйте сборку

Операция: Запуск сборки

Чтобы гарантировать целостность, запускайте сборку продукта всякий раз, когда требуется включить в него изменения. Это помогает синхронизировать изменения и предоставляет всем разработчикам рабочую платформу. Чтобы ускорить слишком медленную сборку, пересмотрите зависимости.

Минимизируйте зависимости	Чем меньше в приложении зависимостей, тем быстрей происходит его сборка
Начало сборки	Запускайте сборку всякий раз, когда можно интегрировать изменения. Создавайте «чистую» сборку только в случае абсолютной необходимости. «Чистые» сборки делаются по ночам. Если процесс сборки не выполняется без предварительной «чистой» сборки, проверьте зависимости

Операция: Проверка сборки

Необходимо поддерживать качество приложения при внесении любого изменения. Проводите проверочные испытания, демонстрирующие, что базовая функциональность сборки не затронута. Проверочные испытания также проводятся для проверки изменений, сделанных в процессе исправления дефекта и для контроля работоспособности вновь добавленных функций.

Проверка основных функций	Выполните минимальное подмножество тестов, в частности, проверочный тест сборки, иногда называемый «дымовым», чтобы убедиться в стабильности базовой функциональности системы
Проверка измененных функций	Запустите дополнительные тесты и убедитесь, что все запланированные пакеты изменений добавлены и изменения корректны
Информирование участников о готовности сборки	Если все тесты прошли, проинформируйте все заинтересованные стороны, что сборка допущена к работе

Операция: Исправление сборки

В процессе проверочных испытаний может возникнуть ситуация, когда исходный код не компилируется на сборочной машине или внешние компоненты не работают корректно. Чтобы исправить сборку, выявите ошибки периода компиляции и выполнения и проинформируйте разработчиков, ответственных за соответствующие компоненты. Задержки в решении проблем со сборкой тормозят выполнение других работ, например тестирования.

Выделение ошибок компиляции	Перед компиляцией запустите сценарий, очищающий используемые при сборке каталоги от ненужных файлов. Компиляция исходного кода и построение компонентов на сборочной машине может не проходить, хотя та же процедура нормально проходила на машине разработчика
Обработка ошибок периода выполнения	Фиксируйте все ошибки периода выполнения и информируйте о них разработчика — напрямую и посредством описателя дефекта
Проверка содержимого	Сравните содержимое сборки с указанным в спецификации: нет ли лишних файлов и не пропущены ли нужные
Проверка установочных сценариев и исполняемых файлов	Просмотрите установочные сценарии и проверьте, расположены ли необходимые файлы в каталогах, указанных в спецификации и сценариях

Проверка версий связанных модулей	Проверьте версии используемых при компоновке внешних модулей, например библиотек сторонних поставщиков. Использование некорректных версий может привести к непредсказуемому поведению приложения
Исправление и перекомпоновка сборки	После решения проблемы нужно сразу заново создать сборку, чтобы не блокировать работу других участников проекта

Операция: Приемка сборки

Когда сборка допускается к работе, это говорит о том, что реализованы некоторые минимальные требования и сборка готова к дальнейшей эксплуатации. Для приемки сборки убедитесь в том, что компиляция и компоновка проходят без сбоев, а проверочные тесты покрывают всю базовую функциональность. Добавьте к сборочным тестам проверки, сделанные в последней итерации.

Поддержка сборочных тестов	Добавляйте или удаляйте разделы сборочных тестов, чтобы они обеспечивали адекватную проверку сборок
Наблюдение за сборкой	Регулярно просматривайте отчеты о деталях сборки

Выпуск продукта

В определенный момент набор реализованных функций становится достаточным для выпуска продукта. Существует несколько типов выпусков: альфа-, бета-версия и окончательная версия или доработанный выпуск. Этот выпуск окажет большее воздействие, чем сама команда, которая его создавала, о нем будут судить пользователи. Для программных продуктов, созданных для внешних потребителей, необходимы маркетинговые и торговые мероприятия. Системы, созданные для внутреннего применения, требуют организации обучения и поддержки эксплуатации. Чтобы заранее начать готовить соответствующих людей, составляется план выпуска. Процесс выпуска и связанный с ним план во многом зависит от типа продукта и способа его применения.

Операции:

Исполнение плана выпуска	Проверьте правильность материала
Проверка выпуска	Создайте свой раздел для выпуска

Заметки о выпуске	Задокументируйте выявленные ограничения
Развертывание продукта	Создайте установочный комплект

Операция: Исполнение плана выпуска

В плане выпуска упоминаются все заинтересованные в получении нового выпуска. Этот план позволяет подготовиться к выпуску продукта изготовителей носителей, заказчиков и группу эксплуатации. План позволяет также проинформировать лиц, ответственных за последующую работу с продуктом, о том, когда они смогут его получить и на какую техническую поддержку вправе рассчитывать.

Проверка материалов, связанных с выпуском	Проверьте, что материалы, относящиеся к маркетингу, торговле, обучению и приемке заказчиком, соответствуют составу продукта
Координация поставки продукта	Обеспечьте поставку или развертывание продукта

Операция: Проверка выпуска

Когда выполнены изложенные в концепции и в описании подхода к тестированию требования к функциям, качеству и срокам, можно рассматривать возможность выпуска продукта (в том числе альфа- или бета-версий). Чтобы сборка стала официальным выпуском, надо подвергнуть ее окончательному регрессионному тестированию и проанализировать его результаты. После завершения тестирования надо решить, исправлять ли обнаруженные дефекты или рассматривать сборку как кандидата на выпуск.

Создание своего раздела для выпуска	Выделите в иерархии проекта свой раздел для кандидата на выпуск. Это защитит код и позволит вносить только выбранные изменения
Выполнение регрессионных тестов	Выполните для кандидата на выпуск полный регрессионный тест
Документирование дефекта	Если тест не проходит, необходимо создать новый отчет о дефекте. Оцените влияние дефекта и научитесь его воспроизводить

Операция: Создание заметок о выпуске

Заметки о выпуске — это краткий набор сведений, сопровождающий каждый выпуск продукта. В этих заметках собраны требования к среде, описания установки и запуска приложения, новых возможностей, исправленных ошибок, известных дефектов и способов обхода проблем. Из заметок о выпуске пользователь узнает подробности о новой версии.

Документирование выявленных ограничений	Опишите требования среды

Операция: Развертывание продукта

Завершающий этап в создании продукта — его развертывание. Развертывание служит для сбора всех компонентов, необходимых для окончательной поставки. В некоторых случаях сюда входит запись продукта на мастер-носитель. В других — установка продукта на технологическом сервере или на веб-узле, с которого он будет загружаться пользователями. Какой бы механизм не применялся, цель одна — передача системы пользователям.

Создание установочного комплекта	Сделайте для продукта установочный комплект, который упростит установку или развертывание приложения
Распространение выпуска	Обеспечьте поставку системы потребителям

Устранение дефекта

Наличие дефекта указывает на потенциальную необходимость изменения уже работающей программы. Устранение дефекта не должно оказывать побочного эффекта. Чтобы исправление дефекта не нарушало работающий код, процесс коррекции должен быть методичным и управляемым. Код, измененный при устранении дефекта, необходимо проверить на соответствие правилам кодирования, автономно протестировать, пересмотреть, интегрировать в приложение и зарегистрировать в системе управления версиями. Все это делает ответственный за обработку дефекта участник проектной группы. Если все перечисленные действия не будут выполнены, «исправление» будет хуже исходной проблемы.

Операции:

Воспроизведение дефекта	Руководствуйтесь описанием дефекта
Создание или изменение теста модуля	Определите область охвата теста модуля
Определение причины возникновения дефекта	Выделите функциональную область
Переназначение дефекта	Измените описание дефекта
Выбор стратегии устранения дефекта	Проанализируйте обнаруженный дефект

Изменение программы	Найдите нужный код
Выполнение теста модуля	Выберите тест модуля из коллекции
Рефакторинг кода	Определите сложность
Обзор кода	Проверьте правильность имен
Интеграция изменений	Проверьте зависимости

Операция: Воспроизведение дефекта

Прежде всего, надо попытаться воссоздать дефект. Если сделать это не удается, значит описание дефекта не содержит достаточную и правильную информацию или неисправность является перемежающейся. Необходимо выяснить действительные условия проявления дефекта, выявить его и исправить.

Использование описания	Если в отчете о дефекте содержится описание последовательности его воссоздания, следуйте этой инструкции
Получение дополнительных сведений	Если воспроизвести дефект не удается, соберите больше сведений. Используйте данные тестов и предыдущие отчеты о дефектах, связанные с решением подобных проблем. Просмотрите отчеты об ошибках, с которыми работали другие разработчики или специалисты по тестированию
Отказ от обработки дефекта	Если проявление дефекта так и не удалось воссоздать, может быть принято решение прекратить обработку дефекта (перевести его в состояние «Closed») на том основании, что он не воспроизводится («Unable to Reproduce»)

Операция: Определение причины возникновения дефекта

Причину возникновения обнаруженного дефекта должен определить разработчик. Для поиска корня проблемы разработчик может применять различные тактики и средства. Стратегия устранения дефекта, как правило, оказывается достаточно очевидной после выявления причины. Для поиска хороши такие инструменты, как журналы, отладчики, листинги и пр.

Выделение функциональной области	Исключите из поиска источников проблемы области кода, не вызывающие подозрения
Трассировка подозрительного кода	Используйте визуальные возможности отладчика при поиске дефекта

Анализ системы всеми доступными средствами	Кроме трассировки применяйте все доступные средства поиска неисправностей, в том числе от сторонних производителей
Локализуйте проблему	Максимально сузьте диапазон поиска дефекта
Анализ кода	Если применение отладчика невозможно из соображений производительности или ограниченности ресурсов, проанализируйте исходный текст строка за строкой

Операция: Переназначение дефекта

Причиной переназначения дефекта может служить недостаток сведений о нем, малый опыт разработчика по устранению подобных проблем или балансировка загруженности исполнителей. При переназначении дефекта добавьте в описатель сведения о причине выполнения этой операции.

Изменение дескриптора дефекта	В бланке документирования дефекта укажите причину переназначения в поле диалога
Изменение владельца	Измените сведения о лице, ответственном за обработку дефекта

Операция: Выбор стратегии устранения дефекта

Разработчик должен выбрать оптимальный метод решения проблемы, исключающий добавление новых дефектов. При выборе способа исправления дефекта учитываются вопросы архитектуры, производительности и ресурсов. Решение должно быть нацелено на устранение проблемы. Однако для соответствия требованиям и обеспечения возможностей пользователей, решение может быть принято в ущерб каким-то функциям системы.

Анализ обнаруженного дефекта	Определите тип дефекта: проблемы с архитектурой, ресурсами или производительностью
Проблема с архитектурой	Определите, на какие архитектурные решения повлияет устранение дефекта
Проблема с ресурсами	Определите фрагменты кода, на которые оказано воздействие
Проблема с производительностью	Определите, на какие фрагменты кода повлияет устранение дефекта

Операция: Изменение программы

После выбора подхода к устранению дефекта разработчик должен внести изменения в код. При этом сборка должна остаться работоспособной, не

должно быть внесено новых ошибок, а в описатель дефекта должны быть внесены соответствующие изменения.

Получение нужного кода	Найдите все файлы, которые нужно изменить для исправления дефекта
Создание нового кода	Для исправления дефекта, возможно, придется написать код в новых файлах
Изменение существующих файлов	Для исправления дефекта может потребоваться изменение существующих файлов: добавление или удаление кода

Операция: Создание или изменение теста модуля

С помощью теста модуля проверяется корректность реализации определенного программного компонента. Выполнение тестов модулей при разработке уменьшает число дефектов, обнаруживаемых на этапе тестирования, и позволяет бороться с регрессом — появлением новых дефектов при исправлении уже выявленных или добавлении новых функций. Тесты модулей не являются тестами функций или взаимодействия; их единственное назначение — проверить автономную работу фрагмента кода. Разработчики должны убедиться в том, что существующие тесты модулей проходят после написания нового кода. Тестирование должно проводиться на протяжении всего проекта — от реализации архитектурных основ в начале до внесения дополнений в конце проекта. Есть разные типы тестовых заданий, например для проверки сценариев и требований к качеству. При этом все разновидности тестов единообразно выполняются и выявляют дефекты.

Определение типа теста модуля	Определите типы разрабатываемых тестов модулей. *Тесты правильной работы* (positive unit tests) проверяют код в нормальном режиме и контролируют корректность результатов. В *тестах неправильной работы* (negative unit tests) умышленно некорректно используется код и проверяется его устойчивость и адекватность обработки ошибок. Тесты с внесением неисправностей позволяют обнаружить аномалии в обработке ошибок
Создание или изменение теста модуля	Для каждой задачи по разработке определите необходимые тесты модулей, которыми можно проверить как можно больше функций. Напишите тест модуля, убедитесь, что он не проходит, напишите или выделите фрагмент кода путем рефакторинга и прогоните тест модуля. Повторите процедуру для всех выбранных тестов модулей

Проверка теста модуля	Прогоните тест и убедитесь, что он не проходит для незавершенных фрагментов и проходит для тех элементов, которые работают как требуется

Операция: Выполнение теста модуля

Тест модуля покрывает определенную область кода. Используя комбинацию таких тестов, разработчики и тестировщики определяют качество определенного раздела программы. Выполняйте тесты модулей после изменения исходного кода. Выполнение теста модуля подтверждает работо-способность той части программы, которая покрывается этим тестом.

Определение подходящих тестов модулей	Найдите тест или тесты, позволяющие наиболее адекватно проверить соответствующий элемент программы
Выполнение теста модуля	Прогоните тест для кода, который он покрывает
Анализ результатов теста	По завершении каждого этапа отметьте его соответствующим образом: Pass (Прошел), Fail (Не прошел), Skip (Пропущен), Warning (Требует внимания) или Blocked (Заблокирован)
Отладка кода	Исправьте ошибки в программе, относящейся к задаче

Операция: Рефакторинг кода

В процессе рефакторинга кода прогоняйте тесты модуля после каждого изменения. При таком подходе риск повреждения программы минимален. По возможности выполняйте автоматизированный рефакторинг, поскольку автоматизированные процессы минимизируют вероятность ошибок. Рефакторинг надо проводить постоянно.

Определение сложности	При добавлении новых функций, обратите внимание на те части программы, структура которых стала более сложной
Применение рефакторинга	При рефакторинге вносите за один раз одно изменение. Измените код и все ссылки на измененную область
Выполнение тестов модуля	Выполните тесты модуля, чтобы убедиться в семантической целостности кода после рефакторинга. Исправьте все неработоспособные тесты модулей

Операция: Обзор кода

Обзор кода применяется для выявления областей, с которыми могут возникнуть проблемы в процессе дальнейшей разработки или тестирования. Он также дает возможность узнать мнения других разработчиков о рассматриваемом коде. Обзоры кода позволяют участникам, работающим в той же области, следовать прецедентам, созданным предыдущими разработчиками. В таких партнерских встречах требуется присутствие второго знающего коллеги, с которым разработчик должен пройти по всем изменениям, прежде чем зарегистрировать код в системе управления версиями. Должны быть выполнены тесты модулей и проведен анализ кода. В обзорах нужно сосредоточиться на областях, которые нельзя проверить с помощью компилятора или посредством анализа кода — на производительности, читабельности, безопасности.

Проверка правильности имен	Имена классов и методов должны отражать назначение соответствующих фрагментов кода
Проверка адекватности кода	Рассматриваемый код должен соответствовать задаче, для которой он написан. Допустимы только такие изменения кода, которые добавляют или изменяют функции системы
Проверка расширяемости	Написанный код должен допускать расширяемость (если ставилась такая задача) или возможность повторного использования в других частях системы
Проверка допустимой сложности кода	Повторяющийся код должен быть собран в общих функциях
Проверка сложности алгоритма	Число возможных ветвей кода должно быть минимальным. Право на существование имеют только явно необходимые ветви
Проверка безопасности кода	Проверьте защиту объектов, уровни привилегий и использование данных в точках входа. При проверке используйте контрольный список
Внесение изменений по результатам обзора	Внесите изменения, намеченные в результате обзора кода, скомпилируйте программу, выполните тесты модулей и выполните анализ кода. Если какие-либо тесты не прошли, выявите ошибки и исправьте код

Операция: Интеграция изменений

Чтобы приложение было устойчивым и согласованным, необходимо организовать интеграцию изменений в код. Когда код состоит из небольших из-

менений, каждое из которых соответствует задаче по разработке или исправлению дефекта, его проще поддерживать в стабильном состоянии. Легче найти и изолировать потенциальную проблему. После интеграции последней задачи по разработке, ответственный за реализацию соответствующего сценария или требования к качеству, разработчик проверяет всю функциональность от начала до конца. Для состояния сценария или требования к качеству устанавливается значение Resolved (Обработан) и описатель назначается тестировщику.

