Paul Miller,
Fujitsu Australia Limited.

Создание эффективных схем в DOORS

Предисловие

Итак, вы приобрели DOORS, произвели инсталляцию, выбрали пилотный проект, босс с нетерпением ожидает обещанный возврат инвестиций, а Вы смотрите на экран монитора, пытаясь сообразить, что же делать дальше. Вы видели демо, может быть даже видели DOORS в действии во время семинара для пользователей, но как же теперь Вам самим заставить его делать все эти замечательные вещи?

Прелесть DOORS заключается не только в том, что его базовая архитектура разработана для управления требованиями, но и в том, что DOORS может быть настроен, для того, чтобы удовлетворять Вашим специфическим потребностям при управлении требованиями. Эта статья ознакомит Вас с процессом, который использовался Фуджитсу (Австралия) для того, чтобы выстроить такую схему DOORS, включающую в себя модули, атрибуты и представления, которая удовлетворяла бы требованиям модели SDLC (System Development Lifecycle), используемой компанией Фуджитсу в части управления требованиями и тестированием.

Эта статья также опишет приемы настройки DOORS в целях более полного использования возможностей продукта в управлении версиями документов, а также пояснит что делать с «подозрительными» входящими связями от объектов, которые были модифицированы позднее рассматриваемого.

Введение

Наличие грамотно сконструированных и гибких связей очень важно для завоевания одобрения пользователей с одной стороны и сохранения душевного равновесия администраторов DOORS. Плохо продуманные схемы могут снизить производительность инструмента, сделав очень сложным доказательство его эффективности, и, потенциально, вызвать у пользователей крайне негативное отношение к инструменту как к причине непроизводительных затрат времени и усилий.

Пользователям постоянно приходится овладевать различными программными инструментами, каждый из которых имеет множество нюансов. Использование схем, которые делают работу инструмента более прозрачной и логичной, позволит облегчить жизнь пользователям и тем самым достигнуть их позитивного восприятия инструмента. Использование же слишком громоздких и сложных в администрировании схем, особенно требующих постоянных «заплаток», может привести к отторжению инструмента конечными пользователями.

Таким образом, Вам нужна схема, удовлетворяющая потребности пользователей, простая в сопровождении, легкая в изменении и функционирующая без сбоев.

Разработка схем в DOORS

Если Вы хотите, чтобы Ваши схемы сформировали стройную систему, Вы должны руководствоваться несколькими базовыми принципами, лежащими в основе дисциплины разработки систем. Неплохо начать с принципа «форма определяется функциональльностью». То есть, форма удовлетворяет цели только в том случае, если она выбрана исходя из проблемной области, для которой создается решение. Другими словами, не создавайте решение, оторванное от реальной проблемы. Возвращаясь к DOORS, Ваши схемы будут действительно полезны, если у Вас есть определенный и ясный в понимании процесс создания спецификация на разрабатываемые продукты. Если у Вас уже есть процедуры, определяющие жизненный цикл разработки спецификаций, они послужат прекрасным источником требований по настройке схем в DOORS с целью их соответствия Вашему процессу разработки спецификаций.

Схема, описываемая в этой статье, базируется на соответствии и улучшении существующих циклов разработки спецификаций и тестирования в рамках нашей SDLC модели.

