Глава 3
Спецификация требований

Предисловие

Одной из наиболее важных задач в работе инженеров является определение требований к разрабатываемой системе. Это очень ответственный шаг в любом налаженном процессе разработки. К сожалению, он часто не выполняется с должным качеством. Лишь немногие инженеры обучены процедурам определения и управления требованиями, что приводит к созданию документов, описывающих требования в стиле "викторианского романа". При этом, документы часто получаются неполными, несогласованными и неоднозначными. Поскольку такие документы преимущественно создаются с использование текстовых описаний, они по сути своей не обладают точностью, достаточной для их проверки до начала этапа проектирования. Это означает, что невозможно оценить качество этих требований каким либо надежным способом. Ошибки в требованиях – это один из наиболее серьезных видов стратегических ошибок в системах и наиболее дорогие в исправлении, поскольку вносятся они на ранних стадиях, а обнаруживаются, как правило, на поздних стадиях проекта, оказывая влияние на многие компоненты системы . UML и процесс Harmony предлагают лучший способ.

Хотя UML не позволяет явным образом отображать требования, он дает возможность отображать связанные наборы требований с использованием концепции варианта использования. Вариант использования может восприниматься как операционное представление системы для одного общего способа ее использования. Будучи таковым, он неявно "содержит" связанный набор требований относящихся к данному способу использования системы. Более детальное использование системы, называемое сценарием, определяет одно конкретное реализовавшееся взаимодействие с системой в рассматриваемом варианте использования. В терминах DODAF и MODAF, варианты использования и сценарии могут использоваться для описания "операционной архитектуры" для взаимодействующих операционных узлов. С точки зрения операционного представления, варианты использования и сценарии являются ключевыми для понимания каким образом операционные узлы (которые в результате будут реализованы в физических системах [1])) взаимодействуют с другими элементами из окружения системы.

Мы также можем моделировать требования путем их спецификации, а не только с использованием операционного представления. Спецификации определяют

[1] Операционный узел может рассматриваться как роль в "операционной архитектуре". Например, "Платформа наблюдения", которая в процессе выполнения миссии, реализуется физическими системами, такими как "Спутник" или "AWACS".

статические и динамические свойства системы. В терминах DODAF/MODAF они соответствуют "системной архитектуре". Для спецификации требований нам необходимо иметь возможность точно и непротиворечиво описывать различные аспекты разрабатываемой системы. Для создания точных спецификаций идеально подходят диаграммы состояний и диаграммы деятельности языка UML.

Вопрос заключается в том, как наилучшим образом определять требования к системе на основе операционного представления и представления спецификаций. Если вы не всегда знали, как это делать на привычном нам русском языке, то насколько сложнее это будет сделать на гораздо менее распространенном и понятном нам языке, таком как UML!

В этой главе предлагается ряд заданий, которые позволят вам попрактиковаться и получить опыт в определении требований, как с использованием операционного представления, так и представления спецификации. Решения к этим заданиям содержатся в главе 8. Сразу скажем, что не существует единственно правильного решения. Существует множество возможных способов для моделирования требований, которые обеспечат их ясное, полное, непротиворечивое и согласованное описание. Разумеется, существует гораздо больше *неправильных* способов для моделирования требований; тем не менее, хороших способов ничуть не меньше. И если ваши решения отличаются от приведенных в этой книге, это отнюдь не значит, что они являются неправильными. Опираясь на собственный опыт, я могу сказать, что наиболее частые ошибки в применении UML связаны с вариантами использования. Я считаю, что это происходит не вследствие того, что варианты использования сами по себе трудны для понимания, а вследствие того, что немногие инженеры обладают достаточной подготовкой в области сбора требований. Многие из них ограничиваются тем, что пытаются делать то, в чем *они* более компетентны -- а именно проектировать систему, используя варианты использования -- вместо того, чтобы собирать и анализировать требования независимым от реализации способом.

Задание 3.1: Идентификация типов требований для системы управления дорожным перекрестком

В UML отсутствует отдельная сущность с названием "требование". Хотя варианты использования, диаграммы последовательности, конечные автоматы, диаграммы деятельности и ограничения разных видов могут использоваться для моделирования свойств требований, отдельная сущность с названием требование отсутствует. По этой причине данная сущность была добавлена в языке SysML (Systems Modeling Language – Язык Моделирования Систем), а Rhapsody позволяет нам добавлять элементы требований в моделях.