Проверка зависимостей	Если задача зависит от других задач, которые еще не завершены, дождитесь, когда они будут интегрированы в систему
Тестирование и интегрирование других задач по разработке	Проверьте, что вносимые вами изменения, связанные с исправлением дефекта, реализацией части сценария или требования к качеству, нормально работают совместно с уже интегрированными изменениями
Регистрация пакета изменений	Увеличьте номер в поле «Resolved in Build» (Решено в сборке) для задачи исправления дефекта или в поле «Integration Build» (Сборка с реализацией) для задачи по разработке
Завершение задачи	Если описатель, с которым связаны сделанные изменения, представляет сценарий или требование к качеству, а вы не являетесь его владельцем, уведомите владельца о том, что завершили внесение изменений

Закрытие дефекта

Оснований для закрытия дефекта может быть несколько. С закрытым дефектом в текущей итерации больше не выполняются никакие действия. Дефект может быть закрыт потому, что он исправлен, его устранение отложено до следующего выпуска продукта, он перестал быть актуальным, его невозможно воссоздать или он соответствует ранее зарегистрированному дефекту. Закрытие дефекта обычно происходит при классификации или после устранения и проверки этого факта.

Проверка исправления	Попытайтесь воссоздать дефект
Закрытие дефекта	Подтвердите, что дефект повторяет существующий

Операция: Проверка исправления

Необходимо убедиться, что дефект исправлен корректно и не нарушены другие функции. После того как разработчик исправил дефект, в дело вступает тестировщик, который отвечает за прогон тестов. Если тесты проходят успешно, дефект считается закрытым. В противном случае он заново назначается разработчику.

Попытка воссоздать дефект	Попытайтесь воспроизвести дефект, выполнив представленную в описателе дефекта последовательность действий
Неожиданное поведение	Выполните смежные функции и попытайтесь найти неожиданные моменты в поведении системы, которые могут быть связаны с дефектом. Особенно важно это в случае проверки задачи по разработке, нереализованной полностью
Получение дополнительных сведений о дефекте	Если в описателе дефекта недостаточно данных или они непонятны, обратитесь к нужному разработчику или тому, кто обнаружил и задокументировал дефект
Переназначение дефекта	Если тестовое задание не проходит, дефект снова назначается разработчику

Операция: Закрытие дефекта

Оснований для закрытия дефекта есть несколько. Зачастую дефекты исправляются разработчиками, затем помечаются как решенные и в конце закрываются менеджерами проектов. Дефект закрывается, если его исправление переносится на другой выпуск; считается неактуальным, если доказано, что его нельзя воспроизвести или подтверждено, что он дублирует существующий.

Обновление описателя повторного дефекта	Если дефект аналогичен существующему, в его описателе надо сослаться на аналог
Неустраняемые дефекты	Если дефект не будет исправлен, подробно опишите причину. Это может быть ограничение системы или соответствие дизайну
Обновление описателя закрытого дефекта	Если больше не удается добиться появления дефекта, добавьте соответствующие сведения в описатель и закройте дефект

Реализация задачи по разработке

Задача по разработке — небольшая часть деятельности разработчика, связанная с требованием к качеству или сценарием. В результате реализации задачи по разработке к системе добавляется новая функция. После завершения задачи по разработке необходимо провести тестирование модулей, обзор кода и его анализ, интеграцию кода и регистрацию его в базе кода. Затем сценарий или требование к качеству передается на тестирование.

Операции:

Оценка задачи по разработке	Оцените длительность задачи по разработке
Создание или изменение теста модуля	Определите тип теста модуля
Написание программы	Выберите компонент
Анализ кода	Определите правила приложения
Выполнение теста модуля	Выберите тест модуля из набора
Рефакторинг кода	Определите сложность
Обзор кода	Проверьте правильность имен
Интеграция изменений	Проверьте зависимости

Операция: Оценка стоимости задачи по разработке

Оценка стоимости задачи по разработке помогает ограничить набор реализуемых функций, определить расписание и распределить приоритеты. Оценку всех задач по разработке и решение всех связанных с этим проблем необходимо выполнить до совещания по планированию итерации. Если общая стоимость задач по разработке превышает затраты на итерацию, задача должна быть отложена или переназначена. После утверждения задачи за ее оценку отвечает разработчик.

Оценка на основе опыта	Проводите оценку с учетом времени выполнения аналогичных задач
Балансировка загрузки	Если в результате оценки выясняется, что объем работ превышает возможный уровень для итерации, совместно с менеджером проекта попробуйте перераспределить загрузку или перенести задачу на другую итерацию

Детализация задачи	Рассмотрите задачу по разработке с учетом других подобных задач, а также требований к качеству и сценариев, для которых еще не назначены задачи по разработке. Создайте для них задачи
Определите способы интеграции	Совместно с другими участниками группы разработки выработайте четкую картину интеграции данной функциональности с другими функциями

Операция: Написание программы

Создайте код, определите измененные области, напишите тесты модулей, проведите обзор кода и внесите в него необходимые изменения, в завершение зарегистрируйте код на сервере управления версиями. Вам также может потребоваться выполнить отладку, рефакторинг и использовать промежуточные версии исходного кода. После того как код написан, необходимо выполнить его обзор и убедиться, что его качество соответствует требованиям и принятым стандартам. Созданный код не должен оказывать вредных побочных эффектов на другие части приложения.

Выбор компонента для кода	Сопоставьте задачу с компонентом, в котором реализуется соответствующая функциональность
Создание новых классов или методов	С помощью диаграммы классов создайте классы, необходимые для реализации данной функциональности
Реализация алгоритма метода	Извлеките существующие классы, которые нужно изменить, из системы управления версиями

Операция: Анализ кода

Анализ кода — это процесс проверки обычного или управляемого кода .NET на предмет соответствия руководящим принципам по разработке. Для управляемого кода .NET в процессе анализа проверяется соответствие генерируемых сборок рекомендациям Microsoft .NET Framework Design Guide-lines. Предлагается автоматическая проверка сборок на наличие более чем 200 видов дефектов, таких как нарушение соглашений по именованию, ошибки в конструкции библиотек, а также проблемы локализации, безопасности и производительности. Задача анализа кода при работе с новыми базами кода — обеспечить отсутствие дефектов. Для существующих баз с большим числом правил формирования предупреждений цель состоит в минимизации числа предупреждений в каждой категории.

Определение приемлемых правил	Выберите приемлемые правила для данной категории приложений
Обеспечение отсутствия дефектов	Проверьте код на соответствие правилам
Уменьшение числа дефектов по категориям	Для существующих баз кода с большим числом проблем с соответствием правилам определите базовые показатели уровня проблем. При использовании автоматизации, определите число предупреждений в существующей базе кода

Реализация задачи по разработке базы данных

Задача по разработке базы данных — это деятельность разработчика, связанная с требованием к качеству или сценарием. В результате реализации задачи по разработке базы данных к системе добавляется новая функция. После завершения этой задачи необходимо провести тестирование модулей, обзор кода и его анализ, интеграцию кода и регистрацию его в существующей базе кода. Затем сценарий или требование к качеству передается на тестирование. Описанный здесь изолированный интерактивный процесс разработки баз данных обеспечивает гибкость той части приложения, которая связана с данными. Это достигается благодаря использованию отдельного проекта разработки базы данных, который обеспечивает автономные изменения. Важнее всего, что эти изменения могут быть проверены автономно с помощью инструментальных средств автоматизированного тестирования, что гарантирует отсутствие регресса. Таким образом, при каждом изменении в базе данных проводится ее тестирование. Все изменения немедленно попадают в вашу изолированную базу данных («песочницу») и сразу же здесь тестируются. Только после этого изменения можно развертывать на рабочем сервере — риск неуправляемых изменений при этом минимален. С другой стороны, такой подход обеспечивает высокую гибкость, благодаря тесному сотрудничеству с группой разработки основного приложения. Использование задач по разработке позволяет участникам проектной группы эффективно взаимодействовать и достигать общих целей.

Операции:

Оценка задачи по разработке базы данных	Проанализируйте поставленную задачу
Обновите локальную среду проекта	Извлеките из системы управления версиями требуемую версию проекта базы данных
Кодирование	Создайте новые или обновите существующие объекты схемы

Выполнение теста модуля	Определите тип теста модуля базы данных
Рефакторинг кода	Определите сложность
Обзор кода	Проверьте правильность имен
Интеграция изменений	Проверьте зависимости

Операция: Оценка задачи по разработке базы данных

Оценка стоимости задач по разработке базы данных помогает ограничить набор реализуемых функций, определить расписание и распределить приоритеты. Оценка всех задач по разработке базы данных выполняется на совещании по планированию итерации. Если общая стоимость задач по разработке превышает затраты на итерацию, задача должна быть отложена или переназначена. Менеджер проекта и бизнес-аналитик определяют приоритеты задач и откладывают выполнение наименее приоритетных из них. После утверждения задачи за ее оценку отвечает разработчик. Прежде чем приступать к оценке задачи по разработке базы данных, разработчик баз данных должен пересмотреть назначенную ему задачу, разобраться в требованиях и обсудить оценку с другими разработчиками баз данных. В результате все разработчики будут иметь единое представление о задаче.

Анализ поставленной задачи	Найдите связанный с поставленной задачей описатель
Детализация задачи	Рассмотрите задачу по разработке с учетом других подобных задач, а также требований к качеству и сценариев, для которых еще не назначены задачи по разработке. Создайте для них задачи
Оценка на основе опыта	Проводите оценку с учетом времени выполнения аналогичных задач
Балансировка загрузки	Менеджер проекта и бизнес-аналитик определяют приоритеты задач и откладывают выполнение наименее приоритетных из них. Если в результате оценки выясняется, что объем работ превышает возможный уровень для итерации, совместно с менеджером проекта попробуйте изменить приоритеты и перераспределить загрузку
Определите способы интеграции	Совместно с другими участниками группы разработки выработайте четкую картину интеграции данной функциональности с другими функциями

Операция: Обновление локальной среды проекта

Прежде чем начать разработку, нужно настроить среду разработки локальной базы данных. Сначала необходимо извлечь из системы управления исходным кодом нужную версию проекта. Этот проект надо собрать и развернуть на изолированном сервере (sandbox). Это и будет локальная изолированная среда, в которой разработчик будет проверять каждое изменение, вносимое в проект базы данных, прежде чем возвращать исходный текст в систему управления версиями и развертывать систему.

Извлеките из системы управления версиями требуемую версию проекта базы данных	Определите версию проекта базы данных, с которым вы хотите синхронизироваться
Определение изолированного сервера	Выберите сервер баз данных, который вы будете использовать в качестве «песочницы»
Компоновка проекта базы данных	Убедитесь, что параметры проекта баз данных настроены на изолированный сервер
Развертывание проекта баз данных на изолированном сервере	Разверните проект баз данных на изолированном сервере

Операция: Кодирование

Следующий шаг — собственно написание кода для задачи по разработке базы данных. Это означает обновление существующих объектов схемы базы данных или создание новых. После реализации очередного пакета изменений его нужно развернуть на изолированном сервере для тестирования.

Создание новых или обновление существующих объектов схемы	При необходимости создайте новые объекты схемы
Сборка проекта базы данных	Соберите проект базы данных
Развертывание проекта баз данных на изолированном сервере	Разверните проект баз данных на изолированном сервере

Операция: Выполнение теста модуля базы данных

С помощью теста модуля проверяется корректность реализации определенного программного компонента. Выполнение тестов модулей при разработке снижает число дефектов, обнаруживаемых на этапе тестирования, и позво-

ляет бороться с регрессом — появлением новых дефектов при исправлении уже выявленных дефектов или при добавлении новых функций. Тесты модулей не являются тестами функций или взаимодействия; их единственное назначение — проверка автономной работы фрагмента кода. Разработчики должны убедиться, что существующие тесты модулей проходят после написания нового кода. Тестирование должно проводиться на протяжении всего проекта — от реализации архитектурных основ в начале до внесения дополнений в конце проекта. При этом все разновидности тестов единообразно выполняются и выявляют дефекты. При тестировании модулей баз данных необходимо перед написанием тестов модулей создать или обновить план генерирования данных, чтобы при тестировании имелись необходимые тестовые данные.

Определение типа теста модуля	Определите типы разрабатываемых тестов модулей базы данных. *Тесты правильной работы* (positive unit tests) проверяют код в нормальном режиме и контролируют корректность результатов. *Тесты неправильной работы* (negative unit tests) умышленно некорректно используют код и проверяют его устойчивость и адекватность обработки ошибок
Создание или изменение плана генерации тестовых данных	Если плана генерации тестовых данных не существует, создайте его
Создание или изменение теста модуля базы данных	Напишите новые тесты модулей для проверки объектов схемы
Выполнение теста модуля базы данных	Убедитесь в правильной настройке параметров выполнения теста модуля базы данных
Анализ результатов теста	Если тест модуля не прошел по той причине, что был неверным, исправьте его и снова выполните
Отладка кода	Исправьте ошибки в программе, относящейся к задаче
Выполнение тестов модуля приложения	При итеративной разработке базы данных иногда целесообразно выполнять тесты модулей приложения совместно с тестами модулей базы данных. Поскольку приложение зачастую очень интенсивно обращается к данным, изменения в базе данных могут повлиять на прикладной слой, если их не принимать во внимание. Совместное выполнение тестов приложения и базы данных даст уверенность в том, что проблемы будут замечены заблаговременно

Операция: Рефакторинг базы данных

В процессе рефакторинга кода прогоняйте тесты модуля после каждого изменения. При таком подходе риск повреждения программы минимален. По возможности выполняйте автоматизированный рефакторинг, поскольку автоматизированные процессы минимизируют вероятность ошибок. Отметим, что рефакторинг надо проводить постоянно.

Определение сложности	При добавлении новых функций обратите внимание на те части программы, структура которых стала более сложной
Применение рефакторинга	При рефакторинге вносите за раз одно изменение. Измените код и все ссылки на измененную область
Сборка проекта базы данных	Соберите проект базы данных
Развертывание проекта баз данных на изолированном сервере	Разверните проект баз данных на изолированном сервере
Выполнение тестов модуля базы данных	Выполните тесты модуля, чтобы убедиться в семантической целостности кода после рефакторинга

Операция: Обзор кода

Обзор кода применяется для выявления областей, с которыми могут возникнуть проблемы в процессе дальнейшей разработки или тестирования. Обзор кода также дает возможность узнать мнения других разработчиков о рассматриваемом коде. Обзоры кода позволяют участникам, работающим в той же области, следовать прецедентам, созданным предыдущими разработчиками. В таких партнерских встречах требуется присутствие второго знающего коллеги, с которым разработчик должен рассмотреть все изменения, прежде чем зарегистрировать код в системе управления версиями.

Проверка правильности имен	Имена объектов схемы базы данных должны отражать назначение соответствующих сущностей
Проверка адекватности кода	Рассматриваемый код должен соответствовать задаче, для которой он написан. Допустимы только такие изменения кода, которые добавляют или изменяют функции системы
Проверка допустимой сложности кода	Повторяющийся код должен быть собран в общих функциях

Внесение изменений по результатам обзора	Внесите все изменения, запланированные в результате обзора
Сборка проекта базы данных	Соберите проект базы данных
Развертывание проекта баз данных на изолированном сервере	Разверните проект баз данных на изолированном сервере
Выполнение тестов модуля базы данных	Выполните тесты модуля и убедитесь в отсутствии регресса

Операция: Интеграция изменений

Чтобы приложение было устойчивым и согласованным, необходимо организовать интеграцию изменений в код. Когда код состоит из небольших изменений, каждое из которых соответствует задаче по разработке или исправлению дефекта, его проще поддерживать в стабильном состоянии. Легче найти и изолировать потенциальную проблему. Для состояния сценария или требования к качеству устанавливается значение Resolved (Обработан) и описатель назначается тестировщику.

Проверка зависимостей	Если задача зависит от других задач, которые еще не завершены, дождитесь, когда они будут интегрированы в систему
Сборка проекта базы данных	Соберите проект базы данных
Развертывание проекта баз данных на изолированном сервере	Разверните проект баз данных на изолированном сервере
Тестирование и интегрирование других задач по разработке баз данных	Проверьте, что вносимые вами изменения, связанные с исправлением дефекта, реализацией части сценария или требования к качеству, нормально работают совместно с уже интегрированными изменениями
Регистрация пакета изменений	Зарегистрируйте измененный код в системе управления версиями исходного кода. Если тест реализован или выполнен не полностью, сохраните весь код как промежуточную версию (shelveset). В противном случае установите признак решенной задачи. В результате описателю будет сопоставлен новый пакет изменений

Завершение задачи	Установите для описателя признак Resolved (Обработан) и назначьте сценарий одному из тех тестировщиков, которые писали тестовые задания для данного сценария. Назначьте этого тестировщика новым владельцем описателя задачи

Тестирование требования к качеству

Задача реализации требования к качеству передается на тестирование, когда в текущей сборке отражены ограничения, накладываемые данным требованием. Для проверки упомянутых ограничений нужно понимать, с чем они связаны. Зачастую с требованием связаны сценарии, которые демонстрируют эти ограничения. Тестирование требования к качеству подразумевает успешное завершение тестов на производительность, безопасность, стресс и нагрузку. На основании результатов тестирования создаются описатели дефектов, в которых документируются обнаруженные проблемы.