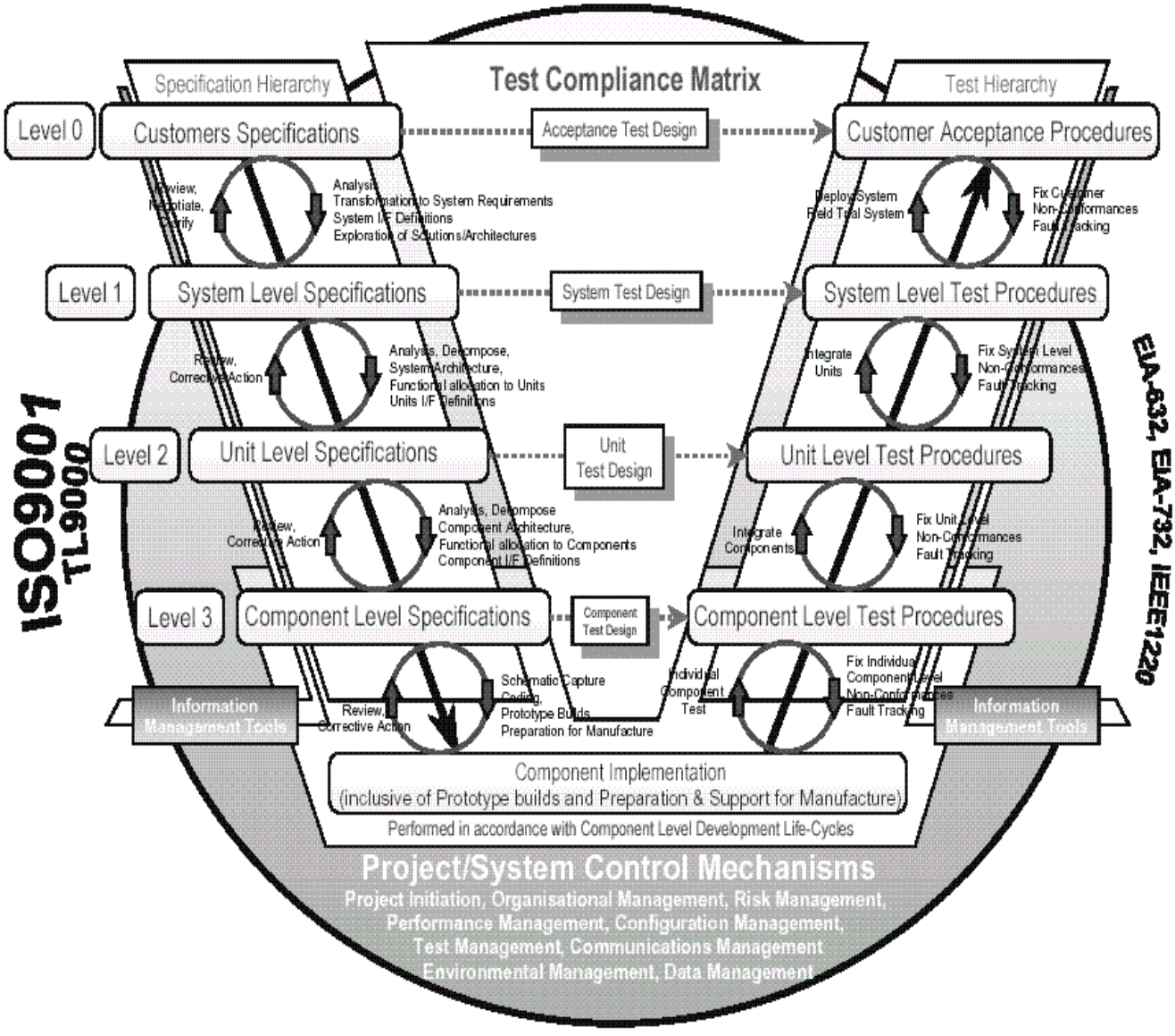

Рис.1 System Development Lifecycle Model (SDLC)

System Development Lifecycle (SDLC)

Схема на рис. 1 описывает SDLC модель, принятую научно-исследовательским центром Фуджитсу (Австралия). Эта модель есть функциональное отображение структуры связей важнейших элементов информации, которые определяют (Specification Hierarchy), реализуют (Component Implementation) и управляют (Project/System Control Mechanisms) разработкой сложных систем. Представление функциональных определений для каждого из этих элементов выходит за рамки данной статьи. Для понимания этой статьи важно лишь следующее:

1. Иерархия спецификаций (Specification Hierarchy) представляет собой последовательную и отслеживаемую декомпозицию родительских требований на одном уровне по отношению к дочерним на уровень ниже. Такой подход отражает концепцию **валидации**, где низкоуровневые требования могут быть отслежены вверх до порождающих их бизнес-требований и пожеланий клиентов.
2. Матрица результатов и тестов, а также иерархия тестов (Test Compliance Matrix & Test Hierarchy) позволяют прослеживать требования от спецификаций до тестовых процедур.Это представляет концепцию **контролируемости**, где каждое требование должно быть проверено и протестировано по меньшей мере один раз. Иерархия тестов (Test Hierarchy) имеет ту же структуру, что и иерархия спецификаций (Specification Hierarchy), что отражает концепцию того, что тесты должны налагаться на требования на том же уровне, на котором они появляются в иерархии спецификаций.

