# Этапы разработки проекта: определение стратегии тестирования и проектирование

## Определение стратегии тестирования

Как отмечалось ранее, на этапе анализа привлекаются группы тестирования, например для получения сравнительных характеристик предполагаемых к использованию аппаратных платформ, операционных систем, СУБД, иного окружения. Кроме того, на данном этапе определяется план работ по обеспечению надежности информационной системы и ее тестирования. Для любых проектов целесообразным является привлечение тестеров на ранних этапах разработки, в частности на этапе анализа и проектирования. Если проектное решение оказалось неудачным и это обнаружено слишком поздно - на этапе разработки или, что еще хуже, на этапе внедрения в эксплуатацию, - то исправление ошибки проектирования может обойтись очень дорого. Чем раньше группы тестирования выявляют ошибки в информационной системе, тем ниже стоимость сопровождения системы. Время на тестирование системы и на исправление обнаруженных ошибок следует предусматривать не только на этапе разработки, но и на этапе проектирования.

Для автоматизации тестирования следует использовать системы отслеживания ошибок (bug tracking). Это позволяет иметь единое хранилище ошибок, отслеживать их повторное появление, контролировать скорость и эффективность исправления ошибок, видеть наиболее нестабильные компоненты системы, а также поддерживать связь между группой разработчиков и группой тестирования (уведомления об изменениях по e-mail и т.п.). Чем больше проект, тем сильнее потребность в bug tracking.

## Проектирование

На этапе проектирования формируется модель данных. Проектировщики в качестве исходной информации получают результаты анализа. Конечным продуктом этапа проектирования являются:

* схема базы данных (на основании ER-модели, разработанной на этапе анализа);
* набор спецификаций модулей системы (они строятся на базе моделей функций).

Если проект небольшой, то в качестве аналитиков, проектировщиков и разработчиков могут выступать одни и те же люди. Возникает вопрос: насколько вообще актуальна передача результатов самому себе? Думаем, что актуальна. Представьте себе, что вы передаете данные кому-либо, кто мало знает о системе. Зачастую это помогает, например, найти не описанные вообще, нечетко описанные, противоречиво описанные компоненты системы.

Все спецификации должны быть точными. План тестирования системы дорабатывается также на этом этапе разработки. Во многих проектах результаты этапа проектирования оформляются единым документом, который называют технической спецификацией. В нем также описывают принятый подход к решению каких-либо сложных технических вопросов.

### Журнал проектирования

При проектировании возникает необходимость регистрировать все обсуждаемые варианты и окончательные решения. Не секрет, что проектировщики порой меняют первоначальные решения. Это может происходить потому, что со временем участники проекта забывают аргументы в пользу принятого решения. Подобную информацию можно хранить в репозитарии используемого CASE-средства, в текстовых файлах, просто на бумаге. Журнал проектирования является полезным материалом для новых членов групп проектировщиков, а также для разработчиков и тестировщиков.

Такой журнал может вестись как на этапе анализа, так и на этапе разработки и тестирования.

### Планирование этапа проектирования

Тщательное планирование важно для любого проекта. Это входит в обязанности руководителя проекта и руководителя группы проектирования (консультации с аналитиками в этом случае будут обязательными). Это позволяет:

* Разбить глобальную задачу на небольшие, независимые задачи. Такими задачами легче управлять, такие задачи легче реализовывать.
* Определить контрольные даты (этапы сдачи), которые позволят определить, насколько успешно продвигается проект, какие направления отстают, какие недогружены, какие работают успешно. Это позволяет обнаружить отставание от сроков сдачи и вовремя предотвратить авралы.
* Определить зависимости между задачами, а также последовательность завершения задач.
* Прогнозировать загрузку персонала, наем временных работников, привлечение других групп разработчиков, привлечение консультантов (если это необходимо).
* Получить четкое представление о том, когда можно начать этап реализации.
* Получить четкое представление о том, когда можно начать этап опытной эксплуатации.

### Перепланирование

Заказчики всегда хотят, чтобы план выполнения работ оставался неизменным. На практике этого редко удается достичь в полном объеме. Определенным компромиссом здесь может стать неизменность установленных сроков сдачи компонентов системы в эксплуатацию.

Задачи проектирования.