Операции:

Определение подхода к тестированию	Определите контекст проекта
Создание теста производительности	Определите назначение теста
Создание теста безопасности	Изучите точки входа
Создание стрессового теста	Спроектируйте тест
Создание нагрузочного теста	Определите назначение нагрузочного теста
Выбор и запуск тестового задания	Выберите, какой тест запускать
Документирование дефекта	Идентифицируйте дефект
Выполнение исследовательских тестов	Установите продолжительность сеанса

Операция: Определение подхода к тестированию

Подход к тестированию — это документ, определяющий стратегию создания и выполнения тестов. В нем также определены стандарты качества для поставляемого продукта. Подход к тестированию служит отправной точкой для планирования тестов в начале проекта, но продолжает существовать и меняться по ходу проекта. Подход должен описывать различные методики, например

тесты, как ручные, так и автоматические. Формулирование подхода к тестированию включает определение контекста теста и его конечной цели (или целей), а также выбор подходящих методик, с учетом целей и в контексте проекта. Перед началом очередной итерации данный документ нужно слегка корректировать, отражая цели тестирования в предстоящей итерации и добавляя описание используемых тестовых данных.

Определите контекст проекта	Определите риски проекта, а также пользователей, на которых может сказаться их воздействие. Найдите особые ситуации, которые могут повлиять на уровень необходимого тестирования. Например, действующие в предметной области нормы могут влиять на направления тестирования
Определение цели теста	Определите цели проекта, достижению которых будет способствовать тест
Оценка возможных методов тестирования	Рассмотрите различные инструментальные средства для тестирования
Определение показателей тестирования	Совместно с разработчиками определите реалистические пороговые значения, которые будут критериями прохождения теста. Это обязательный показатель

Операция: Создание теста производительности

С помощью теста производительности определяется время реакции приложения и проверяется соответствие требованиям к качеству. Тестирование производительности необходимо проводить в каждой итерации и резервировать время для устранения основных проблем и для проведения повторного тестирования.

Определение назначения теста	Необходимо четко сформулировать назначение теста. Например, с помощью теста проверяется отсутствие существенных различий во времени отклика системы для пользователей с различными способами подключения к Интернету
Описание конфигурации теста	Определите конфигурацию теста
Проектирование теста	Систематизируйте тестируемые события и соответствующие им сценарии. Разделите сценарии на области, критичные и не критичные к производительности. Совместно с архитектором постройте модель производительности, которая будет основой для кодирования

Документирование этапов тестирования	Выделите и опишите этапы тестирования, чтобы тестировщики могли единообразно выполнять и интерпретировать тесты. Ошибки на этом шаге могут привести к неверной оценке производительности. Везде, где это возможно, автоматизируйте создание тестовых заданий

Операция: Создание теста безопасности

Тесты безопасности или испытания на проникновение (penetration tests) имеют дело с угрозами, выявленными при проектировании. Тестирование этого типа делится на три части: изучение, определение нарушений и розыск. В результате испытания на проникновение могут быть обнаружены новые уязвимости, что повлечет за собой формулирование новых требований к качеству или идентификацию дефекта. В результате реализации такого требования или устранения дефекта должны быть заблокированы соответствующие точки входа и доступ к ресурсам. Из этого следует, что тестировщики должны быть осведомлены об элементах модели угроз не хуже архитекторов. Данный вид тестирования требует специальных навыков: нужно уметь имитировать действия потенциального злоумышленника. Доступ к модели угроз дает тестировщикам возможность систематически атаковать систему в целях проверки, не нарушая ее работу.

Изучение точек входа	Определите точки входа системы и ее функциональных блоков для защиты внутренних объектов. Используйте сведения из модели угроз для определения возможных направлений атак
Идентификация проблем	Напишите тестовые задания с запуском тестов прямого или псевдослучайного доступа к объектам. Для проверки специальных защитных функций требуется прямое вмешательство. Например, посмотрите, нельзя ли из URL получить идентификатор сеанса и изменить номер счета
Использование слабостей защиты	Добавьте тестовые задания с использованием любых слабостей защиты. Некоторые из этих тестов будут исследовательскими, а не нацеленными на исправление проблем. Учитывайте время, необходимое для описания способов обхода защиты. Несанкционированный доступ к системе является дефектом и для его исправления надо знать способ доступа к защищенному ресурсу. Сценарии запуска тестов безопасности должны отражать общую тактику защиты секретных данных, предохранения от несанкционированного доступа и запрета доступа для нелегитимных пользователей

Операция: Создание стрессового теста

Стресс-тесты позволяют определить предельные возможности приложения. С их помощью выясняют верхнюю границу, перейдя которую приложение не обеспечивает приемлемого уровня реактивности и полностью прекращает работу. Например, можно дать приложению высокую нагрузку и посмотреть, нет ли утечек памяти и нормально ли освобождаются объекты сборщиком мусора.

Стресс-тесты должны проводиться в каждой итерации. Они являются разновидностью тестов производительности и позволяют проверять экстремальные условия работы приложения, например, когда нагрузка превышает типичную или длится дольше обычного. Стресс-тесты позволяют спрогнозировать поведение приложения при превышении максимально возможной нагрузки. Также они служат для проверки устойчивости и надежности приложения.

Определение назначения теста	Назначение стресс-теста должно быть четко сформулировано, наряду с описанием конкретных параметров и условий, таких как предельные значения
Проектирование автоматизированного теста	Систематизируйте особенности тестовой среды, условия тестирования, включая предпосылки, число виртуальных пользователей, сценарий и ресурсы. Разберитесь с распределением действий и сценариев
Создание теста	Задокументируйте выполняемые пользователем действия. Добавьте проверку корректности получаемых результатов в стрессовых условиях. Чтобы тесты были динамичными, используйте привязку к данным — обращение к таким источникам данных, как SQL Server, Microsoft Excel или Microsoft Access

Операция: Создание нагрузочного теста

Нагрузочный тест является разновидностью теста производительности. Нагрузочные тесты позволяют проверить соответствие приложения требованиям к качеству в условиях нагрузки. С их помощью измеряется производительность приложения в типичных сценариях работы пользователей. Нагрузочные тесты позволяют спрогнозировать поведение приложения при достижении максимальной нагрузки. Нагрузочными тестами целесообразно проверять те 20% кода приложения, на которые приходится 80% его работы.

Определение назначения теста	Назначение нагрузочного теста должно быть четко сформулировано, наряду с описанием конкретных параметров и условий, таких как допустимый диапазон значений
Проектирование автоматизированного теста	Систематизируйте особенности тестовой среды, условия тестирования, включая предпосылки, число виртуальных пользователей, сценарий и ресурсы. Разберитесь с распределением действий и сценариев
Создание теста	Задокументируйте выполняемые пользователем действия. Добавьте проверку корректности получаемых результатов в стрессовых условиях. Чтобы тесты были динамичными, используйте привязку к данным — обращение к таким источникам данных, как SQL Server, Microsoft Excel или Microsoft Access

Операция: Выбор и запуск тестового задания

Самый важный момент в тестировании — это собственно выполнение тестов соответствующей сборки. Результаты тестов могут быть совершенно разными: от успешного завершения до полного выхода системы из строя. Нельзя забывать фиксировать результаты тестов в описателе соответствующего сценария или требования к качеству.

Определение запускаемого теста	Определите тесты, которые нужно выполнить, и присвойте им приоритеты на основе сведений в описателе задачи тестирования, общего плана тестирования и положений документа, определяющего подходы к тестированию
Определение конфигурации теста	Установите и настройте тестовую конфигурацию, включая аппаратные средства, программы и тестовые данные
Получение сборки	Извлеките сборку из системы контроля версий или получите ее из другого авторизованного источника
Выполнение теста	Выберите папку с тестом сценария или требования к качеству
Анализ результатов теста	Свяжите с результатами теста проверенный сценарий или требование к качеству
Закрытие описателя	Если все тесты из задачи тестирования выполнены, закройте описатель данной задачи

Операция: Документирование дефекта

Прежде чем создавать описатель нового дефекта, проверьте, не регистрировалась ли прежде подобная проблема. В противном случае, зафиксируйте все подробности в системе отслеживания дефектов. Необходимо описать последовательность воссоздания проблемы, после чего, в процессе классификации, ей будет назначен приоритет. Опишите дефект как можно подробней — это поможет специалистам определить оптимальный путь его устранения. Каждому открытому дефекту должен быть назначен владелец.

Проверка повторения дефекта	Прежде чем создавать описатель нового дефекта, проверьте, нет ли описания аналогичного дефекта в базе данных
Описание дефекта	Если аналогов у дефекта нет и он может быть воспроизведен, выполните последовательность действий, приводящую к его появлению и подробно обрисуйте происходящее в описателе
Назначение дефекта	Определите основную область проблемы. Если затрагивается несколько областей, выберите одну

Операция: Выполнение исследовательских тестов

Исследовательское тестирование — это способ проверки продукта без определенного набора тестов. Для исследовательского тестирования подходят многие эвристические методы. Среди них — использование собирательных образов, прилагательные и наречия, анализ переменных, поиск диапазона и тестирование различных состояний.

Представленный в данном руководстве эвристический метод описывает, как продукт тестируется с точки зрения собирательного образа с целью сбора новых требований. Для проведения подобного тестирования представьте собирательный образ и выполняйте функции системы, пытаясь достичь определенной цели. Если вы обнаружите новые потенциальные цели или выясните, что существующие функции не удовлетворяют требованиям собирательного образа, добавьте новые или измените существующие сценарии. Ограничивайте сеансы исследовательского тестирования двумя часами.

Установка продолжительности сеанса	Продолжительность сеанса исследовательского тестирования не должна быть меньше получаса и больше двух часов
Удовлетворение потребностей собирательного образа	Выберите собирательный образ из опубликованного списка персонажей

Обнаружение новых целей	Проанализируйте трудности, с которыми сталкивается собирательный образ. Если преодолеть их очень сложно, откройте новый дефект или добавьте в список сценариев новый сценарий или требование к качеству

Проверка сценария

Типичный сценарий включает как задачи тестирования, так и задачи по разработке. Эти задачи назначаются тестировщикам, которые разрабатывают и прогоняют тестовые задания. Задача реализации сценария передается на тестирование, когда реализованные функции интегрированы в сборку и она готова к тестированию. Чтобы проверить, что в сборке реализованы требуемые функции, следующие из сценария, нужно разбираться в нем и в граничных условиях. Для проверки всех функций сборки и граничных условий необходимо написать проверочные тесты. Все обнаруженные дефекты должны быть задокументированы с помощью описателей.

Операции:

Определение подхода к тестированию	Определите контекст проекта
Создание проверочного теста	Определите область тестирования и среду
Выбор и запуск тестового задания	Выберите, какой тест запускать
Документирование дефекта	Идентифицируйте дефект
Выполнение исследовательских тестов	Установите продолжительность сеанса

Операция: Создание проверочного теста

Проверочными тестами исследуют соответствие реализованных функций требованиям сценариев. При проверочном тестировании приложение рассматривается как «черный ящик» и особое внимание уделяется конечным пользователям. Проверочные тесты не должны быть завязаны на проектные решения. Для написания проверочных тестов исследуйте трудности, которые возникнут в различных ситуациях. Не все трудности указаны в сценариях. При проектировании и написании тестовых сценариев для проверочных тестов специалисты рассматривают ситуации, которые могут иметь место в реальной жизни.

Определение области тестирования и среды	Выделите область, в которой должен выполняться тест. Тесты итерации — это набор автоматических тестовых заданий, запускаемых после проверочных тестов сборки. При этом прохождение этих тестов не означает работоспособность сборки. Заметим, что только автоматические тесты могут быть тестами итерации
Выяснение подробностей выполнения теста	Определите тестовые данные, необходимые для тестового задания. Извлеките из системы управления версиями описание подхода к тестированию. В раздел, относящийся к соответствующей итерации, добавьте описание тестовых данных
Создание тестовых заданий	Создайте для сценария папку с тестами, если ее еще нет

Создание проекта базы данных

Операции по созданию проекта базы данных включают в себя разработку первоначального проекта базы данных для использования командой разработчиков. В рамках основного процесса создается автономное представление разрабатываемой базы данных, которое доступно всей команде посредством системы управления исходным кодом. Создание такого проекта позволяет вести независимую итеративную разработку базы данных одновременно несколькими разработчиками в стиле, который уменьшает риски при внесении изменений. Эта операция состоит из таких действий, как создание, конфигурирование и тестирование проекта базы данных.

Зачастую бывает нужно создать управляемую среду разработки для уже существующей базы данных. В этом случае необходимо дополнительное действие — импорт рабочей базы данных в новый проект. Выполнить это можно, только обладая определенными правами в рабочей базе данных, а потому данную задачу стоит поручить администратору базы данных, который обладает необходимыми правами доступа. Мастер проекта базы данных (Database Project Wizard) полезен при выполнении действий, описанных ниже, он упрощает создание проекта базы данных. После его завершения, однако, может потребоваться дополнительное конфигурирование.

Операции:

Создание проекта базы данных	Изучите поставленную задачу
Импорт существующей базы данных	Импортируйте схему существующей базы данных в проект базы данных

Установка параметров сборки и развертывания	Укажите сервер, где будет производиться разработка
Изменение сгенерированных сценариев	Откорректируйте сценарии, выполняемые перед развертыванием
Проверка проекта базы данных	Выполните сборку проекта базы данных
Размещение проекта базы данных в службе управления исходным кодом	Поместите проект базы данных в систему управления исходным кодом

Операция: Создание проекта базы данных

Первым шагом в этой операции является создание проекта базы данных на основе подходящего шаблона. Кроме того, должны быть заданы различные параметры проекта базы данных, поскольку от этого зависит дальнейшая жизнь проекта.

Изучение поставленной задачи	Просмотрите описатели задач, поставленных перед вами
Создание проекта новой базы данных	Решите, какой тип проекта вам нужен — SQL Server 2000 или SQL Server 2005
Конфигурирование свойств проекта	Решите, как будет организован проект базы данных — по типам объектов или схемой
Установка необязательных параметров	Сконфигурируйте необязательные параметры базы данных, включая SET-параметры

Операция: Импорт существующей базы данных

Зачастую бывает нужно создать проект базы данных на основе уже существующей базы путем импорта ее схемы. Это позволяет импортировать схему определения объектов и добавить их к автономному проекту базы данных, размещенному в файловой системе, тем самым предоставляя возможность менять их локально внутри проекта.

Импорт существующей базы данных в новый проект	Укажите соединение с базой данных, из которой будет производиться импорт схемы данных, а также укажите учетную запись с необходимыми полномочиями в ней

Операция: Установка параметров сборки и развертывания

В течение всего времени разработки базы данных, разработчики будут постоянно, на итеративной основе собирать и разворачивать проект базы дан-

ных на своих локальных серверах. Именно поэтому так важно во время первоначального конфигурирования проекта настроить параметры для сборки и установки, задаваемые по умолчанию, которые бы наилучшим образом подходили для среды разработки.

Выбор сервера, на котором будет производиться тестирование	Это должен быть локальный сервер баз данных, предназначенный для тестирования итеративных изменений
Настройка параметров установки	Решите, будет ли база данных при каждой установке автоматически создаваться с нуля, либо на ней будут разворачиваться только выполненные изменения

Операция: Изменение сгенерированных сценариев

Проект базы данных ограничен ее уровнем, а потому не поддерживает напрямую объекты уровня сервера, да и некоторые объекты схемы данных также не поддерживаются проектом напрямую. Однако все эти объекты легко могут быть добавлены в сценарии, запускаемые до и после развертывания базы данных. Поэтому следует при создании проекта внести необходимые изменения в сценарии, добавив туда нужные объекты и выражения SQL.

Изменение сценария, запускаемого перед установкой	В этот сценарий добавьте учетные записи сервера, тем самым задав соответствие между пользователями и учетными записями
Изменение сценария, запускаемого после установки	В этом сценарии выполните настройку прав доступа к объектам схемы, так чтобы они создавались сразу с необходимыми разрешениями: запрет, разрешение или отзыв прав

Операция: Проверка проекта базы данных

После выполнения всех необходимых настроек проекта следует собрать и развернуть его на локальном сервере, чтобы убедиться в его готовности. Это очень важный шаг, помогающий выявить в сценариях пропущенные или неверно заданные конфигурационные параметры.