Схема DOORS, разработанная для Фуджитсу, строилась таким образом, чтобы поддерживать взаимосвязи, информацию и функциональные потребности иерархии спецификаций, иерархии требований и матрицу результатов тестов, представленные в вышеописанной модели.

Создание схем в DOORS

Подсказка номер один – не усложняйте схему сверх надобности. Каждая дополнительная деталь означает трудозатраты не только сейчас, но и в будущем. Когда потребуются модификации, Вам, скорее всего, придется выполнять их во многих проектах и модулях. Для того, чтобы сохранить свое время, каждый раз убеждайтесь, что добавляемая пользовательская настройка действительно полезна и будет использоваться.

Разрабатывая схему, проделайте следующее:

1. Определите необходимый набор типов модулей: Спецификации, Тестовые процедуры, Архитектура и т.д.
2. Определите необходимый набор видов отображения информации (View) и составьте описание содержимого колонок, формирующих эти виды.
3. Определите взаимотношения между модулями в терминах связей и специфические атрибуты, необходимые для управления этими связями.
4. Определите права пользователей на доступ к проектам, модулям, объектам, видам, атрибутам и т.п.
5. Определите список DXL скриптов, необходимых для предоставления дополнительной пользовательской функциональности.
6. Разработайте синтаксис наименований модулей для облегчения навигации и нахождения модулей в окнах DOORS.

7. Разработайте стандартные форматы печати и макеты для печатной продукции.
8. Подумайте об общем характере внешнего вида каждого модуля в отдельности и всех их в совокупности. Пользователям понравится, если внешний вид всех модулей будет единообразен.
9. Продумайте возможное взаимодействие с внешними средствами управления информацией или средстами разработки.

Шаг первый. Определите типы модулей.

В первую очередь выберите, какие из общих типов модулей Вам потребуются. Если Вы понимаете процесс создания Информационной Архитектуры - это очень Вам поможет. В нашем случае, SDLC модель играет роль представления архитектуры основных источников важной информации.

Анализ SDLC модели показывает, что нам требуется как минимум два базовых типа модулей, Спецификации и Тестовые процедуры. На основе этих базовых типов могут создаваться разные вариации типов, но так как они будут создаваться на основе одного базового типа, у них будет единообразное представление и поведение. Каждый базовый тип был настроен указанием атрибутов соответствующих объектов, видов, макетов заголовков и базовых прав доступа. Эти модули были сохранены как восстанавливаемые шаблоны модулей. Что-то вроде .dot-файлов шаблонов в Microsoft Word.

Инструкции по разработке представлений, атрибутов объектов, прав доступа и DXL скриптов для включения в базовые типы шаблонов модулей будут рассмотрены ниже.

Каждый базовый тип модуля включает в себя набор атрибутов. Это помогает в управлении общей схемой при помощи описания каждого экземпляра модуля. Также это увеличивает производительность DXL скриптов, которые должны будут оперировать над определенной выборкой модулей. Примеры атрибутов уровня модуля приведены ниже:

- Base Type (перечисление): Specification Module, Test Module.
- Base Type Version (строка): версия базового типа, изменяется когда вносятся изменения в базовый тип и экземпляры обновляются для синхронизации с базовым типом.
- SDLC Hierarchical Position (перечисление): Customer Level, System Level, Unit Level, Component Level.
- Model Element (перечисление): Specification Hierarchy, Test Hierarchy, Project Management, Support

Эти атрибуты уровня модуля помогают скриптам DXL сфокусировать свои операции на соответствующих группах модулей. Например: сбор данных только из модулей спецификаций уровня System Level и тестовых процедур, модификация данных во всех модулях спецификаций уровня Unit Level, проведение аудита версий используемых базовых типов.