## Ранние стадии

### Рассмотрение результатов анализа

Это собственно процесс передачи информации от аналитиков проектировщикам. На практике это итерактивный процесс. У проектировщиков неизбежно будут возникать вопросы к аналитикам, и наоборот. Информация о системе будет постоянно уточняться. При разработке схемы базы данных может измениться информационная модель, полученная на этапе анализа, например потому, что имеющееся проектное решение нестабильно либо медленно работает при реализации его посредством выбранной СУБД или в силу иных причин. Проверить, охватывает ли анализ все бизнес-процессы системы (то есть осуществить проверку на полноту), проектировщики не в состоянии, но проверку информационной модели на непротиворечивость и корректность проектировщики провести могут. Это позволяет отследить ошибки в информационной модели и не повторить их в модели данных. Если результаты хранятся в репозитарии CASE-средства, то такая проверка на корректность может быть произведена автоматически.

### Семинары

Ранние стадии проектирования сопряжены с нудной и утомительной работой. Проектировщики и аналитики должны достигнуть полного понимания требований заказчика. Семинары являются быстрым и эффективным способом обмена информацией. Хороший способ убедиться в том, правильно ли проектировщики понимают назначение той или иной подсистемы, - взять один или несколько сценариев бизнес-процессов и проиграть их. Это можно сделать в форме простой диаграммы потока данных, причем необходимо указать не только автоматизированные, но и ручные функции.

### Критические участки

Критические участки системы изучаются при первом обследовании системы и уточняются на этапе анализа. Термин «критические» может означать жизненно важные как для нормального функционирования информационной системы с точки зрения бизнеса (например, время простоя автоматизированной линии изготовления материала X не должно превышать одной минуты), так и для успешной реализации и приемки проекта. Критические с точки зрения бизнеса участки информационной системы хорошо подходят для макетирования. На основе этих макетов (работы макетирования выполняют проектировщики и группы тестирования) тестеры дают оценку качества как информационной модели, так и модели данных. Также макетирование позволяет показать, какие требования и какими средствами могут быть выполнены, а какие требования - не могут.

Часто на этапе проектирования выявляются критические участки, которые не были очевидными на этапе анализа. Это влечет за собой необходимость уточнения информационной модели. Часто это связано с особенностями реализации тех или иных возможностей в выбранной СУБД. Некоторые функции, которые на этапе анализа выглядят простыми, могут стать очень сложными, когда дело дойдет до физической реализации. Например:

* В выбранной СУБД отсутствует эффективный механизм сканирования деревьев, а при анализе выявлено большое количество справочников и выбраны интерфейсы представления в виде деревьев, кроме того это понравилось заказчику, а СУБД при большом справочнике работает слишком медленно.
* Другая распространенная неприятность - неполно реализованная ссылочная целостность. В СУБД не реализованы каскадные модификации, в информационной модели нормализованные отношения предполагают наличие каскадных удалений и обновлений. Реализация же таких механизмов посредством триггеров оказалась слишком медленной, и уровень каскадирования триггеров ниже, чем уровень каскадных операций, определенных в информационной модели.

Такие моменты могут инициировать не только изменение информационной модели, но и смену СУБД.

Между критическими участками проекта и его рисками существует тесная связь. Критический участок разработки (например, срыв сроков первого этапа сдачи проекта заказчику) может стоить целого проекта. Причинами срыва могут быть как ошибки проектирования, так и нехватка персонала.

### Оценка ограничений

Ограничениями, известными с момента обследования бизнес-процессов, являются смета затрат и сроки внедрения. Могут быть и другие ограничения, например допуск персонала к той или иной информации (группы аналитиков к информации о бизнес-процессах в фирме, ограничения доступа к секретной информации и т.п.).

Решения относительно выбора аппаратной платформы, как правило, необратимы, поскольку тесно связаны со сметой затрат и наличием обслуживающего персонала. Например, решения на платформе RS/6000 и Intel с точки зрения сметы затрат выглядят одинаково, но персонала, способного квалифицированно обслуживать RS/6000, нет, и руководство не согласно оплатить обучение сотрудников, хотя решение на основе RS/6000 обладает более высокой масштабируемостью. Это может послужить причиной выбора платформы Intel. Аналогичные причины могут влиять и на выбор операционной системы.

Если проект является расширением или модернизацией существующей информационной системы, то число унаследованных ограничений также может быть большим. На этапе проектирования осуществляется обязательная проверка требований к информационной системе в свете выявленных ограничений. Менять платформу, операционную систему или СУБД на этапе реализации сложно, а на этапе опытной эксплуатации практически невозможно (на это просто не хватит времени, если не произойдет чудо). Чем большее количество компонентов системы уже реализовано, тем сложнее произвести подобную замену. Большинство СУБД сейчас работают на нескольких аппаратных платформах и нескольких операционных системах, но если есть участки кода проекта, зависимые от операционной системы или аппаратной платформы, то их изменение может обойтись очень дорого. В данной статье мы не будем обсуждать вопросы переносимости кода, поскольку они выходят за рамки этого цикла.