Удобно воспринимать требование в UML как элемент модели, который определяет некоторый аспект системы с внешней, а не внутренней, точки зрения. Обратите

внимание, что требования не являются классами, хотя как и классы они существуют только во время проектирования, так как на основе классов создаются экземпляры, а на основе требований нет. Требования – это в некотором смысле ограничение "правильности", определяемое для системы.

Я считаю, что существует несколько различных типов требований, как это показано на Рис. 3-1. *Операционные требование* определяет каким образом система должна взаимодействовать с другими элементами (действующими лицами) в ее окружении. Например, если беспилотный летательный аппарат (БЛА) должен взаимодействовать с несколькими GPS-спутниками, чтобы ориентироваться на местности, тогда то как производится данное взаимодействие является операционным аспектом данной системы. *Функциональные требования* имеют отношение к поведению системы. Если БЛА должен уметь наводить ориентируемый оптический сенсор на определенную точку поверхности земли, то это будет являться функцией, реализуемой в этом БЛА. Если функциональное требование формулируется при помощи глагола, то требование к качеству, будет выражаться наречием, поскольку оно будет отвечать на вопрос "сколько", "как много". Для ориентируемого оптического сенсора примером требований к качеству могут быть: датчик должен обеспечивать возможность регулирования с шагом 0.1 градуса и с точностью 0.01 градуса. Параметрическое требование не является ни операционным, ни функциональным. Предположим, что БЛА должен быть сконструирован так, чтобы его вес не превышал 2100 фунтов. Вес БЛА является свойством системы, которое может быть статическим или динамическим, в зависимости от ситуации. Однако это свойство не является ни поведенческим, ни операционным; тем не менее, это требование необходимо включить. И, наконец, *проектные требования* имеют отношения *непосредственно* к проектированию, а не к самой системе. Примерами проектных требований могут быть: БЛА должен быть спроектирован так, чтобы в следующей его версии возможно было бы повторно использовать не менее 70% от созданной модели, или расходы на поддержку системы не должны превышать 1200000 долларов США (для проекта Койот такое требование является весьма реалистичной оценкой).

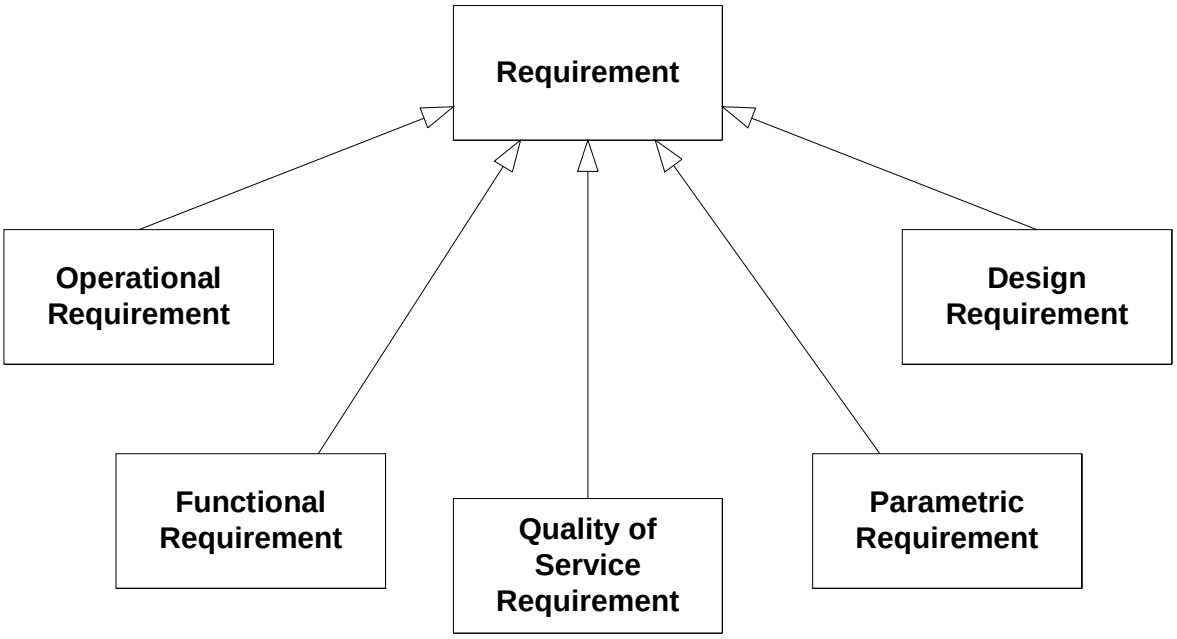

Рис. 3-1: Типы требований

Мы моделируем данные типы требований различными способами. Например, для моделирования надежных коммуникаций удобно использовать диаграммы состояний; однако у вас не получится использовать диаграмму состояний для отображения требования, что «нос БЛА должен быть темно-серого цвета».