При модификации схемы базового типа модуля, изменения в первую очередь накладываются на шаблон модуля. При каждом изменении соответственно изменяется атрибут Base Type Version шаблона модуля. Существующие экземпляры модулей, использующих этот базовый тип, вручную изменяются и их атрибут Base Type Version модифицируется, чтобы соответствовать шаблону. Это и есть причина, по которой схема не должна быть слишком сложной, иначе каждое изменение в базовых типах будет приводить к значительным усилиям по модификации экземпляров модулей, использующих модифицированные базовые типы во всех проектах. DXL скрипты, конечно, могут облегчить ситуацию, но если схема очень сложная и подвержена постоянным модификациям - возрастает вероятность ошибок, связанных с работой скриптов.

Каждый раз, когда необходимо создать новый экземпляр модуля спецификаций или тестовых процедур, соответствующий сохраненный базовый шаблон должен быть загружен в соответствующий проект. При этом должны быть соответствующим образом изменены атрибуты уровня модуля.

Шаг второй. Продумайте виды (Views).

Следующим шагом шагом определитесь, какого рода базовая информация может понадобиться различным пользователям, имеющим доступ к модулю. Крайне рекомендуется посоветоваться с пользователями о том, какую информацию они должны видеть. Это будет иметь далеко идущие благоприятные последствия.

Каждый вид определяется как набор колонок, которые представляют из себя атрибуты DOORS (мы обсудим их в следующей секции). Инструмент вроде Microsoft Excel может быть использован, чтобы быстро набросать прототип представления и получить отзыв пользователей.

Рассмотрим представления модуля спецификаций. Каждый пользователь захочет видеть основной текст требований и их уникальные идентификаторы. Поэтому эти данные должны появляться во всех представлениях. Менеджеры захотят получить возможность утверждать, устанавливать приоритеты, назначать ответственных, проверять запросы на изменения. Поэтому им нужен свой, особый вид отображения информации. Инженерам-тестировщикам потребуется оценить проверяемость требований и сопоставить требования тестовым процедурам. Поэтому им тоже нужно свое особое представление. Авторы и читатели захотят иметь представления, отображающие подробности об источниках входящих ссылок и целях исходящих ссылок.

В целях сбора статистики, а также избавления от неопределенностей, убедитесь, что хотя бы одно представление содержит колонку, декларирующую тип информации объекта. Например, отображает ли колонка настоящее требование или просто некую информацию? Служебные записи, такие как вступление, цель, определения, заголовки и т.п. обычно не включаются в статистику по требованиям. Это позволит Вам отфильтровать объекты, которые не являются требованиями как таковыми.

Вот несколько примеров видов отображения информации:

Вид по умолчанию: это такой вид , который видят пользователи при открытии модуля. Он должен отображать наиболее важную информацию, такую как текст самого требования, уникальный идентификатор (UID), тип информации, несколько атрибутов.

Вид архитектурного дизайна: содержит атрибуты со ссылками на сценарии использования или какую-либо другую информацию о дизайне, используемую для получения более детальных «дочерних» требований уровнем ниже.

Вид для менеджеров: имеет атрибуты - утверждение, приоритет, кто отвечает, рассмотрение предложений на изменения.

Вид родительских и дочерних узлов: выводит полный текст и уникальный идентификатор входящих родительских и исходящих дочерних требований.

Вид истории изменений: атрибуты могут отображать такую информацию, как автор требования, дата последнего изменения и т.д.

Вид планирования тестов: может отображаться все то, что связано с тестированием, - планируемый метод (тест, документация, инспекция, расчет), тестовая процедура, которую планируется использовать и т.д.

Когда все виды разработаны, защитите их от нежелательной модификации установкой особых прав доступа, таких, чтобы только администраторы могли бы изменить их представление.

Подумайте о префиксах в названиях видов, которые помогли бы сортировать виды в списке в том порядке, который Вы бы хотели.

Также обратите внимание на дополнительные свойства видов, такие как сортировка, фильтрация, текущий объект, позиция окна и т.д. Пользователи будут раздражаться, если при каждом открытии нового модуля колонки модуля будут прыгать с места на места, отключаться фильтры и сортировка.

Научите пользователей сохранять персональные представления с пользовательским именем (логином) в качестве префикса. Это позволит группировать пользовательские представления, не смешивая их с базовыми представлениями схемы.