Если какие-либо требования не могут быть удовлетворены в принципе, принимается решение о доведении этого факта до сведения спонсоров проекта (руководства фирмы). Обнаружение неработоспособности системы в процессе эксплуатации ничем хорошим обернуться не может, особенно если до руководства фирмы дойдет информация о том, что невыполнимость требований была известна заранее.

### Определение целевой архитектуры

Под выбором архитектуры мы понимаем и выбор платформы (платформ), и выбор операционной системы (операционных систем). В  системе могут работать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем. Если к автоматизации того или иного бизнеса до вас уже приложили руки, причем неоднократно, вы можете обнаружить настоящий «зверинец» платформ и операционных систем. Перенос ПО на ту или иную платформу - процесс не безболезненный, да и управление разнородной сетью может также стать делом проблемным. Если же обстоятельства таковы, что ПО на рабочих местах конечных пользователей должно работать под управлением нескольких операционных систем (ОС), то следует обязательно выделить зависимые от ОС участки кода и жестко описать интерфейсы обмена компонентов информационной системы, сделав их независимыми от ОС. При написании кода модулей, работающих под управлением нескольких ОС, следует ориентироваться на ту из них, которая обладает наиболее жесткими требованиями.

Кроме определения платформы следует выяснить следующее:

* Будет ли это архитектура «файл-сервер» или «клиент-сервер».
* Будет ли это 3-уровневая архитектура со следующими слоями: сервер, ПО промежуточного слоя (сервер приложений), клиентское ПО.
* Будет ли база данных централизованной или распределенной. Если база данных будет распределенной, то какие механизмы поддержки согласованности и актуальности данных будут использоваться.
* Будет ли база данных однородной, то есть будут ли все серверы баз данных продуктами одного и того же производителя (например, все серверы только Oracle или все серверы только DB2 UDB). Если база данных небудет однородной, то какое ПО будет использовано для обмена данными между СУБД разных производителей (уже существующее или разработанное специально как часть проекта).
* Будут ли для достижения должной производительности использоваться параллельные серверы баз данных (например, Oracle Parallel Server, DB2 UDB и т.п.).

Эти решения часто принимаются до начала этапа проектирования (на этапе анализа). На этапе проектирования полезно еще раз рассмотреть все причины выбора той или иной архитектуры, провести тесты производительности и надежности критических участков информационной системы. Это позволит избежать тяжело устраняемых ошибок проектирования. Довольно часто на этапе проектирования возникают непредвиденные технические проблемы, например аналитики не учитывают группу пользователей, имеющих доступ к информационной системе посредством текстовых терминалов. Это ошибка анализа, но выявляется она только на этапе проектирования. Такие проблемы должны решаться вместе с аналитиками, которые инициируют изменение информационной модели.

Если в среде используется сеть, то на этапе проектирования необходимо определить требуемые уровни сервиса сети и спроектировать ее топологию. Необходимо также провести тесты сети, чтобы увидеть, обеспечивает ли существующая сеть должную пропускную способность и имеется ли резерв пропускной способности сети. Если результат отрицательный, то следует четко описать необходимые изменения аппаратного обеспечения и топологии сети.

### Выделение потенциальных узких мест в информационной системе

Если заказчик заявит, что производительность системы не имеет никакого значения, примите это замечание с юмором. Это означает лишь то, что время ответа системы на запрос не является (или не кажется заказчику в данный момент) критическим. Попробуйте спросить, приемлемо ли время ответа системы, равное одному часу или одному дню. Вряд ли ответ на этот вопрос будет положительным.

Производительность важна для любой информационной системы. Узким местом называют момент падения производительности системы. Конкретный ответ на вопрос, где узкие места данной системы, может дать лишь специальное направленное тестирование. Но это не означает, что оценка потенциальных узких мест невозможна. Одним из хороших методов является график нагрузки на систему в течение дня, недели, месяца и т.п. Можно построить диаграмму, на которой будет отражено время работы тех или иных бизнес-процессов, а также требуемое для данного бизнес-процесса время ответа системы. Такие диаграммы помогают выявить момент, когда нагрузка будет наиболее интенсивной. Количество пользователей, одновременно работающих с тем или иным компонентом, отражается на диаграмме посредством весового коэффициента (рис. 1).