В данном задании вам необходимо создать новую модель и добавить две диаграммы (подойдут как диаграммы вариантов использования, так и диаграммы моделирования объектов). Назовите одну из этих диаграмм “Обзор основных требований”, а другую – “Требования к специальному режиму”. Проанализируйте формулировку задачи для системы управления дорожным перекрестком в приложении А до раздела "Конфигурационные параметры", и добавьте в модель каждое требование в виде отдельного элемента. Свяжите полученные элементы требований друг с другом отношениями зависимости, используя следующие стереотипы:

- «включает» -- когда большее требование включает меньшее.
- «расширяет» -- когда меньше требование уточняет или добавляется к главному требованию.
- «вытекает» -- когда требование являет более детальной переформулировкой другого.
- «разъясняет» -- когда некий комментарий разъясняет смысл требования.

Задание 3.2: Определение вариантов использования для системы управления дорожным перекрестком

Варианты использования определяют возможные способы использования системы с точки зрения внешнего операционного представления. Ниже перечислены отличительные признаки вариантов использования:

- Предоставляет значимый результат как минимум для одного действующего лица.
- Содержит как минимум три сценария, каждый из которых содержит несколько сообщений, которыми обмениваются действующие лица и система.
- Логически содержит множество операционных, функциональных требований или требований к качеству.
- Не содержит информации о внутренней структуре системы.
- Не зависит от других вариантов использования и может выполняться параллельно с ними (возможность параллельного выполнения – необязательное требование)
- На самом "верхнем уровне" не должно быть менее трех и не более трех десятков вариантов использования.
    - "Крупные" варианты использования, могут быть подвергнуты декомпозиции на более мелкие варианты использования с использованием отношения зависимости со стереотипами «включает» и «расширяет» .
    - Варианты использования могут быть специализированы при помощи отношения обобщения, которое показывает, что один вариант использования является уточнением другого варианта использования.

Идентифицировать варианты использования можно разными способами. Некоторые аналитики предпочитают сначала определять сценарии взаимодействия, а затем группировать их в группы, связанные общим способ использования системы. Другие предпочитают вначале определять способы использования системы, и уже затем детализировать их при помощи многочисленных сценариев. Третьи вначале определяют действующих лиц, взаимодействующих с системой, и фокусируются на способах взаимодействия каждого из действующих лиц с системой.

Выберите одну из описанных выше стратегий и определите варианты использования для системы управления дорожным перекрестком. Также определите действующих лиц во внешнем окружении системы и создайте для системы диаграмму использования. После этого проверьте удовлетворение всем перечисленным выше критериям для того, чтобы убедиться, что ваши варианты использования выбраны верно.

Задание 3.3: Группировка требований по вариантам использования

Так как в UML отсутствует сущность "требование", мы свободно можем добавлять новые типы зависимостей, которые могут быть нам полезны для того чтобы связать данные элементы модели. В дополнение к описанным выше стереотипам для отношения зависимости между требованиями, я также использую стереотип «специфицирует» для того чтобы показывать, что вариант использования частично специфицируется элементом- требованием. Я рисую отношение зависимости от варианта использования к элементу требования.

Возьмите один вариант использования, идентифицированные в предыдущем задании, и создайте для него новую диаграмму использования. На этой диаграмме воспользуйтесь отношениями зависимости для привязки операционных требований к данному варианту использования, тем самым уточнив вариант использования. Вы можете также добавить отношения между самими требованиями, где это необходимо.

Задание 3.4: Идентификация вариантов использования для системы БЛА Койот

В своей практике я встречался с моделями, которые содержали 500 и более вариантов использования одного уровня. Такие модели было практически невозможно использовать. *Всегда помните одно из приведенных ранее критериев качественных вариантов использования: На самом "верхнем уровне" должно быть не менее трех и не более трех десятков вариантов использования.* Если система является очень сложной, то необходимо определять отношения между вариантами использования разного уровня абстракции. Варианты использования нижнего уровня все равно будут присутствовать в модели, но благодаря определенным отношениям мы сможем видеть требования как в иерархическом представлении, так и в горизонтальном. На практике я пришел к выводу, что это является ключевым аспектом полезности моделей вариантов использования для сложных систем.