Шаг третий. Атрибуты.

Теперь, когда продуманы необходимые пользователям виды, мы должны создать соответствующие атрибуты для каждой колонки с информацией.

Для того, чтобы Вам было легче управлять атрибутами создаваемой схемы, используйте префиксы в их именах. Это позволит группировать атрибуты и отделить их от атрибутов, встроенных в DOORS.

Рекомендуется защитить определение атрибутов базовой схемы при помощи ограничения прав доступа к определению каждого атрибута. С тем, чтобы только администраторы могли менять определения атрибутов.

Ниже мы приведем несколько подсказок, касающихся установки свойств атрибутов.

Будьте осторожны со свойством «Inherit» («Наследование»). Если его установить одновременно со свойством «Affect Change Bars», только родительский объект будет восприниматься системой как модифицированный, даже если дочерние объекты унаследуют изменения. Также, если изменить значение атрибута дочернего объекта, его режим наследования изменится с «acquired» (унаследованный) на «specific» (индивидуальный, т.е. не унаследованный). Этот атрибут уже не будет автоматически изменять свое значение при изменении родительского объекта. Это может ввести пользователей в заблуждение, поскольку они могут полагать, что при изменении родительского объекта, изменения наследуют все дочерние объекты. Но все дочерние объекты с индивидуальными значениями атрибутов останутся неизмененными.

Не устанавливайте свойства «Affect Change Bars» и «Affect Change Dates» слишком часто. Пользуйтесь ими только для критически важных атрибутов. Это поможет пользователям отслеживать критические изменения в истории изменений.

При использовании в атрибутах DXL скриптов, настоятельно рекомендуется хранить их с использованием внешнего средства версионного контроля. Более подробно эта тема будет рассмотрена в разделе «Шаг шестой. Использование DXL.».

Шаг четвертый. Права доступа.

В предыдущих частях уже прозвучало несколько советов в отношении установления особых прав доступа для защиты видов и атрибутов Вашей схемы от нежелательной модификации.

На уровне модуля установите права доступа к базовым шаблонам модуля. Отмените наследование от родителя и установите права только на чтение для группы «Everyone else». При этом при загрузке шаблона модуля только пользователь с именем Administrator будет иметь изначальный доступ. Авторам можно предоставить доступ на чтение и запись. DOORS имеет очень гибкую систему прав доступа. Некоторым пользователям можно предоставить ограниченные права администрирования, чтобы не было необходимости в каждом случае задействовать логин Administrator.

При определении прав доступа отдельных пользователей для совместного редактирования объектов (shared) подходите к этому проще и не углубляйтесь слишком далеко в иерархию объектов. По возможности предоставляйте такие права обычной группе пользователей, а не персонально каждому.

Шаг пятый. Модули связей (link).

Уделите внимание модулям связей . Не стоит помещать все связи в один модуль, поскольку в последствии это осложнит использование различных видов анализа.

В случае нашей SDLC модели, были созданы отдельные модули связей для каждого типа связей между элементами модели. Например, полезно иметь связи между спецификациями уровней System Level Specification и Unit Level Specification помещенными в отдельный модуль связей, связи между уровнями System Level Specification и System Level Test Procedures – в своем отдельном модуле связей и т.д.

Свойства модуля могут быть установлены таким образом, что при создании пользователем связи при помощи механизма drag-and-drop, информация о связи автоматически помещалась в соответствующий модуль связей . Если этого не сделать, придется учить пользователей проверять правильность выбранного модуля связей перед тем, как создавать их с помощью drag-and-drop.

Такой метод назначения типов связей определенным модулям связей также помогает в лучшем контроле прав на установление связей.

Шаг шестой. Использование DXL.

Скрипты DXL – мощнейший инструмент DOORS. Они легки в написании и способны обеспечить действительно ценные функции, замещающие сложные и чреватые ошибками ручные операции.