В приведенном примере явно видны 3 пика активности системы, максимальный из которых приходится на 11 часов. Использован тип диаграммы с накоплением.

А в диаграмме, представленной на рис. 2, видна активность касс в течение рабочего дня и повышение активности загрузки данных в нерабочее время. В такие диаграммы следует также добавлять вес, отражающий сложность бизнес-процесса, например в данном примере самый высокий весовой коэффициент будут иметь отчеты. Оценка весов определяется особенностями каждого конкретного бизнеса - где-то она может быть высокой, где-то низкой.

Ответ на вопрос, насколько потенциальные узкие места являются реальными, может дать только тестирование. Здесь оправданно применение специальных средств моделирования сценариев приложений. Следует отметить, что оценка точности детектирования узкого места в системе очень зависит от объема обрабатываемых данных. Следует уделить внимание генерации тестовых данных и проверке узких мест уже на этих данных. Часто информационная система не сразу выходит на проектную мощность, как правило, она работает некоторое время в режиме первоначального накопления информации, которое может продолжаться и несколько дней, и несколько месяцев. Как правило, предполагаемый порог объема обрабатываемых данных известен на этапе анализа, но реальный объем физических данных можно точно оценить только на этапе проектирования. Если сгенерировать предполагаемый объем тестовых данных нельзя (не хватает мощности техники или есть иные причины), то тесты проводят на меньшем объеме данных и пытаются построить оценки поведения системы на реальном объеме данных.

Более точно узкие места системы оцениваются на этапе разработки. Здесь уже есть реализованные компоненты системы. Средства автоматизации тестирования (например, LoadRunner, WinRunner и др.) позволяют отследить операции, которые выполняет то или иное приложение (но данные средства могут отследить далеко не все возможные типы приложений и то, насколько они подходят для тестирования вашего проекта, - это решение такого же порядка, что и выбор средства разработки приложения), автоматически сгенерировать сценарий запуска имитаторов работы реальных приложений и построить оценки узких мест системы.

### Продукты третьих фирм

На этапе проектирования оценивают возможность и эффективность использования продуктов третьих форм в разработке данной информационной системы. Например, существует задача выполнения некоторого набора работ (определенных пакетных заданий и т.п.) по заданному графику. Далеко не всегда целесообразно включать в проект создание утилиты контроля запуска приложений, поскольку есть масса утилит, выполняющих эти операции, в том числе и свободно распространяемых. Существует и другая причина, по которой с ПО третьих фирм следует хотя бы ознакомиться. Не факт, что в мировой практике решения задач, подобных вашей, не встречаются. Если реализации третьих фирм известны, то следует с ними ознакомиться хотя бы для того, чтобы не повторять неудачные решения и взять на заметку удачные. Вероятно, какой-либо из существующих продуктов может быть интегрирован в создаваемую вами информационную систему. Для этого, возможно, потребуется создать интерфейс обмена данными между ПО третьей фирмы и вашим. Следует оценить целесообразность как разработки собственного компонента, так и интеграции уже готового аналогичного компонента.

### Использование CASE-средств

CASE-средства предоставляют много преимуществ. На одной чаше весов будет автоматизация работы, предоставляемая CASE, а на другой - ненавистная задача преобразования результатов анализа в формат этого CASE (если для формализации результатов анализа использовался другой CASE-инструмент или не использовался никакой). Некоторые CASE-средства позволяют непосредственно перейти к проектированию, а к анализу можно вернуться путем обратного проектирования. К сожалению, при использовании обратного проектирования в CASE-средстве создается весьма вредная иллюзия того, что данные анализа регистрируются, хотя на самом деле этого практически никогда не происходит, поскольку информация, содержащаяся в спроектированной структуре, отличается от результатов анализа. Некоторые полезные данные получить можно, но построить полную картину вряд ли удастся.

### Инфраструктура

Для проектирования и реализации необходимы аппаратные ресурсы и специальное программное обеспечение. Кроме того, требуется механизм, позволяющий контролировать создаваемую документацию и код. Эти вопросы лучше решать на ранних стадиях проектирования, а не на стадии разработки. Мы поговорим об этом ниже, в разделе «Проектирование процессов и кода». При групповой разработке вам понадобятся средства контроля согласованности кода. Если разработка идет под разными платформами (аппаратная платформа и ОС), то хорошим решением может оказаться PVCS. Для платформ Windows 98, NT, 2000 может оказаться приемлемым решение, предлагаемое Microsoft - MS Source Save. Кроме того, многие средства разработки также предоставляют возможности контроля исходного кода.