Сборка проекта базы данных	Выполните сборку проекта базы данных
Установка проекта базы данных на локальном сервере для проверки	Установите проект базы данных на локальном сервере

Операция: Размещение проекта базы данных в службе управления исходным кодом

После проверки проект базы данных можно разместить в службе управления исходным кодом, чтобы он был доступен всей команде.

Регистрация проекта в службе управления исходным кодом	Убедитесь, что служба управления исходным кодом сконфигурирована правильно
Публикация проекта в службе управления исходным кодом	Поместите проект в службу управления исходным кодом и присвойте этому действию описатель
Описатель завершения создания проекта	Завершите задачу по созданию проекта базы данных и назначьте ее менеджеру проекта

Развертывание проекта базы данных

Каждый проект базы данных в конце концов должен быть развернут в рабочей среде. В развертывании проекта базы данных описывается, как администратор баз данных может превратить это в управляемый и предсказуемый процесс. Он заключается в получении правильной версии проекта базы данных и проверке его путем построения сборки, установки и выполнения набора тестов модулей. Администратор баз данных должен понять, какие изменения нужно внести в сгенерированные сценарии. И, наконец, сценарий создания базы данных должен быть запущен на рабочем сервере с последующей проверкой базы данных.

Операции:

Синхронизация проекта базы данных	Изучите назначенную вам задачу по развертыванию базы данных
Проверка сборки	Выполните сборку проекта
Выполнение тестирования модулей базы данных	Выполните тестирование модулей базы данных
Анализ изменений	В службе управления исходным кодом просмотрите историю изменений, выполненных после последней установки
Разработка сценария создания базы данных	Определите соединение к базе данных, где будет выполняться установка, а также необходимые свойства создания
Резервное копирование рабочей базы данных	Создайте резервную копию рабочей базы данных

Развертывание на тестовом сервере	Выполните сценарий создания базы данных на тестовом сервере, сконфигурированном как рабочий сервер
Установка базы данных	Выполните сценарий создания базы данных на рабочем сервере

Операция: Синхронизация проекта базы данных

Перед развертыванием администратору баз данных следует синхронизировать локальную базу данных с требуемой версией.

Изучение поставленной задачи в рамках работ по развертыванию	Просмотрите описатель задачи, поставленной перед вами в рамках работ по развертыванию
Синхронизация проекта базы данных с требуемой версией	Определите версию, с которой будет синхронизирован проект базы данных

Операция: Проверка сборки

Перед установкой проекта базы данных на рабочий сервер его важно сначала собрать и установить на локальном сервере, чтобы проверить качество сборки.

Сборка проекта базы данных	Выполните сборку проекта
Установка проекта базы данных на тестовый сервер	Разверните проект базы данных на локальном тестовом сервере

Операция: Выполнение тестирования модулей базы данных

Перед установкой необходимо выполнить набор тестов модулей проекта базы данных, чтобы убедиться в отсутствии внесенных в него дефектов.

Выполнение тестирования модулей базы данных	Убедитесь в правильном конфигурировании исполняемых тестов модулей
Анализ результатов тестирования	Если тесты завершились неудачно по причине содержащихся в них ошибок, создайте описание дефекта и передайте его владельцу теста модуля для изучения и исправления

Операция: Анализ изменений

Администратору баз данных важно точно понимать, какие изменения предполагается передавать на сервер баз данных. Определить их он может либо просмотром на сервисе управления исходным кодом истории изменений с момента предыдущей установки, либо сравнением схем данных между рабочей и развертываемой базой данных.

Просмотр истории изменений с момента предыдущей установки при помощи службы управления исходным кодом	Воспользуйтесь историей службы управления исходным кодом для просмотра всех изменений, выполненных после последней установки проекта базы данных
Сравнение схемы данных для анализа различий между рабочим сервером и проектом	Создайте сеанс сравнения схем баз данных для выявления различий между базой данных на рабочем сервере и проектом
Просмотр файла журнала изменений	Просмотрите файл журнала изменений проекта базы данных для понимания изменений, произведенных в ней с момента предыдущей установки

Операция: Разработка сценария создания базы данных

Сценарий создания базы данных должен быть сгенерирован и при необходимости изменен. В качестве места установки в проекте указывается рабочий сервер. Затем генерируется сценарий. Для учета специфических особенностей окружения в полученный сценарий могут быть внесены изменения. Эти изменения могут включать команды переноса данных.

Задание соединения к серверу, на котором будет выполняться создание, а также задание дополнительных параметров	Сконфигурируйте параметры проекта базы данных, задав соединение к серверу, где будет создаваться база данных, а также связанные с этим свойства
Генерация сценария создания базы данных	Сгенерируйте сценарий создания базы данных
Изменение сценария создания базы данных	Просмотрите полученный сценарий создания базы данных и убедитесь, что он выполняет ожидаемые действия, необходимые для синхронизации исходного проекта с рабочей базой данных

Операция: Создание резервной копии рабочей базы данных

Перед развертыванием базы данных необходимо сделать резервную копию существующей базы данных. Это чрезвычайно важно, так как дает возможность вернуть базу данных в исходное состояние в случае каких-либо ошибок.

Создание резервной копии рабочей базы данных	Непосредственно перед развертыванием создайте полную резервную копию рабочей базы данных, используя стандартные технологии резервирования, применяемые в вашей организации

Операция: Установка базы данных на тестовом сервере

Перед установкой на рабочий сервер можно выполнить необязательный шаг — протестировать развертывание на тестовом сервере, имеющем аналогичную конфигурацию. При этом выполняются все действия, которые будут выполняться при реальном развертывании — запуск сценариев и проверка их результатов.

Выполнение на тестовом сервере сценариев развертывания	Перед установкой на рабочий сервер запустите сценарии развертывания в редакторе T-SQL на тестовом сервере, чтобы убедиться в их успешности
Сравнение схем данных для проверки развертывания	Создайте сеанс сравнения схем данных для просмотра различий между базой данных проекта и базой данных на тестовом сервере

Операция: Установка базы данных

Теперь все готово для установки базы данных на рабочем сервере. Установка выполняется запуском на сервере сценария создания базы данных с последующей проверкой результатов.

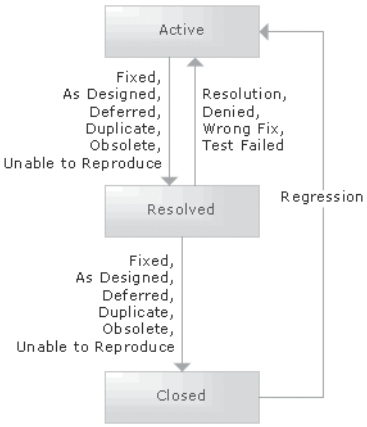
Запуск сценария создания базы данных на рабочем сервере	Используя редактор T-SQL, выполните сценарий создания базы данных на рабочем сервере
Сравнение схем данных для проверки развертывания	Создайте сеанс сравнения схем данных для просмотра различий между базой данных проекта и рабочей базой данных
Закрытие описателя установки	Отметьте задачу установки как выполненную

Описатели

Дефект

В описателе дефекта хранятся данные об имевшей место или существующей проблеме с системой. Назначение описателей дефектов — предоставление пользователям сведений, необходимых для полного понимания влияния проблемы на систему. Информация, содержащаяся в отчете об ошибке, должна облегчать ее воспроизведение. Проблемы должны четко определяться из результатов тестов. Четкость и доходчивость описания дефекта, как правило, повышают вероятность его устранения.

Состояния и переходы



Состояние: Новый: Чтобы разработчики могли устранить дефект, он должен быть зарегистрирован сразу после обнаружения. Прежде чем зарегистрировать дефект, нужно проверить существующие данные об ошибках и убедиться, что такой дефект еще не зарегистрирован.

- **Переход: От нового к активному:** Новый описатель дефекта создается в узле Work Items (Описатели) Проводника команды.
  - **Основание: Новый:** Дефект считается новым, когда создается впервые. Указывайте основание для создания всех описателей дефектов как «New» (Новый), если они не являются сбоями при сборке продукта.
  - **Основание: Сбой при сборке:** Основанием для создания описателя дефекта считается сбой при сборке, если выявленная проблема является прямым следствием безуспешной сборки продукта.

Состояние: Активный: Когда вы обнаруживаете дефект и вносите данные о нем с помощью Team Explorer, описатель дефекта автоматически устанавливается в активное состояние. Активное состояние указывает на то, что проблема существует и ее надо решать.

- **Переход: От активного к решенному:** Чтобы перевести проблему в состояние решенной, ее надо назначить или лицу, создавшему описатель дефекта, или тестировщику, чтобы они смогли проверить, устранен ли дефект.
  - **Основание: Исправлен:** Дефект считается исправленным («Fixed») после внесения изменений в соответствующий код и его регистрации в системе управления версиями. Свяжите дефект с набором изменений (changeset) после регистрации исправленного кода.
  - **Основание: Так задумано:** Основание решения проблемы «As Designed» (Так задумано) указывается, если кажущийся дефект соответствует ожидаемому состоянию или поведению системы.
  - **Основание: Отложен:** Основание «Deferred» (Отложен) устанавливается для дефекта, который не будет исправляться в данной итерации. Его исправление откладывается до следующих итераций или версий.
  - **Основание: Повтор:** Основание разрешения проблемы «Duplicate» (Повтор) устанавливается для дефектов, которые уже имели место. Добавьте ссылку на повторяющийся дефект, чтобы автору записи о дефекте было проще подтвердить повторение ошибки, прежде чем закрыть запись.
  - **Основание: Неактуален:** Основание «Obsolete» (Неактуален) устанавливается для дефектов, которые уже неприменимы к продукту. Например, если речь идет о проблеме, связанной с функцией, от которой отказались.
  - **Основание: Невозможно воспроизвести:** Основание «Unable to Reproduce» (Невозможно воспроизвести) устанавливается для ошибок, которые разработчик не может заново воссоздать на своем компьютере.

Состояние: Решен: Дефект находится в состоянии решенного, когда он отработан разработчиком или во время классификации. Основанием для перехода в это состояние может быть «Fixed» (Исправлен) или «As Designed» (Так задумано).

- **Переход: От решенного к закрытому:** Дефект считается закрытым, когда его решение проверено инициатором создания описателя дефекта или тестировщиком.
  - **Основание: Исправлен:** Дефект закрывается с основанием «Fixed» (Исправлен), когда автор описателя дефекта убеждается, что сборка продукта содержит исправления.
  - **Основание: Так задумано:** Дефект закрывается с основанием «As Designed» (Так задумано), когда автор описателя дефекта соглашается с тем, что дефект соответствует запланированному поведению системы.

- **Основание: Отложен:** Дефект закрывается с основанием «Deferred» (Отложен), если автор описателя дефекта согласен, что исправление дефекта нужно отложить.
- **Основание: Повтор:** Дефект закрывается с основанием «Duplicate» (Повтор), если автор описателя дефекта подтверждает, что этот описатель соответствует уже зафиксированной проблеме.
- **Основание: Неактуален:** Дефект закрывается с основанием «Obsolete» (Неактуален), если автор описателя дефекта согласен, что описанная проблема больше не может иметь отношения к продукту.
- **Основание: Невозможно воспроизвести:** Дефект закрывается с основанием «Unable to Reproduce» (Невозможно воспроизвести), если автор описателя дефекта не может воссоздать ситуацию возникновения зафиксированной проблемы или дать более подробные инструкции для ее повторной инициации.

- **Переход: От решенного к активному:** Если решение не может быть проверено, состояние описателя дефекта возвращается к активному («Active»). Например, если при работе исправленного кода снова повторяется та же проблема, тестировщик может вернуть дефект в состояние активного. При возврате дефекта в активное состояние не забудьте переназначить дефект соответствующему специалисту для классификации или исправления.
  - **Основание: Решение запрещено:** Дефект возвращается в состояние активного, если решение проблемы недопустимо. Чтобы упростить работу другим людям, которые будут заниматься этой проблемой, предоставьте конкретную информацию о причинах запрета.
  - **Основание: Неверное исправление:** Дефект возвращается в состояние активного, если исправление было некорректным. Подробно опишите, как и почему исправленная версия работала неверно.
  - **Основание: Тест не проходит:** Дефект возвращается в состояние активного, если тесты показывают, что ошибка не исправлена. Подробно опишите, какой тест и в какой сборке (build) не прошел.

Состояние: Закрыт: Дефект в состоянии «закрыт» не требует дальнейших действий в текущей версии продукта. Дефект закрывается после проверки его решения.

- **Переход: От закрытого к активному:** Закрытый дефект может быть переведен в состояние активного, если регрессионное тестирование показывает повторное возникновение проблемы.
  - **Основание: Рецидив:** Если при регрессионном тестировании дефект обнаруживается вновь, переведите его в активное состояние и классифицируйте. В поле «Reason» (Основание) установите значение «Regression» (Рецидив).

Поля

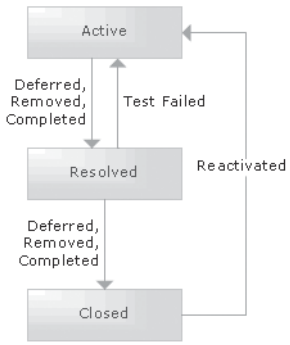
Название	Обязательное. В поле «Title» (Название) кратко описывается решаемая проблема. Название должно быть достаточно информативным, чтобы специалисты, осуществляющие классификацию, могли понять, на какую часть системы влияет проблема и каким образом
Область	Поле «Area» (Область) применяется для группировки дефектов по функциям или проектным группам в иерархии проекта. Область должна быть допустимым узлом в иерархии проекта
Итерация	В поле «Iteration» (Итерация) указывается итерация, в которой дефект исправлен
Кому назначен	В этом поле («Assigned To») указывается специалист, который в данный момент отвечает за обработку дефекта. Если для устранения дефекта требуется несколько различных исправлений, он может рассматриваться как сценарий и может быть назначен следующему в иерархической цепочке лицу. После интеграции всех частей исправления отчет о дефекте назначается тестировщику
Приоритет	Обязательное. Приоритет («Priority») — это субъективная оценка важности. Приоритет 1 указывает, что продукт не готов к поставке и должен быть исправлен как можно скорее. Приоритет 2 указывает на важный дефект, который нет необходимости исправлять немедленно, но необходимо устранить до поставки продукта. Приоритет 3 указывает на дефект, который можно исправлять или нет, в зависимости от ресурсов, времени и рисков
Состояние	Обязательное. В поле «State» (Состояние) указывает одно из возможных состояний дефекта: «Active» (Активный), «Resolved» (Решенный) или «Closed» (Закрытый)
Основание	Обязательное. В поле «Reason» (Основание) указывается, на каком основании дефект переведен в его текущее состояние. Например, «Fixed» (Исправлен) — одно из возможных оснований перевода дефекта в состояние решенного
Описание	Поле «Description» (Описание) служит для описания проблемы и шагов для ее воспроизведения
Архив	По мере обработки ошибки в поле «History» (Архив) накапливаются записи. При каждом изменении, связанном с дефектом, в это поле добавляется запись со сведениями о том, какие сделаны изменения, почему, а также другими подробностями

Препятствие	В поле «Issue» значения «Yes» (Да) и «No» (Нет) указывают на наличие или отсутствие проблем, каким-то образом препятствующих исправлению дефекта. Если в поле указано «Yes» (Да), в отчете о проблемах менеджера проекта будут содержаться сведения о соответствующем дефекте
Найдено в сборке	В этом поле («Found in Build») указывается номер сборки, в которой был обнаружен дефект
Решено в сборке	В этом поле («Resolved in Build») хранится номер сборки, в которой проблема была решена
Название теста	В этом поле («Test Name») указывается название теста, связанного с данным дефектом
Идентификатор теста	В этом поле («Test ID») указывается идентификатор теста, связанного с данным дефектом
Расположение теста	В этом поле («Test Path») указывается путь, по которому расположен тест, связанный с данным дефектом
Ссылки	Ссылки («Links») на связанные с данным описателем другие описатели, гиперссылки, наборы изменений или файлы с исходным кодом
Вложения	В этом поле («File Attachments») указываются файлы, содержащие дополнительные сведения о дефекте
Ранг	Относительный приоритет с учетом других описателей («Rank»)
Классификация	Содержит результаты совещания по классификации («Triage»). Если поле не заполнено, значит дефект не классифицирован

Требования к качеству

В требованиях к качеству фиксируются такие характеристики системы, как производительность, загрузка, доступность, специальные возможности, удобство обслуживания и сопровождения. Эти требования обычно имеют форму ограничений на функционирование системы.

Состояния и переходы



Состояние: Новое: Требования к качеству добавляются в список, который находится в папке требований библиотеки документов, или с помощью описателя в Team Explorer (Проводник команды).

- **Переход: От нового к активному**
  - **Основание: Новое:** Требование к качеству активируется как новое при первом создании.