Тем не менее, не относитесь слишком легкомысленно к процессу написания кода DXL только потому, что это – скрипты. Написание DXL скриптов - такой же ответственный процесс, как и написание любого другого программного обеспечения. DXL скрипты должны быть документированы, написаны с соблюдением общего стиля кодирования и помещены под конфигурационный контроль. Скорее всего, Вам потребуются определенные усилия для поддержки и сопровождения используемого пакета DXL скриптов. Поэтому не используйте чрезмерно большого количества активных DXL скриптов.

Старайтесь не дублировать функции общего назначения в разных DXL скриптах. Это может привести к ошибкам, если Вы модифицируете такую функцию в одном скрипте, не модифицировав соответствующим образом все остальные скрипты. Правильным подходом будет сохранение функций общего назначения в *.inc файлах. Тогда у Вас будет всего один экземпляр такой функции.

Управление версиям документов

Как Вы уже, наверное, догадались, DOORS имеет встроенный функционал работы с версиями документов (baselines). Перед тем, как установить очередную версию документа, необходимо провести аудит всех предложенных и реализованных изменений. Вы можете сделать эту процедуру более быстрой и надежной, используя функциональность помощника установки версий документов DXL Baseline Assistant. Этот помощник предлагает список функций, собирающих важную информацию и способных обнаружить проблемы, которые может не заметить оператор, проводящий аудит. Использование этого инструмента позволяет сконцентрироваться на проверке тех критериев, которые затруднительно проверить в автоматическом режиме. Вот некоторые полезных функций, выполняемых помощником:

Pre Baseline Statistics (Предварительная статистика): общее число объектов, помеченных как требования (не information-only объекты), общее число требований, помеченных на удаление, общее число новых требований, добавленных позднее последней версии документа, общее число требований, модифицированных позднее последней версии документа. Эта полезная для проведения аудита информация, характеризующая общую стабильность спецификации.

Find Unresolved Change Proposals (Неразрешенные предложения на изменения): Находит объекты, связанные с предложениями на изменения, находящимися в состоянии «новое». Например, с предложениями, по которым еще не принято решение.

Find General Rules Violations (Нарушения основных правил): Например, мы имеем определенные правила стиля, позволяющие контролировать целостность информации, такие, как запрет на одновременное заполнение текстом заголовка объекта (Object Heading text) и его основного текста (Object Text).

Perform Requirements Quality Check (Проверка качества требований): Осуществляет в текстах требований поиск слов, которые объявлены как неоднозначные определения, допускающие ложное толкование.

Create Parent Child Snapshot (Снимок родительских и дочерних узлов): При фиксировании версии модуля желательно сохранить значения объектов других модулей, связанных с объектами данного модуля входящими или исходящими ссылками. Так как объекты в других модулях будут со временем меняться, такой снимок поможет зафиксировать их состояние на момент фиксирования версии модуля. Эта функция позволяет сохранить полученную информацию в виде (View), в котором текущие значения связанных ссылками объектов будут выводиться в соседних столбах со значениями, зафиксированными при установке версии модуля.

После проведения вышеуказанных проверок может быть сгенерирован отчет с указанием уникальных идентификаторов (UID) «подозрительных» объектов. Такой отчет может быть представлен на рассмотрение комиссии по аудиту, как основание для принятия решения о фиксирования версии документа.

Некоторые функции могут быть использованы после фиксирования версии документа. Например, сразу после фиксирования версии документа можно удалить помеченные на удаление объекты, так как их значения зафиксированы в текущей версии и более не нужны в следующих версиях.

> *Примечание переводчика: вышеописанная функциональность присутствовала в версии DOORS v.5.x. В текущих версиях DOORS v.9.x DXL Baseline Assistant уже не используется. Более подробную информацию о работе с версиями документов Вы можете найти в руководстве администрирования DOORS.*

«Подозрительные» ссылки

Возможность связывать объекты и отслеживать эти связи – одна из самых мощных возможностей DOORS. Особенно важно знать, что происходит с объектом, который является источником входящей связи.

Например, инженеры-тестировщики работают с тестовыми процедурами, имеющими исходящие связи к требованиям, которые, возможно, находятся в процессе модификации. Даже несмотря на то, что авторы требований могут очень ответственно относиться к последствиям своих модификаций, обычный временной прессинг в ходе выполнения проекта может осложнить своевременное информирование всех членов команды о внесенных коррективах.