### Проектирование базы данных

Здесь мы изложим задачи, касающиеся проектирования реальной базы данных.

### Построение модели данных

Работа проектировщиков базы данных в значительной степени зависит от качества информационной модели. Информационная модель не должна содержать никаких непонятных конструкций, которые нельзя реализовать в рамках выбранной СУБД. Следует отметить, что информационная модель создается для того, чтобы на ее основе можно было построить модель данных, то есть должна учитывать особенности реализации выбранной СУБД. Если те или иные особенности СУБД не позволяют отразить в модели данных то, что описывает информационная модель, значит, надо менять информационную модель, так как производитель СУБД вряд ли будет оперативно менять собственно СУБД ради вашего конкретного проекта (хотя и такие, правда единичные, случаи имели место).

Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных. После этого для разработчиков информационной системы создается пробная база данных. С ней начинают работать разработчики кода. В идеале к моменту начала разработки модель данных должна быть устойчива. Проектирование базы данных не может быть оторвано от проектирования модулей и приложений, поскольку бизнес-правила могут создавать объекты в базе данных, например серверные ограничения (constraints), а также хранимые процедуры и триггеры, - в этом случае часто говорят, что часть бизнес-логики переносится в базу данных. Проектирование модели данных для каждой СУБД содержит свои особенности, проектные решения, которые дают хороший результат для одной СУБД, но могут оказаться совершенно неприемлемыми для другой. Ниже перечислим задачи, которые являются общими для проектирования моделей данных:

* выявление нереализуемых или необычных конструкций в ER-модели и в определениях сущностей;
* разрешение всех дуг, подтипов и супертипов;
* изучение возможных, первичных, внешних ключей, описание ссылочной целостности (в зависимости от реализации декларативно или с использованием триггеров);
* проектирование и реализация денормализации базы данных в целях повышения производительности;
* определение части бизнес-логики, которую следует реализовать в базе данных (пакеты, хранимые процедуры);
* реализация ограничений (ограничений и триггеров), отражающих все централизованно определенные бизнес-правила, генерация ограничений и триггеров;
* определение набора бизнес-правил, которые не могут быть заданы как ограничения в базе данных;
* определение необходимых индексов, кластеров (если таковые реализованы в СУБД), определение горизонтальной фрагментации таблиц (если это реализовано в СУБД);
* оценка размеров всех таблиц, индексов, кластеров;
* определение размеров табличных пространств и особенностей их размещения на носителях информации, определение спецификации носителей информации для промышленной системы (например, тип raid-массивов, их количество, какие табличные пространства на них размещаются), определение размеров необходимых системных табличных пространств (например, системного каталога, журнала транзакций, временного табличного пространства и т.п.);
* определение пользователей базы данных, их уровней доступа, разработка и внедрение правил безопасности доступа, аудита (если это необходимо), создание пакетированных привилегий (в зависимости от реализации СУБД это роли или группы), синонимов;
* разработка топологии базы данных в случае распределенной базы данных, определение механизмов доступа к удаленным данным.

Подробнее на каждом из перечисленных пунктов мы остановимся в части «Схема базы данных».

### Создание базы данных для разработчика

Чаще всего базу данных создает администратор баз данных - если он есть; в противном случае это приходится делать проектировщикам. Физическая база данных нужна разработчикам информационной системы для разработки кода, а проектировщикам для проверки их идей. Проектировщики и разработчики могут работать как с одной и той же схемой, так и с разными схемами. В процессе разработки проекта, как правило, создается несколько версий схемы базы данных. Следует обязательно вести журнал изменений схемы (вручную или в репозитарии case) и жестко контролировать версии схемы.

### Проектирование процессов и кода

Параллельно с проектированием схемы базы данных требуется выполнить проектирование процессов, чтобы получить спецификации всех модулей. Если часть бизнес-логики хранится в базе данных (ограничения, триггеры, хранимые процедуры), то оба эти процесса проектирования тесно связаны. Главная цель проектирования заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. Определения модулей раскрываются в технической спецификации программ. Возможно, что некоторые атомарные функции, полученные на этапе анализа, вообще не будут отображены в какие-либо модули, а будут преобразованы в ручные процедуры или принципы работы.