В данном задании вам предстоит идентифицировать варианты использования для системы БЛА Койот (см. приложение В). Из-за сложности данной системы вначале необходимо определить "высокоуровневые" варианты использования, которые затем декомпозировать на более детальные варианты использования. Используйте стереотип «включает» для отношений зависимости, чтобы показать отношения между вариантами использования разных уровней абстракции. Добавьте новый пакет для хранения параметрических требований, таких как вес системы и вес полезной нагрузки. Установите для этого пакета стереотип «параметрические» для того, чтобы отобразить, что он будет содержать параметрические требования к системе. Проверьте полученную модель на соответствие критериям, сформулированным в задании 3.2.

Задание 3.5: Определение параметрических требований

Определите параметрические требования для системы БЛА Койот. Помните, что данные требования определяют неоперационные аспекты системы, такие как физические характеристики. Добавьте эти элементы в пакет, созданный для этих целей в предыдущем задании. Создайте диаграмму использования, отображающую данный пакет и содержащиеся в нем элементы требований.

Задание 3.6: Определение требований к качеству для вариантов использования

Варианты использования – являются наборами связанных между собой функциональных требований. Для названий вариантов использования рекомендуется применять "сильные" глаголы[2]. Если варианты использования и содержащиеся в них требования являются глаголами, то требования к качеству являются наречиями, то есть они уточняют варианты использования или отдельные требования относящиеся к данным вариантам использования. Они отвечают на вопросы "как много", "как быстро" и "до какой степени". Такие требования ранее назывались нефункциональными требованиями, но сегодня они чаще называются требованиями к качеству. Каким образом определяются требования к качеству?

Требования к качеству являются прежде всего требованиями, которые могут быть привязаны к вариантам использования или другим требованиям. Я использую стереотип «квалифицирует» для отношений зависимости между элементами модели и требованиями к качеству. В данном задании вам предлагается взять вариант использования "Управление полетом БЛА" и на отдельной диаграмме использования добавить все требования к качеству, которые влияют на данный вариант использования, привязав их отношениями зависимости[3].

Следует отметить, что хотя Rhapsody предоставляет специальный (взятый из SysML) элемент, достаточно часто для представления требований также используются элементы-ограничения. Такие ограничения (или элементы-требования) могут быть отображены на диаграммах использования, а также на диаграммах детализирующих варианты использования: диаграммах последовательности, временных диаграммах и диаграммах состояний, о чем будет рассказано более подробно далее в этой главе.

---

[2] Согласно существующим соглашениям по именованию, для имен вариантов использования должны использоваться сильные глаголы, хотя иногда для реактивных систем основанных на состояниях, вы можете встретить существительное "…режим" в названии варианта использования.

[3] Мне часто задают вопрос, нужно ли создавать отдельный пакет, содержащий требования к качеству. Я всегда отвечаю "Разумеется, нет." Если требование к качеству применяется к варианту использования или требованию, связанному с вариантом использования, то это наречие должно быть помещено следом за глаголом, на который оно влияет, что означает, что данное требование должно быть размещено в том же пакете, что и вариант использования, к которому оно относится. Если это параметрическое требование, то требование к качеству должно быть помещено в тот же пакет с параметрическим требованием, которое оно характеризует.

Задание 3.7: Представление взаимодействия: Определение сценариев для дорожного перекрестка

До сих пор мы занимались определением требований и идентификацией вариантов использования, с которыми они связаны. Все что мы делали является лишь небольшой модернизацией традиционных подходов по работе с требованиями. Это *может* быть полезно, но при большом количестве требований остаются все традиционные проблемы: требования трудны для понимания, неоднозначны и практически не могут быть проверены. Могут ли широкие возможности UML для моделирования помочь нам справиться с этими трудностями?

Ответ однозначный: ДА (могу поспорить, что вы догадались, что я отвечу именно так ;-)). Мы называем данный подход "детализацией вариантов использования". Для начала мы рассмотрим применение сценариев как средства определения операционных требований для выбранного варианта использования. В следующих заданиях мы рассмотрим как для этих целей могут использоваться формальные языки (конечные автоматы и диаграммы деятельности).