Состояние: Активное: Работа с требованием к качеству начинается в активном состоянии. Бизнес-аналитик фиксирует требование, указывая для него информативное название, и заполняет поле описания как можно более подробными сведениями о требовании. После того как требование полностью сформулировано, бизнес-аналитик назначает его ведущему разработчику. В поле «Specified» (Сформулировано) устанавливается значение «Yes» (Да) и требование остается в активном состоянии на время его реализации. Ведущий разработчик координирует с другими разработчиками действия, необходимые для реализации требований.

- **Переход: От активного к обработанному**
  - **Основание: Завершено:** Требование к качеству переводится в состояние обработанного на основании «Completed» (Завершено), если команда разработчиков закончила написание кода, реализующего данное требование. Ведущий разработчик назначает требование тестировщику.
  - **Основание: Отложено:** Требование к качеству считается обработанным с основанием «Deferred» (Отложено), если в данной итерации реализовать его невозможно. Реализация требования может быть отложена из-за нехватки времени у разработчиков или обнаружения проблем, блокирующих работу. В поле «Iteration» (Итерация) нужно указать правильный номер итерации, в которой требование будет реализовано. Если осуществление требования откладывается до следующей версии продукта, поле «Iteration» нужно оставить пустым. Не забудьте подробно описать причины, по которым отложена реализация требования, и укажите планируемые сроки работы над ним.

- **Основание: Удалено:** Требование к качеству удаляется («Removed»), если больше не считается целесообразным. При удалении требования проверьте поля «Issue» (Препятствие) и «Exit Criteria» (Условия завершения). Обычно для удаленных требований в этих полях содержится «No» (Нет).

Состояние: Обработанное: После реализации требования к качеству ведущий разработчик устанавливает его состояние в «Resolved» (Обработанное). Ведущий разработчик также назначает это требование тестировщику, после чего может начаться проверка требования.

- **Переход: От обработанного к закрытому**
  - **Основание: Завершено:** Требование к качеству закрывается как «Completed» (Завершено), когда тестировщик сообщает о прохождении тестов. При завершении работы с требованием проверьте поля «Issue» (Препятствие) и «Exit Criteria» (Условия завершения). Обычно для реализованных требований в этих полях содержится «No» (Нет).
  - **Основание: Отложено:** Требование к качеству считается закрытым с основанием «Deferred» (Отложено), если в данной итерации реализовать его невозможно.
  - **Основание: Удалено:** Требование к качеству закрывается как удаленное («Removed»), если оно больше не считается целесообразным.
- **Переход: От обработанного к активному**
  - **Основание: Тест не проходит:** Требование к качеству возвращается в активное состояние, если не проходит один или несколько тестов. Тестировщик должен снова назначить данное требование ведущему разработчику, который его зафиксировал. Тестировщик также должен создать соответствующий описатель дефекта для сбоя теста.

Состояние: Закрытое: Тестировщик переводит работу с требованием в состояние «Closed» (Закрыто) при прохождении всех тестов. Требование также закрывается, если оно откладывается, удаляется или разбивается на более мелкие.

- **Переход: От закрытого к активному**
  - **Основание: Повторная активация:** Отложенное требование к качеству активируется заново, когда начинается итерация, на которую назначена его реализация. Если требование все еще нужно задокументировать, назначьте его бизнес-аналитику. Если же требование готово к реализации, назначьте его ведущему разработчику. При повторной активации удаленных требований, следуйте той же процедуре, что и для отложенных требований.

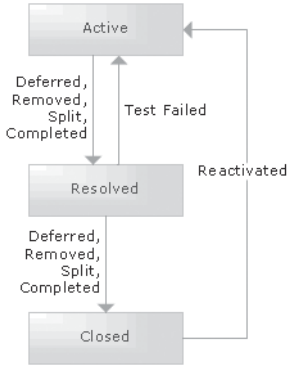
Название	Обязательное. Название должно быть максимально информативным
Область	Поле «Area» (Область) применяется для группировки требований к качеству по функциям или проектным группам. Область должна быть допустимым узлом в иерархии проекта
Итерация	В поле «Iteration» (Итерация) указывается итерация, в которой требование к качеству реализовано в коде
Тип	Имеется пять типов требований к качеству: «Load» (Нагрузка), «Stress» (Стресс), «Performance» (Производительность), «Platform» (Платформа), «Other» (Прочее) и «Security» (Безопасность)
Кому назначено	В данном поле («Assigned To») указывается специалист, который в данный момент отвечает за реализацию требования
Состояние	Обязательное. В поле «State» (Состояние) указывается одно из возможных состояний требования: «Active» (Активное), «Resolved» (Обработанное) или «Closed» (Закрытое)
Основание	В поле «Reason» (Основание) указывается, на каком основании требование к качеству переведено в его текущее состояние. Например, «Completed» (Завершено) — одно из возможных оснований перевода требования в состояние закрытого
Описание	Поле «Description» (Описание) служит для описания требования к качеству. Опишите требование как можно подробнее, чтобы разработчику было проще его реализовать, а тестировщику — проверить
Архив	По мере обработки ошибки в поле «History» (Архив) накапливаются записи. При каждом изменении требования в это поле добавляется запись со сведениями о том, какие сделаны изменения, почему, а также другие подробности
Препятствие	В поле «Issue» значения «Yes» (Да) или «No» (Нет) указывают на наличие или отсутствие проблем, каким-то образом препятствующих реализации требования. Если в поле указано «Yes» (Да) в отчете о проблемах, который получает менеджер проекта, будут сведения о данном требовании

Условия завершения	В поле «Exit Criteria» значения «Yes» (Да) или «No» (Нет) указывают, входит ли реализация данного требования в список обязательных работ (backlog) для текущей итерации. Это поле применяется для синхронизации различных представлений списка обязательных работ (список сценариев, набор требований к качеству и план итерации). Если условия завершения установлены в «Yes» (Да), данное требование к качеству отображается в контрольном списке проекта и одном из конечных продуктов. Данное поле в следующей версии будет переименовано в «Iteration Backlog» (Список обязательных работ в итерации)
Ранг	Значение в этом поле («Rank») указывает важность данного требования к качеству относительно всех прочих требований к программному продукту
Сборка с реализацией	В поле «Integration Build» хранится номер сборки (build), в которой содержится реализация данного требования
Идентификатор	В поле «ID» хранится уникальный идентификационный номер требования к качеству
Приблизительный порядок величины	Приблизительный порядок величины («Rough order of magnitude») — это способ оценки трудозатрат на реализацию сценария или требования к качеству. Если требуется выполнить не более шести заданий, требующих 1–2 дня, в этом поле указывается значение 1. Если требуется выполнить от 6 до 12 заданий, требующих 1–2 дня, в этом поле указывается значение 2. Если трудозатраты больше, в данном поле указывается значение 3 и рассматривается возможность разбиения на части задачи реализации сценария или требования к качеству

Сценарий

Сценарий (scenario) — это разновидность описателя, в котором записывается один из возможных вариантов взаимодействия пользователя с сис-темой. Сценарий содержит запись последовательности действий пользователя, которые он выполняет для достижения некоторой цели. В одних сценариях фиксируются успешные попытки, в других — безуспешные. При записи сценария нужно указывать, какой конкретный путь достижения цели описывается, поскольку таких путей множество.

Состояния и переходы



Состояние: Новый: Сценарии добавляются в список, который находится в папке сценариев библиотеки документов.

- **Переход: От нового к активному**
  - **Основание: Новый:** Сценарий считается новым, когда создается впервые.

Состояние: Активный: Начальное состояние сценария — активный. Бизнес-аналитик создает сценарий, указывая для него информативное название, и заполняет поле описания как можно более подробными сведениями о нем. После того как сценарий полностью описан, бизнес-аналитик назначает его ведущему разработчику. В поле «Specified» (Описан) устанавливается значение «Yes» (Да) и сценарий остается в активном состоянии на время его реализации. Ведущий разработчик координирует с другими разработчиками действия, необходимые для реализации данного сценария.

- **Переход: От активного к обработанному**
  - **Основание: Завершен:** Сценарий переводится в состояние обработанного на основании «Completed» (Завершен), если команда разработчиков закончила написание кода, реализующего данный сценарий. Ведущий разработчик назначает сценарий тестировщику.
  - **Основание: Разделен:** Сценарий переводится в состояние обработанного на основании «Split» (Разделен), если выяснилось, что он слишком объемный или что требуется описание дополнительных деталей. При разбиении сценария создайте новые сценарии и свяжите их с начальным сценарием.
  - **Основание: Отложен:** Основание «Deferred» (Отложен) устанавливается для сценария, который не будет реализован в данной итерации. Реализация сценария может быть отложена из-за нехватки времени у разработчиков или обнаружения проблем, блокирующих работу. В поле «Iteration» (Итерация) нужно указать правильный номер итерации, в которой сценарий будет реализован. Если реализация сценария откла-

дывается до следующей версии продукта, поле «Iteration» (Итерация) нужно оставить пустым. Не забудьте предоставить подробное описание причин, по которым отложена реализация сценария, и укажите планируемые сроки работы над ним.

- **Основание: Удален:** Сценарий удаляется («Removed»), если его реализация больше не считается целесообразной. При удалении сценария проверьте поля «Issue» (Препятствие) и «Exit Criteria» (Условия завершения). Обычно для удаленных сценариев в этих полях содержится «No» (Нет).

Состояние: Обработанный: После реализации сценария ведущий разработчик устанавливает его состояние в «Resolved» (Обработанный). Ведущий разработчик также назначает этот сценарий тестировщику, после чего можно начинать его проверку.

- **Переход: От обработанного к закрытому**
  - **Основание: Завершен:** Сценарий закрывается как «Completed» (Завершен), когда тестировщик сообщает о прохождении тестов. При завершении работы со сценарием проверьте поля «Issue» (Препятствие) и «Exit Criteria» (Условия завершения). Обычно для завершенных сценариев в этих полях содержится «No» (Нет).
  - **Основание: Разделен:** Сценарий закрывается на основании «Split» (Разделен), если выяснилось, что он слишком объемный или требуется описание дополнительных деталей.
  - **Основание: Отложен:** Основание «Deferred» (Отложен) устанавливается для сценария, который не будет реализован в данной итерации.
  - **Основание: Удален:** Сценарий удаляется («Removed»), если его реализация больше не считается целесообразной.
- **Переход: От обработанного к активному**
  - **Основание: Тест не проходит:** Если какие-либо тесты сценария не проходят, тестировщик должен вернуть сценарий в активное состояние и снова назначить его ведущему разработчику, создавшему данный сценарий. Тестировщик также должен создать соответствующий описатель дефекта для сбоя теста.

Состояние: Закрытый: Тестировщик закрывает сценарий при прохождении всех тестов. Сценарий также закрывается, если он откладывается, удаляется или разбивается на более мелкие.

- **Переход: От закрытого к активному**
  - **Основание: Повторная активация:** Сценарий может быть заново переведен в активное состояние при изменении функциональности.

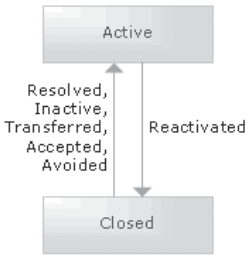
Название	Обязательное. Название должно отражать цель выполнения сценария. Название должно быть максимально информативным
Область	Поле «Area» (Область) применяется для группировки сценариев по функциям или проектным группам. Область должна быть допустимым узлом в иерархии проекта
Итерация	Итерация, в которой реализован сценарий
Кому назначено	Ответственный за работу со сценарием
Состояние	Обязательное. В поле «State» (Состояние) указывается одно из возможных состояний сценария: «Active» (Активный), «Resolved» (Обработанный) или «Closed» (Закрытый)
Основание	В поле «Reason» (Основание) указывается, на каком основании сценарий переведен в его текущее состояние. Например, «Completed» (Завершен) — одно из возможных оснований перевода сценария в состояние закрытого
Описание	Описание сценария на достаточно общем уровне. Подробное описание сценария должно быть в одном из конечных продуктов
Архив	По мере обработки сценария в поле «History» (Архив) накапливаются записи. При каждом изменении, связанном со сценарием, в это поле добавляется запись со сведениями о том, какие сделаны изменения, почему, а также другие подробности
Препятствие	В поле «Issue» значения «Yes» (Да) или «No» (Нет) указывают на наличие или отсутствие проблем, каким-то образом препятствующих реализации сценария. Если в поле указано «Yes» (Да), в отчете о проблемах менеджера проекта будет присутствовать соответствующий сценарий
Условия завершения	В поле «Exit Criteria» значения «Yes» (Да) или «No» (Нет) указывают, входит ли данный сценарий в список обязательных работ (backlog) для текущей итерации. Это поле применяется для синхронизации различных представлений списка обязательных работ (список сценариев, набор требований к качеству и план итерации). Если условия завершения установлены в «Yes» (Да), данный сценарий отображается в контрольном списке проекта и одном из конечных продуктов. Данное поле в следующей версии будет переименовано в «Iteration Backlog» (Список обязательных работ в итерации)
Ранг	Значение в этом поле («Rank») указывает важность данного сценария относительно всех прочих сценариев

Сборка с реализацией	В поле «Integration Build» хранится номер сборки («Build»), в которой содержится реализация данного сценария
Идентификатор	В поле «ID» хранится уникальный идентификационный номер сценария
Приблизительный порядок величины	Если требуется выполнить от 6 до 12 заданий, требующих 1–2 дня, в этом поле указывается значение 2. Приблизительный порядок величины («Rough order of magnitude») — это способ оценки трудозатрат на реализацию сценария или требования к качеству. Если трудозатраты больше, в данном поле указывается значение 3 и рассматривается возможность разбиения на части задачи реализации сценария или требования к качеству. Если требуется выполнить не более шести заданий, требующих 1–2 дня, в этом поле указывается значение 1.

Риск

Важной составляющей управления проектом является идентификация рисков и контроль над ними. Риск — это вероятное событие или фактор, который может оказать негативное влияние на проект в будущем. Описатели рисков позволяют документировать и отслеживать технические и организационные риски проекта. Рискам можно сопоставить задачи, которые нужно выполнить для их уменьшения. Например, создание некоторого архитектурного прототипа может снизить технический риск. Участники проекта должны относиться к выявлению рисков как к положительному явлению, поскольку это дает максимум данных о возможных проблемах. Обстановка в коллективе должна быть такой, чтобы лица, идентифицирующие риски, не боялись высказывать свою — новаторскую или спорную — точку зрения. Проектные команды с позитивным отношением к рискам имеют больше шансов своевременно обнаружить и учесть риски, чем команды с негативным к ним отношением.

Состояния и переходы



Состояние: Новый: Описатель нового риска создается в любой момент, когда определяется риск. Этот новый описатель должен содержать название

и описание, которые четко определяют потенциальные последствия риска. Также должно быть указано возможное воздействие риска. Новый описатель риска может быть создан любым участником команды и должен быть назначен специалисту, который будет отслеживать этот риск до его закрытия или переназначения.

- **Переход: От нового к активному:** Описатель нового риска создается в любой момент, когда определяется риск. Для его создания используется меню в Team Explorer.
  - **Основание: Новый:** Описатель нового риска создается в любой момент, когда потенциальный риск может повлиять на проект. Для его создания используется меню в Team Explorer.

Состояние: Активный: Когда с помощью Team Explorer создается новый описатель риска, его состояние автоматически устанавливается в «Active» (Активный). Активное состояние риска указывает, что может произойти некоторое событие, способное повлиять на проект. Каждый риск должен быть назначен некоторому владельцу.

- **Переход: От активного к закрытому**
  - **Основание: Снижен:** Риск может быть закрыт с указанием основания «Mitigated» (Снижен), если были предприняты некоторые действия для предотвращения возникновения рискованного события и/или уменьшения воздействия риска до приемлемого уровня.
  - **Основание: Неактивный:** Риск может быть закрыт на основании «Inactive» (Неактивный), когда вероятность его возникновения можно игнорировать. Никаких действий предпринимать не требуется.
  - **Основание: Переведен:** Риск может быть закрыт на основании «Transferred», если его можно вынести за рамки проекта. При этом сам риск не исчезает. Он лишь не влияет на проект в его текущем состоянии. Примеры перевода рисков: перемещение риска на следующую версию, привлечение сторонних консультантов или покупка готового программного компонента вместо его разработки.
  - **Основание: Принят:** Некоторые события невозможно предотвратить или снизить их воздействие. В таких случаях члены команды принимают риск, осознавая его эффект. При этом указывается основание закрытия «Accepted» (Принят).
  - **Основание: Аннулирован:** Риска может больше не быть из-за изменений в проекте или изменений в самом риске. Отслеживать его в проекте больше нет оснований и он закрывается как «Avoided» (Аннулирован).

Состояние: Закрытый: Закрываются риски, более не представляющие угрозы для проекта. При ретроспективном анализе итерации, в которой был закрыт риск, риск может подробно обсуждаться.

- **Переход: От закрытого к активному**
  - **Основание: Повторная активация:** Риск может возникнуть снова и при этом повторно активируется. Его состояние при этом меняется с «Closed» (Закрытый) на «Active» (Активный).