### Выбор средств разработки

Не следует откладывать выбор средств разработки на самый последний момент. Если проектировщик не слишком хорошо представляет себе набор средств, которые будут использованы для разработки проекта, то следует для начала составить перечень возможных средств, затем провести консультации с техническими специалистами (хорошо знающими средства-кандидаты), оценить, с какими средствами персонал уже работал, а какие являются для них абсолютно новыми. Часто выбор средств разработки определяет именно фактор квалификации персонала.

### Отображение функций на модули

На этапе анализа уже разработан перечень функций, которые будут реализованы. На этапе проектирования этот перечень еще раз анализируется и корректируется. Однозначное соответствие между функцией и модулем вряд ли возможно. Дело в том, что на этапе анализа функции организованы по бизнес-категориям, а на этапе проектирования их придется реорганизовывать для упрощения разработки. Проектировщики могут принять решение объединить несколько функций, обладающих общими свойствами, или выделить какое-то общее свойство (или их набор) в отдельный модуль, а также разбить сложную функцию на несколько модулей. Подробнее эти вопросы рассматриваются в разделе «Спецификации функций».

### Интерфейсы программ

При проектировании модулей определяют разметку меню, вид окон, горячие клавиши и связанные с ними вызовы. Существуют два вида перемещения по программам:

* с контекстом, когда целевая экранная форма частично или полностью заполняется автоматически данными, связанными с теми, что находятся в исходной экранной форме;
* без контекста, когда целевая экранная форма не заполняется вовсе или частично заполняется автоматически данными, не связанными с теми, что находятся в исходной экранной форме.

Часто автоматически заполняемые данные экранной формы группируют (располагают рядом), а перемещение по заполняемым пользователем полям организуют так, как это делал бы сам пользователь, работая с реальным бумажным документом. Такие интерфейсы воспринимаются пользователем легче, и он намного быстрее осваивает новое ПО.

### Интегрирование и наследование механизмов обмена данными

Информационная система редко разрабатывается с нуля. Чаще проектировщики сталкиваются с задачами наследования данных из старых систем, которые уже выполняют какие-либо задачи автоматизации бизнеса. Такие системы могут на начальном этапе быть интегрированы в новую систему и постепенно заменяться новыми, более современными модулями. Этот подход может навязываться руководством фирмы для того, чтобы ускорить ввод новой информационной системы. Следует рассмотреть все плюсы и минусы такой постепенной интеграции (минусов, как правило, оказывается больше). Одну операцию придется делать в любом случае: переносить ценные данные, хранящиеся в старой информационной системе, в новую, то есть проектировать механизмы конвертации данных. Возможно, что придется делать конвертацию данных не только из старой системы в новую, но и обратно (полную или частичную), поскольку возможен вариант развития событий, при котором старая и новая информационные системы будут работать параллельно - хотя бы в период опытной эксплуатации новой системы.

Кроме вопросов наследования собственно данных из старых информационных систем, возможно, вам придется также решать задачи взаимодействия вашего ПО с продуктами третьих фирм. В этом случае вам следует изучить интерфейсы обмена данными ПО других разработчиков и обеспечить должный уровень поддержки этих интерфейсов в разрабатываемой информационной системе.

### Определение спецификаций модулей

Это основная часть функционального проектирования. Здесь решаются следующие задачи:

* преобразование функциональных определений анализа в реализуемые модули;
* спецификации, которые выражают функциональные возможности каждого модуля в физических категориях;
* определение средств разработки для каждого модуля (или выделенных групп модулей), если используются несколько средств разработки в одном проекте;
* определение последовательности реализации модулей и зависимостей модулей.

Спецификации модулей различают по степени детализации и содержанию даже в рамках одного проекта. Определяют, сколько времени требуется для того, чтобы сгенерировать тот или иной модуль, сколько необходимо на тестирование того или иного модуля, а также на тестирование совокупности сгенерированных модулей. Кроме того, следует разработать специальные метрики - шаблоны, которые позволяют оценить, сколько времени потребуется на создание исходного кода модуля. Для ускорения процесса разработки следует рассмотреть возможность использования генераторов исходного кода - это целесообразно, если вам предстоит разработать большое количество несложных модулей, а время разработки ограничено. Следует использовать шаблоны кода для устранения рутинных операций. Подробнее эти вопросы рассматриваются в разделе, посвященном спецификациям функций, - вы найдете его в одной из следующих статей данного цикла.