Сценарий можно рассматривать как определенную последовательность входных и выходных воздействий, которая описывает один реализовавшийся случай взаимодействия в выбранном варианте использования. Для одного варианта использования, как правило, могут быть определены много различных сценариев (их число может достигать нескольких десятков), которые отображают различные комбинации входных и выходных воздействий и различный порядок их следования. Определение сценариев очень полезно, так как сценарии связывают функциональность системы с тем, как система будет взаимодействовать с внешним окружением (т.е. действующими лицами). Эксперты в предметной области легко воспринимают сценарии и легко могут отличать правильные от неправильных, в тоже время они теряются, когда сталкиваются с формальными языками. И, наконец, сценарии полезны на любом уровне абстракции. На уровне спецификации системы сама система (или вариант использования системы) отображается в виде одной линии жизни – остальные линии жизни представляют действующих лиц. На уровне спецификации подсистем линия жизни системы "раскрывается", и внутренние части системы (подсистемы) также отображаются как линии жизни. На уровне спецификации кооперации объектов линии жизни подсистем декомпозируются на элементарные (неделимые) объекты, которые взаимодействуют между собой, полностью реализуя требуемую функциональность.

На сегодняшний момент для представления сценариев наиболее часто используются диаграммы последовательности. Иногда также используются временные диаграммы и диаграммы коммуникации (ранее они назывались "диаграммами кооперации"), но обычно для определения сценариев на этапе проектирования.

Стандартная проблема для использования сценариев, заключается в том, что существует *очень много возможных сценариев*, так что люди увязают в них, как в

болоте -- особенно в сценариях "дождливого дня". Это сценарии, в которых описывается возникновение различных внештатных ситуаций, которые система должна некоторым образом обработать. Я рекомендую *вначале* определять сценарии "солнечного дня", а уже затем переходить к сценариям "дождливого дня".

Сколько должно быть сценариев? Минимальный набор сценариев для варианта использования должен отражать все операционные требования и требования к качеству операций, определенных для данного варианта использования. Каждое требование должно быть отражено хотя бы в одном сценарии. Что касается внештатных ситуаций, то обычно их лучше всего разделить на *классы эквивалентности*. Каждый класс эквивалентности представляет собой множество возможных отказов системы, которые идентифицируются и обрабатываются одинаковым образом. Так, в аппарате для анестезии отказ, возникающий когда эндотрахеальная трубка выскакивает из горла пациента, относится к тому же классу эквивалентности, что и для случая, когда перекрывается или отключается подача кислорода. В обоих этих случаях, внештатная ситуация обнаруживается одним способом (например, по отсутствию изменения давления в конце выдоха), и система одинаково реагирует на такую ситуацию (один и тот же сигнал тревоги для дежурного врача). Таким образом, необходимо определить только один сценарий "не удалось наполнить газовой смесью легкие пациента". Я рекомендую использовать элементы-ограничения для перечисления всех специфических внештатных ситуаций, которые соответствуют некоторой общей внештатной ситуации. Я не считаю, что необходимо или полезно создавать много одинаковых сценариев, отличающихся только спецификой внештатной ситуации.

Для данного задания возьмите один вариант использования, определенный для системы управления дорожным перекрестком, и определите для него три различных сценария. Будьте аккуратны при определении действующих лиц – это должны быть объекты, находящиеся за границами системы, которые взаимодействуют с системой во время реализации варианта использования, причем эти взаимодействия должны приводить к значимым результатам. Для каждого из сценариев также определите предусловия. Мне нравится делать это, добавляя элементы замечаний или ограничения в верхней части диаграммы последовательности.

Задание 3.8: Представление взаимодействия: Сценарии наблюдения в оптическом диапазоне для системы БЛА Койот

Система БЛА Койот намного превосходит по размерам систему управления дорожным перекрестком Роадраннер. Следовательно, для нее существует большее количество вариантов использования, а также более глубокая их вложенность. Несмотря на это, для нее может использоваться тот же самый подход для описания операционных аспектов системы. В данном случае еще более важно правильно сгруппировать и точно определить назначение каждого сценария, поскольку будет очень много вариантов использования и каждый из них будет включать много сценариев. В данном задании возьмите вариант использования ”Оптическое наблюдение” и определите для

него три сценария. В первом сценарии оператор при помощи джойстика управляет видеокамерой для наблюдения за местностью над которой пролетает БЛА Койот. Во втором сценарии оператор визуально обнаруживает цель, задает цель на изображении и дает команду на автоматическое сопровождение цели камерой при облете БЛА над близлежащей территорией. В третьем сценарии оператор использует ручное управление камерой для наблюдения за территорией и, после обнаружения вероятной цели, выполняет ее трехкратное увеличение, после чего просматривает близлежащую территорию, используя ручное управление камерой.