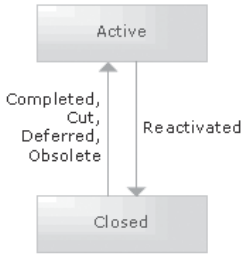
Название	Обязательное. В поле «Title» (Название) кратко описывается потенциальный риск. Название должно быть достаточно информативным, чтобы участникам команды было понятно, с чем связан риск
Область	Поле «Area» (Область) применяется для группировки рисков по функциям или проектным группам. Область должна быть допустимым узлом в иерархии проекта
Кому назначено	Ответственный за отслеживание риска. Обычно это менеджер проекта или архитектор, но ответственным может быть и любой другой член проектной группы
Серьезность	Серьезность («Severity») содержит оценку влияния неблагоприятного эффекта, уровень потерь или возможных затрат на устранение риска («Low» — низкая, «Medium» — средняя, «High» — высокая или «Critical» — критическая)
Основание	В поле «Reason» (Основание) указывается, на каком основании риск переведен в его текущее состояние. Например, «Avoided» (Аннулирован) — одно из возможных оснований перевода риска в состояние закрытого
Состояние	Обязательное. В поле «State» (Состояние) указывается одно из возможных состояний риска: «Active» (Активный) или «Closed» (Закрытый)
Итерация	Итерация, в которой может произойти рискованное событие
Приоритет	Приоритет описывает важность риска по отношению к другим рискам; удобен для определения последовательности, в которой следует снижать риски
Ссылки	Ссылки («Links») на связанные с данным описателем другие описатели, гиперссылки, наборы изменений или файлы с исходным кодом
Вложения	В этом поле («File Attachments») указываются файлы, содержащие дополнительные сведения о риске
Препятствие	В поле «Issue» значения «Yes» (Да) или «No» (Нет) указывают на наличие или отсутствие проблем, каким-то образом препятствующих обработке риска. Если в поле указано «Yes» (Да), в отчете о проблемах менеджера проекта будет присутствовать соответствующий сценарий

Условия завершения	В поле «Exit Criteria» значения «Yes» (Да) или «No» (Нет) указывают, входит ли обработка данного риска в список обязательных работ (backlog) для текущей итерации. Это поле применяется для синхронизации различных представлений списка обязательных работ (список сценариев, набор требований к качеству и план итерации). Если условия завершения установлены в «Yes» (Да), данный риск отображается в контрольном списке проекта и одном из конечных продуктов. Данное поле в следующей версии будет переименовано в «Iteration Backlog» (Список обязательных работ в итерации)
Описание	В данном поле содержится описание риска, его воздействие на проект и возможные варианты его уменьшения
Архив	В данном поле фиксируются все изменения, происходящие с описателем

Задача

Описатель задачи служит для представления некоторой работы. Представители каждой роли используют описатели задач по-своему. Например, разработчикам компонентов с их помощью назначаются работы, необходимые для реализации сценариев и требований к качеству. Тестировщики используют описатели задач для распределения работ по написанию и прогону тестовых заданий. Описатели задач также могут применяться для предупреждения о возврате в предыдущее состояние или для предложения выполнить некоторые исследования. Вообще, описатель задачи является универсальным средством распределения работ в рамках проекта. Отметим, что некоторые поля описателей задач применяются только для определенных ролей.

Состояния и переходы



Состояние: Новый: Новый описатель задачи может быть создан в любой момент, когда обнаруживается потребность в выполнении той или иной работы. Существуют три типа задач. Задачи по разработке связаны с выработками архитектурных решений, а также с реализацией сценариев или требований к качеству. Тестовые задачи соответствуют необходимым тестам. Третий

тип относится к работам, выходящим за рамки этих двух областей. Для создания описателя задачи используется Team Explorer (Проводник команды).

- **Переход: От новой к активной**
  - **Основание: Новая:** Задача считается новой, если создается впервые.

Состояние: Активная: Когда с помощью Team Explorer создается новый описатель задачи, его состояние автоматически устанавливается в «Active» (Активный). Задача активна, когда выполняется некоторая связанная с ней работа. Каждой задаче по разработке или тестированию должен быть назначен владелец и дисциплина.

- **Переход: От активного к закрытому**
  - **Основание: Завершена:** Задача закрывается как завершенная («Completed»), если проделаны все составляющие ее работы.
  - **Основание: Отложена:** Основание «Deferred» (Отложена) устанавливается для задачи, которая не будет выполнена в данной итерации. Выполнение задачи может быть отложено из-за нехватки времени у разработчиков или обнаружения проблем, блокирующих работу. В поле «Iteration» (Итерация) нужно указать правильный номер итерации, в которой задача будет выполнена.
  - **Основание: Неактуальна:** Задача закрывается с основанием «Obsolete» (Неактуальна), если ее описатель представляет работу, которая больше не нужна для реализации продукта.
  - **Основание: Вырезана:** Задача закрывается с основанием «Cut» (Вырезана), если реализуемые ею функции удалены из продукта.

Состояние: Закрытая: Задача в состоянии «закрыта» не требует дальнейших действий в текущей версии продукта. Задача по разработке закрывается после интеграции созданного кода в систему. Тестовая задача закрывается, когда проходят все тесты для области, к которой относится данная задача.

- **Переход: От закрытого к активному**
  - **Основание: Повторная активация:** Задача по разработке или тестированию может быть заново переведена в активное состояние при изменении функциональности.

Название	Обязательное. В поле «Title» (Название) кратко описывается решаемая задача. Название должно быть достаточно информативным, чтобы специалисты, осуществляющие классификацию, могли понять, на какую часть системы влияет решаемая задача и каким образом
Дисциплина	Указывает тип задачи: разработка, тестирование или прочее. Задачи по разработке и тестированию предполагают определенное состояние работ при закрытии задачи

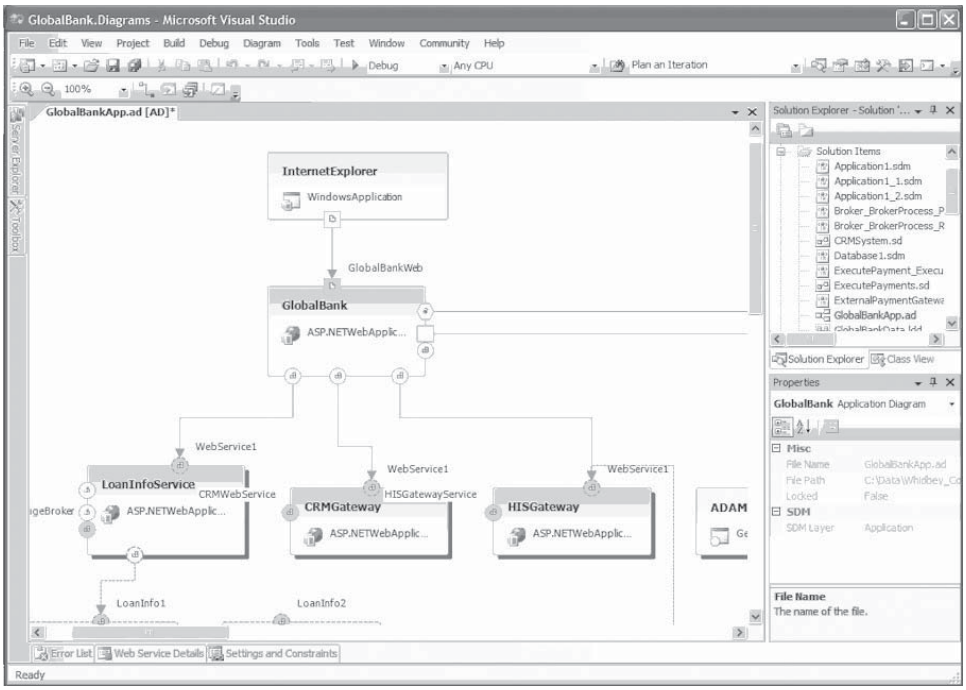
Область	Поле «Area» (Область) применяется для группировки задач по функциям или проектным группам. Область должна быть допустимым узлом в иерархии проекта
Итерация	Запланированная итерация, в которой произошло исправление дефекта
Кому назначена	Ответственный за выполнение задачи
Состояние	Обязательное. В поле «State» (Состояние) указывается одно из возможных состояний задачи: «Active» (Активная) или «Closed» (Закрытая)
Основание	Обязательное. В поле «Reason» (Основание) указывается, на каком основании задача переведена в ее текущее состояние. Задача может быть закрыта, потому что она завершена, отложена, вырезана или неактуальна
Ранг	Обязательное. Ранг (поле «Rank») — это субъективная оценка важности. Значение «1» указывает на высокую важность задачи — ее необходимо выполнить как можно раньше. Значение «2» присваивается достаточно важным задачам, которые должны быть выполнены после задач с рангом 1. Ранг 3 имеют задачи, выполняемые после задач с рангом 1 и 2
Краткое описание	Краткое описание задачи
Подробное описание и архив	Подробное описание задачи и набор всех предыдущих записей о задаче
Вложения	Ссылки на файлы или другие описатели. Для задач по разработке указываются ссылки на соответствующие сценарии или требования к качеству. Также указываются файлы с поясняющими текстами и другие вспомогательные данные
Препятствие	В поле «Issue» значения «Yes» (Да) или «No» (Нет) указывают на наличие или отсутствие проблем, каким-то образом препятствующих решению задачи. Если в поле указано «Yes» (Да), задача будет отображена в отчете о проблемах менеджера проекта
Условия завершения	В поле «Exit Criteria» значения «Yes» (Да) или «No» (Нет) указывают, является ли данная задача одной из обязательных работ («Backlog») для текущей итерации. Это поле применяется для синхронизации различных представлений списка обязательных работ (список сценариев, набор требований к качеству и план итерации). Если условия завершения установлены в «Yes» (Да), данная задача отображается в контрольном списке проекта и одном из конечных продуктов. Данное поле в следующей версии будет переименовано в «Iteration Backlog» (Список обязательных работ в итерации)

Сборка с реализацией	Номер сборки, содержащей реализованные функции
Оставшаяся работа (в часах)	Объем работ, оставшихся до завершения задачи. Данное поле синхронизируется с Microsoft Project и может использоваться при применении последнего
Проделанная работа (в часах)	Объем проделанной для решения задачи работы. Данное поле синхронизируется с Microsoft Project и может использоваться при применении последнего

Результаты работ

Диаграмма приложения

На диаграмме приложения отображается программный комплекс в целом и содержатся такие компоненты, как веб-службы, веб- и Windows-приложения, а также связанные ресурсы, среди которых внешние базы данных, веб-службы и службы Biz-Talk. На этой диаграмме показаны взаимосвязи между этими приложениями и их текущая конфигурация.

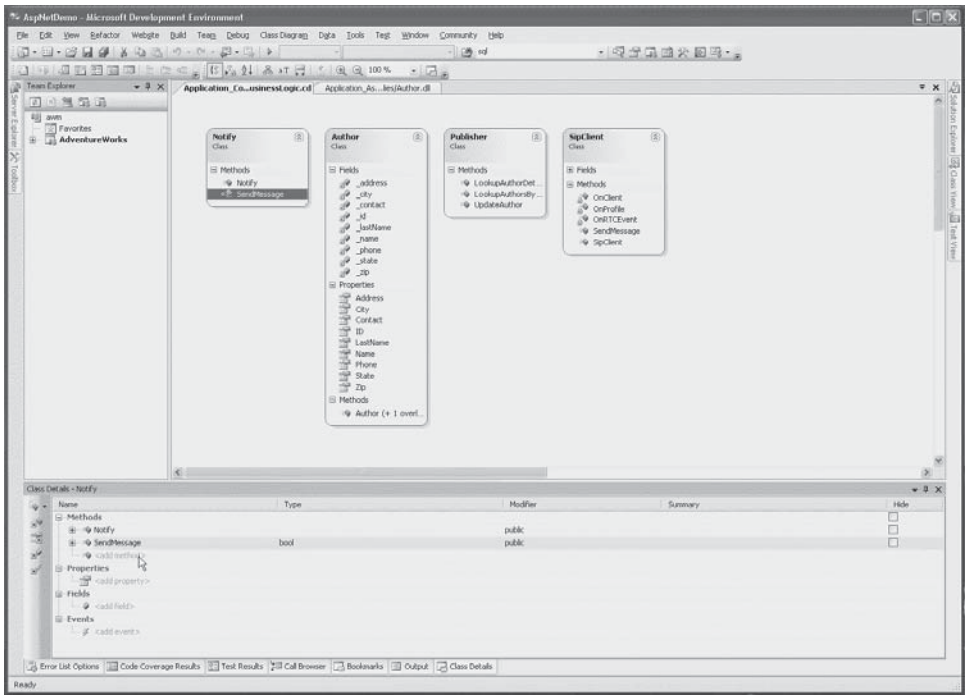


Пакет изменений

Пакет изменений — это логическое объединение сделанных изменений. Изменения бывают задержанными, отложенными и зафиксированными. Задержанные изменения — это группа изменений, выполненных в рамках одной акции, но еще не переданных в базу данных на публикацию и постоянное хранение. Отложенные изменения — это изменения, которые еще не переданы в базу данных, но все их содержимое, включая измененные файлы, находится на сервере. Зафиксированные изменения — это набор изменений, которые сохранены в архиве базы данных и доступны всем для просмотра.

Диаграмма классов

На диаграмме классов отображается логическое или физическое объединение классов и их взаимосвязи.

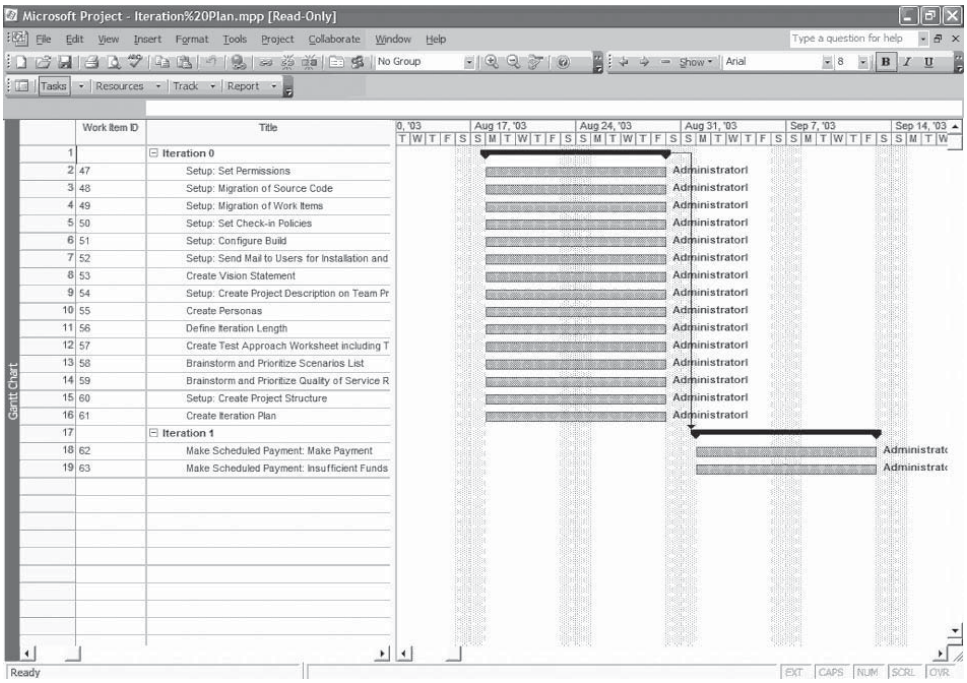


Исходный код

В исходный код входят все виды исходного кода: написанные на языках программирования, сценариев, разметки и запросов.

План итерации

План итерации — это набор сценариев, требований к качеству и задач, относящийся к одной итерации. Окончательно план утверждается на собрании по планированию итерации, непосредственно перед ее началом. Он может быть выполнен как в формате Microsoft Excel, так и Microsoft Project.