Использование механизма отслеживания «подозрительных» связей позволяет при внесении изменений в объекты верхнего уровня (в нашем случае - требования), связанные входящими связями с объектами нижнего уровня (в нашем случае – тестовые процедуры), специальным образом помечать эти объекты нижнего уровня (тестовые процедуры) при условии, что объекты верхнего уровня (требования) были удалены или модифицированы позже момента создания объектов нижнего уровня (тестовых процедур).

DOORS имеет встроенный механизм отслеживания «подозрительных» связей. Тем не менее, Вы можете создать собственный, включив в него некоторую дополнительную функциональность. Например, отслеживание объектов верхнего уровня (требований), помеченных на удаление.

Средство отслеживания «подозрительных» связей может быть встроено для работы в реальном времени в форму скрипта DXL Layout. В нашем случае, такой скрипт трассирует исходящие связи от тестовых процедур до требований и выводит текст тестовой процедуры, «подозрительного» требования и их уникальные идентификаторы.

Если требование было помечено на удаление, скрипт выводит текст DELETED и уникальный идентификатор требования рядом с текстом рассматриваемой тестовой процедуры. Кроме того, скрипт меняет цвет выводимого текста на красный, чтобы привлечь внимание авторов.

Если же требование было модифицировано позднее рассматриваемой тестовой процедуры, то исходящая связь рассматривается как «подозрительная», и скрипт выводит текст MODIFIED и уникальный идентификатор измененного требования рядом с текстом рассматриваемой тестовой процедуры. Скрипт меняет цвет выводимого текста на красный, чтобы привлечь внимание авторов.

Если рассматриваемая тестовая процедура и требование, с которой она связана, были модифицированы в один день (запись даты модификации объектов ведется с точностью до дня), скрипт исследует историю измененного требования. Он находит запись о том, какой именно атрибут требования был модифицирован, время его модификации с точностью до секунды и сравнивает его с соответствующими показателями рассматриваемой тестовой процедуры.

Когда автор тестовой процедуры находит «подозрительную» связь, он может обсудить модификацию требования с его автором и узнать, необходима ли модификация рассматриваемой тестовой процедуры.

В том случае, если требование было модифицировано позднее рассматриваемой тестовой процедуры, автор процедуры узнает о необходимых изменениях в своем объекте, вносит их, тем самым изменяя время последней модификации процедуры и снимая пометку MODIFIED с исходящей ссылки. Даже если нет необходимости в модификации рассматриваемой тестовой процедуры, простое добавление и удаление пробела в ее тексте поможет снять привлекающую внимание пометку MODIFIED с исходящей связи.

Если требование было помечено на удаление, автор рассматриваемой тестовой процедуры обсуждает ситуацию с автором помеченного на удаление требования. Снять

пометку DELETED с исходящей связи можно двумя способами: снять пометку на удаление с требования, к которому направлена связь, или удалить саму исходящую связь.

Обе эти метки, MODIFIED и DELETED, помогают привлечь внимание авторов к возникшей проблеме и инициируют ее разрешение. Дополнительные скрипты могут проводить такого рода аудит по всем проектам и представлять результаты аудита в ежедневных отчетах.

Заключение

Последний совет: потратьте время на то, чтобы выработать четкое представление о том, как должен выглядеть Ваш процесс разработки ПО, которым вы хотите управлять при помощи DOORS. И уже после этого начните настраивать DOORS таким образом, чтобы он соответствовал Вашему процессу. Мы использовали в качестве основы процесса вышеописанную модель SDLC и настройка DOORS велась с целью адаптации инструмента к этой модели.

Если у Вас еще нет аналога SDLC модели, проведите анализ Вашей информационной архитектуры, разработайте соответствующие блок-схемы процессов составления спецификаций и тестовых процедур. Вознаграждением за хорошо спланированную схему будет гибкая и надежная система управления требованиями, которая будет удовлетворять пользователей и легко администрироваться.

Адаптированный перевод с английского статьи
*“**Creating and Managing a Resilient DOORS Schema**”,*
Paul Miller (Manager – Quality & Systems Administration Department, Fujitsu Australia Limited) выполнен компанией Telelogic Rus, Москва.