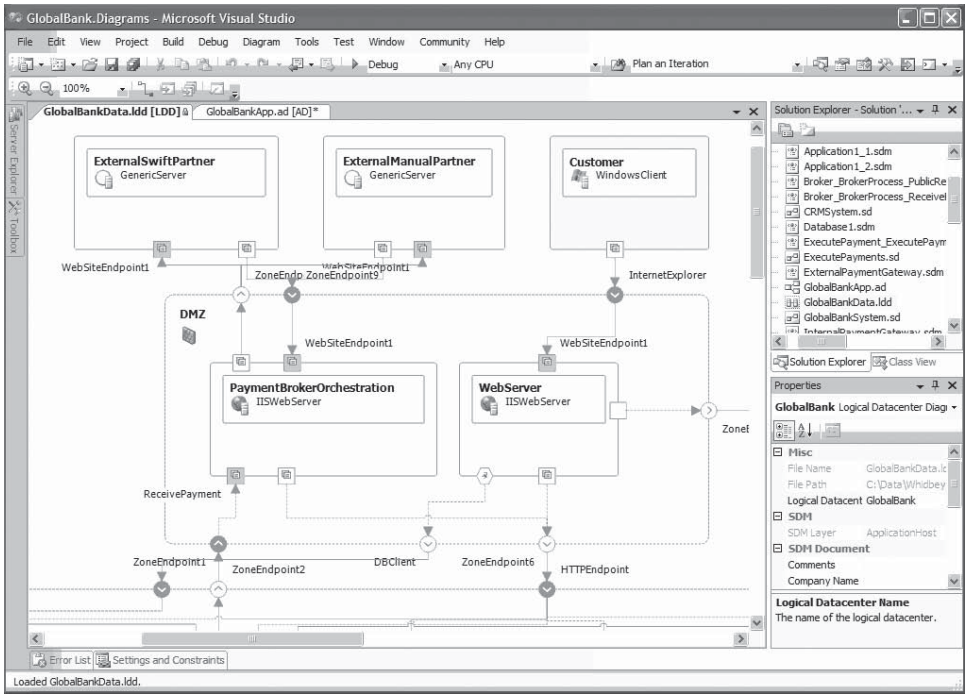


Нагрузочный тест

Сценарий нагрузочного теста включает набор тестов, описания нагрузки и различных инструментов, помогающих проверить приложение в режиме нагрузочного и стрессового тестирования.

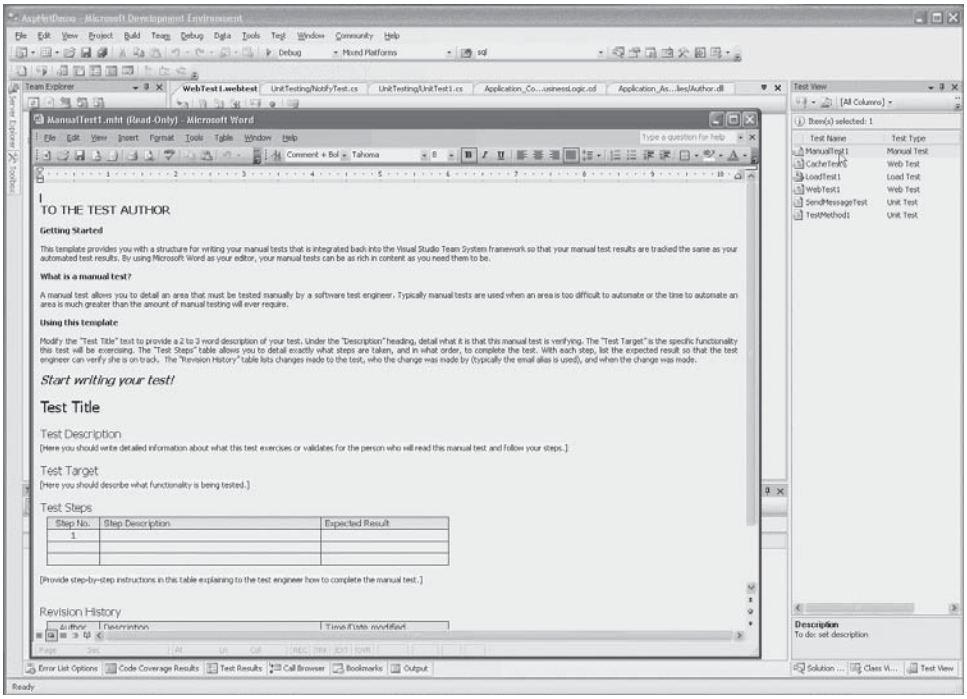
Логическая диаграмма центра обработки данных

В логической диаграмме центра обработки данных определены (задокументированы) конкретные конфигурации серверов приложений, таких как Internet Information Server, SQL Server или BizTalk Server, каждый из которых играет свою роль, например защищенного публичного веб-сервера. На этой диаграмме показываются взаимосвязи между серверами. Логические серверы могут объединяться в зоны, определяющие логические границы связей. Зоны могут иметь ограничения на типы логических серверов, которые в них могут быть включены, а также на направление и виды связей, ведущих за пределы зоны.



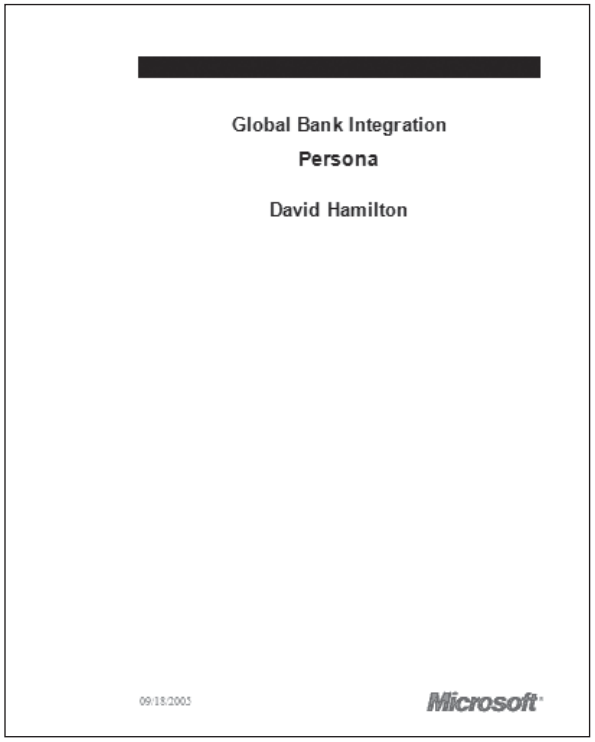
Ручной тест

Тесты, выполняемые в ручном режиме, описываются в документах формата Word или в простых текстовых файлах, содержащих перечень действий, которые нужно выполнить в процессе тестирования.



Собирательный образ

Собирательный образ (persona) описывает типичные умения, возможности, потребности, желания, рабочие привычки, задачи и данные об образовании определенной группы пользователей. Собирательный образ — это вымышленный персонаж, объединяющий в себе самые важные характеристики реальной группы пользователей. Размышлять о целой группе пользователей проще, представляя себе одного человека, так как одного легче понять, чем группу. Каждый раз, имея дело с собирательным образом, мы как бы работаем с отдельным человеком, а на самом деле обращаемся к целой группе, которую он олицетворяет.


Global Bank Integration

Persona

David Hamilton

09/18/2005

Microsoft

Контрольный список проекта

Контрольный список проекта — это перечень задач, который должен быть составлен до начала проекта или итерации.

Список требований к качеству

Данный список представляет собой перечень требований к качеству, которые необходимо реализовать. После проведения обзора и назначения приоритетов, список требований к качеству и список сценариев определяют минимально приемлемый уровень проекта.

План выпуска продукта

В план выпуска продукта включаются все мероприятия, связанные с выпуском приложения. Он содержит материалы для всех участников, вовлеченных в выпуск. План выпуска продукта может быть составлен в Microsoft Office Project.

Описание сценария

В сценарии описывается определенная часть взаимодействия пользователя с системой. В сценарии содержится запись последовательности действий пользователя, которые он выполняет для достижения некоторой цели. В одних сценариях фиксируются успешные попытки, в других — безуспешные. При написании сценариев следует быть конкретным. Так как возможно бесконечное количество возможных сценариев, важно записать только те из них, которые имеют характерные отличия.

Список сценариев

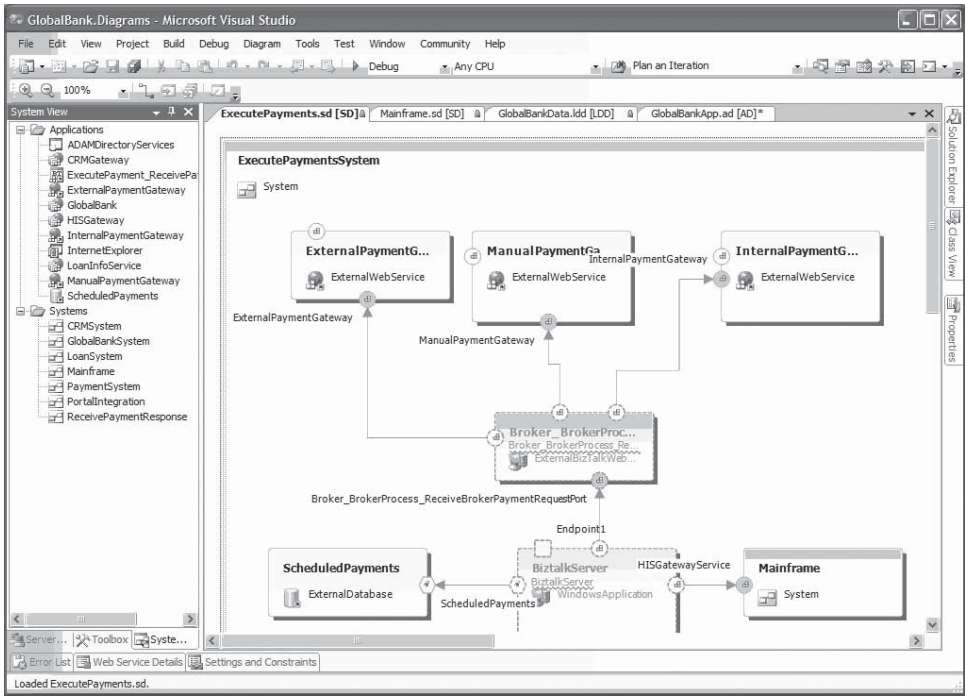
В списке сценариев содержится перечень обязательных работ по реализации сценариев проекта. После проведения обзора и назначения приоритетов, список требований к качеству и список сценариев определяют минимально приемлемый уровень проекта.

Раскадровка

Описание деталей пользовательского интерфейса в сценариях может быть как текстовым, так и визуальным. Раскадровка используется для иллюстрации некоторых деталей в сценариях до написания кода. Она может быть создана в Microsoft Office PowerPoint или в любом другом графическом редакторе.

Диаграмма системы

Диаграмма системы — это представление структуры приложений и/или других систем в виде совокупности компонентов. Связи между приложениями и системами показывают, как они конфигурируются при развертывании системы. Для иллюстрации внешних интерфейсов компонентов могут использоваться конечные точки. Системы могут быть вложены друг в друга на любую глубину. Системы могут разрабатываться как снизу вверх путем объединения существующих приложений или систем, так и сверху вниз путем описания характеристик системы с применением так называемых приложений-прототипов (shadow application).



Групповая сборка

Групповая сборка — это компиляция всех файлов, библиотек или компонентов в новый набор исполняемых файлов, библиотек или компонентов. Групповые сборки, прошедшие контрольное тестирование, называются самотестирующимися (self-test), а не прошедшие — самоотрицающими (self-toast).

Подход к тестированию

Описание цели тестирования, тестового покрытия, методов тестирования и тестовых данных содержится в соответствующей электронной таблице. Для каждой итерации имеется свой раздел с описанием целей тестирования этой итерации и применяемых методов.

Результат тестирования

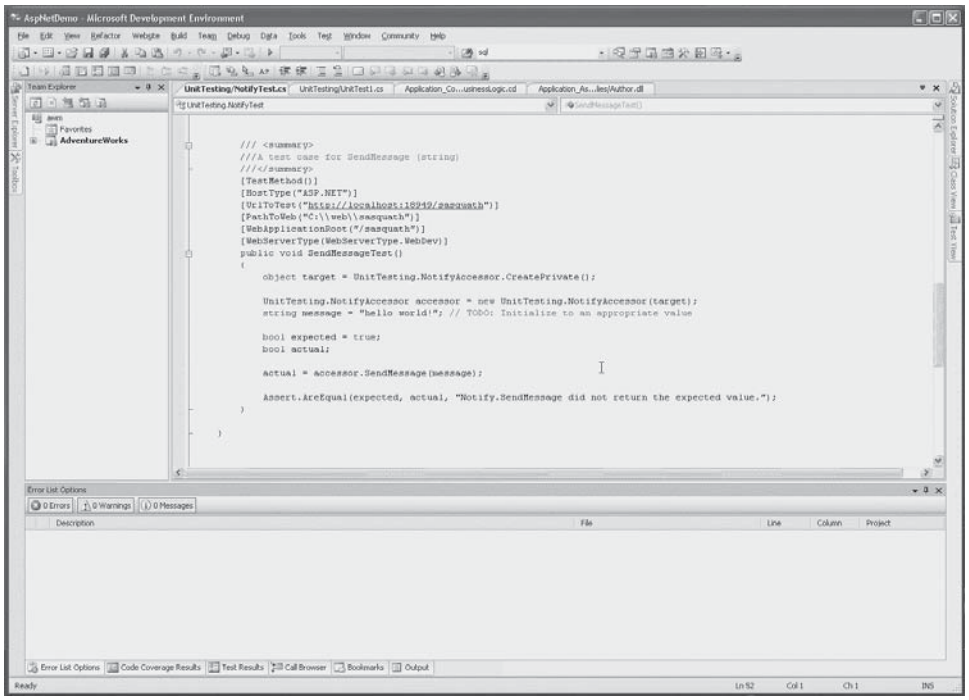
Результат тестирования — это заключение о результатах выполнения тестов. Возможные заключения: тест пройден, не пройден или не завершен.

Модель угроз

Модель угроз позволяет отслеживать использование точек входа, через которые можно получить доступ к внутренним ресурсам. Если угроза возможна, то она становится уязвимостью в защите.

Тест модуля

Тесты модулей обычно пишутся разработчиком для проверки поведения отдельных методов или их наборов. Кроме того, они могут использоваться для тестирования методом «прозрачного ящика» или в сочетании с нагрузочным и стрессовым тестированием.

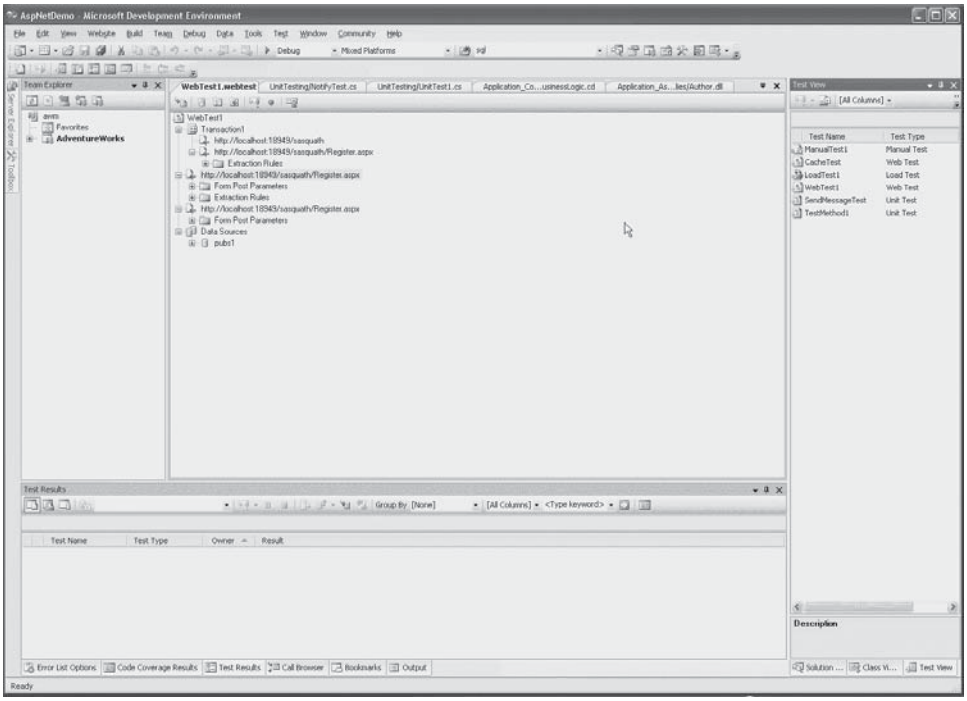


Концепция

Концепция проекта — это описание законченного продукта, его достоинств и пользы, которую заинтересованные стороны смогут получить от его использования.

Веб-тест

С помощью веб-тестов проверяют веб-страницы и HTTP-запросы.



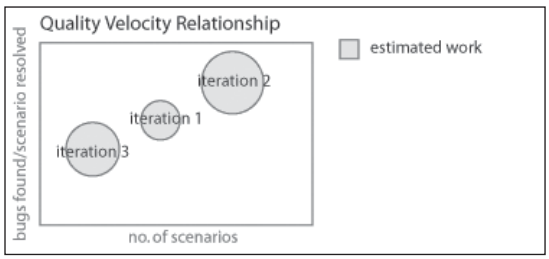
Прототип

Прототип — это небольшой фрагмент кода, зачастую исследовательский по своей природе, предназначенный для решения определенной проблемы или для снижения риска. Прототип может быть выполнен на любом языке программирования.

Отчеты

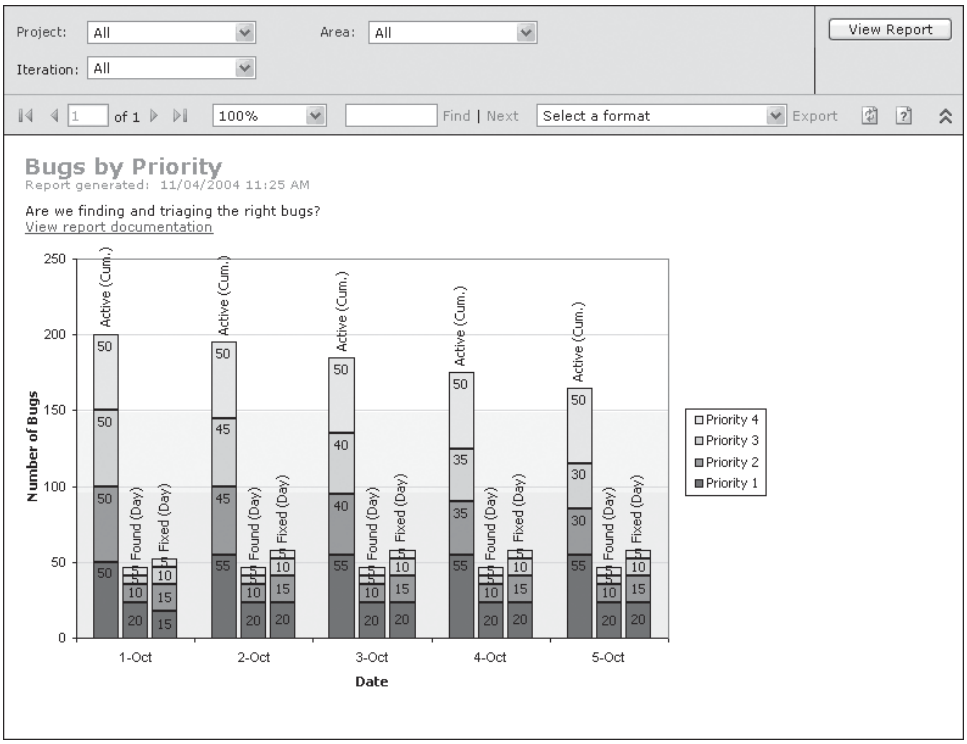
Качество или скорость

Как много сценариев может быть выполнено без потери качества? Пока команды в работе используют поговорку «поспешишь — людей насмешишь», в проекте есть резервы для ускорения. Задача менеджера проекта состоит в том, чтобы найти такой баланс, при котором скорость разработки будет максимальной при должном качестве. На следующей диаграмме показано отношение предполагаемого объема итерации к качеству.



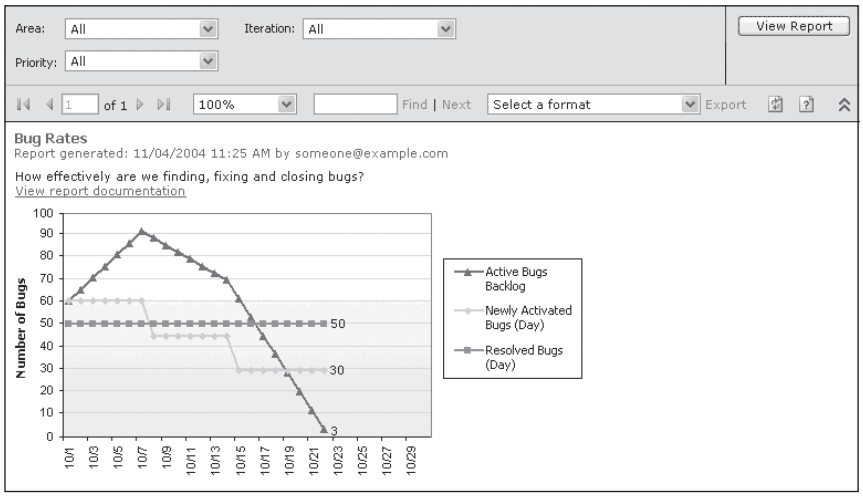
Приоритетные дефекты

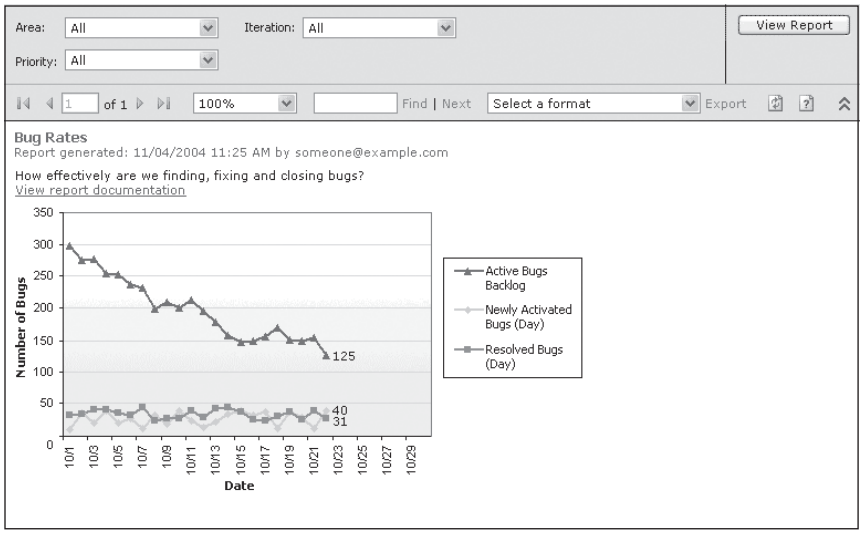
Правильно ли был выполнен поиск дефектов и назначение им приоритетов? Отчет, на котором показаны приоритетные дефекты, помогает понять, насколько эффективным был их поиск и определение важности. Выявление дефектов является неотъемлемой частью процесса разработки продукта. Однако зачастую пользователю досаждают именно те дефекты, которые непросто обнаружить. Если дефекты, имеющие высокую важность, не были выявлены, а имеется чрезмерно большое количество мелких недостатков, то следует направить усилия тестирования на поиск серьезных дефектов. Процесс определения приоритетов дефектов таит в себе опасность переоценить их настолько, что их количество превысит возможности их устранения, либо, наоборот, недооценить их, из-за чего пользователи будут крайне недовольны продуктом.



Интенсивность дефектов

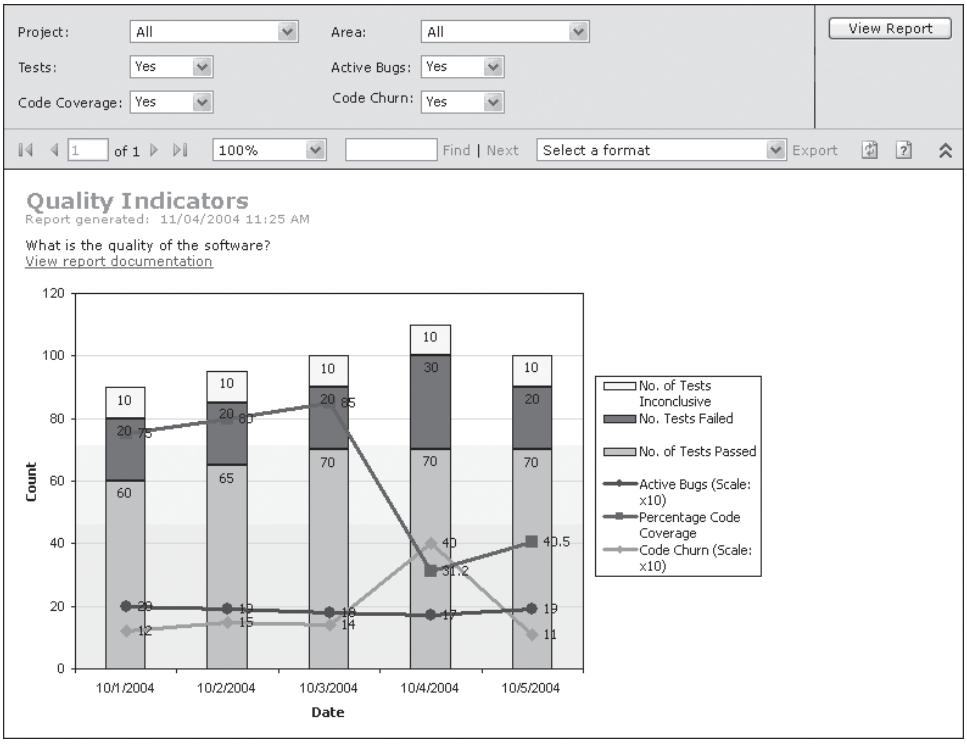
Насколько эффективно обнаруживаются, исправляются и закрываются дефекты? Оценивать их интенсивность лучше всего в сопоставлении со всей текущей деятельностью команды и другим метрикам на диаграмме индикаторов качества (Quality Indicators). Например, высокий коэффициент обнаружения дефектов может свидетельствовать как о плохо написанном, не полностью интегрированном коде, так и об эффективном тестировании. С другой стороны, низкий коэффициент обнаружения может означать высокое качество продукта, либо неэффективное тестирование. Для правильной оценки полезны показатели покрытия кода тестами, изменчивости кода и интенсивности дефектов.





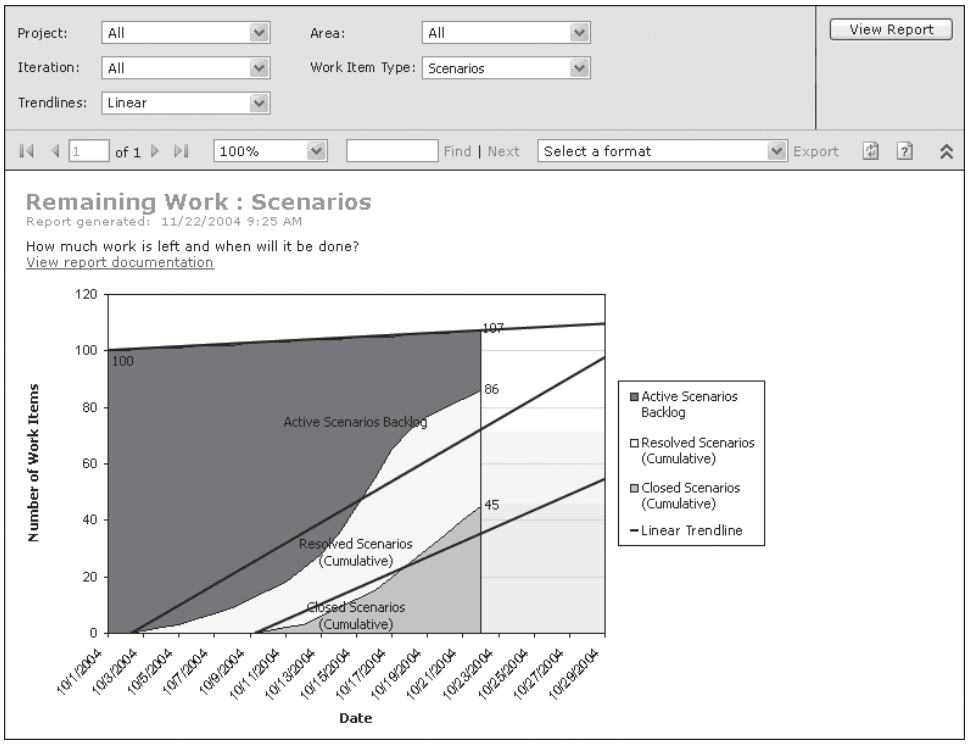
Индикаторы качества

Каково качество продукта? В идеальном случае на диаграммах коэффициентов прохождения тестов, количества дефектов и изменчивости кода можно увидеть похожие картины, но так бывает нечасто. В случае обнаружения несоответствий следует более детально изучить соответствующие серии сборок и данных. На следующей диаграмме объединены результаты тестов, покрытие ими кода, изменчивость кода и количество дефектов, что позволяет увидеть картину с разных сторон.



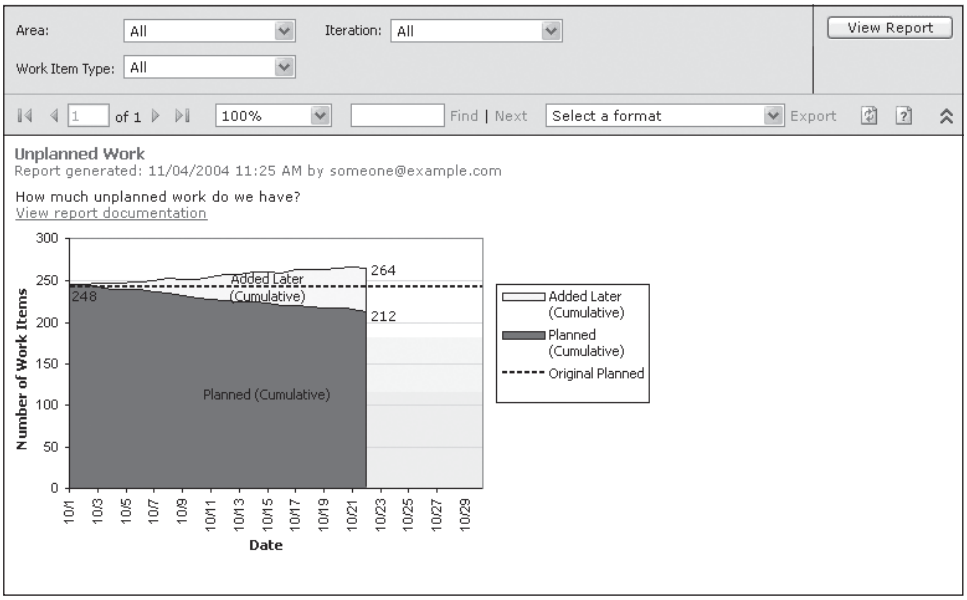
Оставшаяся работа

Сколько еще работы предстоит выполнить и когда она должна быть закончена? На сводной диаграмме процессов показано количество оставшейся работы, измеряемой в количестве сценариев и требований к качеству, обработанных и закрытых в процессе итерации.



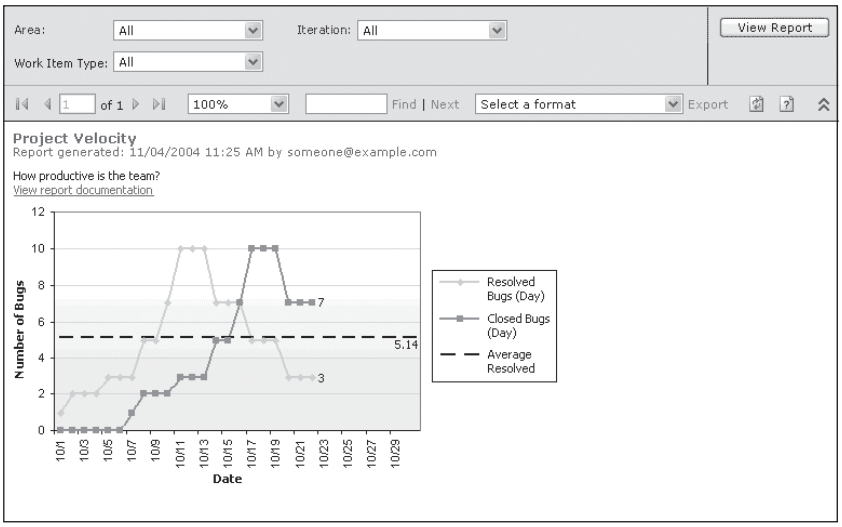
Внеплановая работа

Каков объем внеплановых работ? На следующей диаграмме отображен не оставшийся объем работ, а общий, разделенный, в свою очередь, на запланированную и внеплановую работу. Крайне редко случается выполнение всей работы по проекту раньше срока, даже в рамках отдельной итерации. Очень полезно резервировать значительный запас времени для непредвиденных работ (таких как исправление дефектов). К тому же, если вдруг не окажется достаточного резерва времени, вы можете быть вынуждены делать внеплановую работу за счет плановой.



Темп

Насколько быстро работает команда? Темп является одним из ключевых параметров такой оценки. Он показывает, насколько быстро выполняются плановые работы, ежедневные изменения плана, а также изменения от итерации к итерации. Эти данные, дополненные оценкой качества, полезны при планировании следующей итерации. Данная диаграмма, так же как и диаграмма оставшейся работы, важна для анализа посуточного темпа в рамках итерации или среднего темпа за итерацию в рамках проекта.



Возобновленные работы

Как много задач приходится возобновлять? Возобновленные работы — те, которые преждевременно были отмечены как завершенные или закрытые. Небольшой уровень таких работ вполне приемлем (скажем, меньше 5%). Однако высокий уровень либо его повышение должны насторожить менеджера проекта. В этом случае следует выявить причину и устранить ее.