Задание 3.9: Представление спецификации: Описание варианта использования

Текстовое представление является очень удобным средством для фиксации требований, поскольку текст обладает мощными выразительными возможностями и хорошо воспринимается другими людьми. Однако оно не обладает достаточной точностью и может быть неоднозначным, создавая кучу проблем, если данное представление является единственным при описании требований. В предыдущих заданиях мы использовали элементы модели “требование” для организации требований в связанные группы. Одновременно полезно предоставлять описание вариантов использования в виде "структурированного текста". В инструментальных средствах, таких как Rhapsody, обычно это делается в поле описания для варианта использования. Я использую следующий формат для описания вариантов использования:

> **Имя:** *имя варианта использования*
> **Владелец:** *отдельное лицо или группа лиц, ответственных за вариант использования*
> **Назначение:** *назначение варианта использования для пользователей системы (какую ценность он несет и какому назначению служит)*
> **Требования (опционально):** *список требований (не обязателен, если используются диаграммы требований)*
> **Данные (опционально):** *данные, получаемые, обрабатываемые или выдаваемые в данном варианте использования – их не обязательно явно описывать здесь, если на диаграммах использования отображаются информационные потоки.*
> **Предусловия:** *что должно иметь место до начала варианта использования*
> **Постусловия:** *что должно иметь место после завершения варианта использования*
> **Ссылки на используемые документы:** *ссылки на используемые стандарты и спецификации*

Разные авторы предлагают различные форматы для описания вариантов использования. Например, некоторые предлагают приводить список действующих лиц или ограничений. Я советую не дублировать в данном описании информацию, уже

представленную (или которую еще предстоит собрать) графически. Поэтому я не включаю в описание ни действующих лиц, ни привожу список основных сценариев.

Цель данного упражнения – воспользоваться приведенным выше или любым другим форматом, который вы выберете, для описания одного варианта использования системы управления дорожным перекрестком Роадраннер (Детектирование ТС) и одного варианта использования для БЛА Койот.

Представление спецификации: Конечные автоматы для сбора требований

Многие системы обладают "реактивным поведением", которое заключается в том, что система ожидает наступления некоторых интересующих ее событий, а затем реагирует на них, выполняя некоторые наборы действий и изменяя свое состояние; после чего ожидает наступления следующего события. Для описания такого поведения идеально подходят конечные автоматы. Мы будем использовать конечные автоматы для описания требований к реактивным системам. Мы можем (и с большой вероятностью будем) использовать конечные автоматы при проектировании и реализации этих же систем, однако в данном разделе нас интересует качественная спецификация требований с использованием конечных автоматов, а не проектирование или реализация.

Многие инженеры бывают удивлены использованию конечных автоматов для спецификации требований, в тоже время конечные автоматы используются для спецификаций требований к сложным реактивным системам уже более 30 лет. Продукт IBM Rational Statemate™ (изначально принадлежащий компании I-Logix) – пример инструмента, который был изначально разработан (в середине 1980-х гг.) для спецификации требований к аэрокосмической системе военного назначения. Он до сих пор используется для данных целей. В Statemate к системе применяется функциональная декомпозиция, а сами функции описываются при помощи диаграмм состояний, таблиц истинности, а также определения действий. Statemate в основном используется системными инженерами для спецификации требований, а *не* разработчиками программного обеспечения или аппаратуры, осуществляющими проектирование данных систем. UML также подходит и активно используется во многих проектах для целей спецификации требований в ясном и недвусмысленном виде.

Использование конечных автоматов имеет множество преимуществ перед спецификацией требований в текстовом виде. Хотя текст является очень гибким инструментом для описания, позволяющим фиксировать очень тонкие моменты, преимущество конечных автоматов заключается в их точности и недвусмысленности. Благодаря этим качествам – точности и недвусмысленности – конечные автоматы являются мощным средством для спецификации требований. Кроме того, конечные автоматы являются *исполняемыми*, свойство которое незаменимо для проверки

согласованности, правильности и точности большого набора требований. Конечные автоматы *можно протестировать*, что позволяет оценить требования еще до того, как система будет создана. Более того, на основе таких формализованных требований можно автоматически сгенерировать тестовые сценарии и использовать их в дальнейшем для приемочного тестирования конечной системы. Текстовое представление для описания требований также полезно, так как конечные автоматы не позволяют удобным и очевидным образом ответить на вопрос "Почему?" – но конечные автоматы гораздо более полезнее текста для спецификации требований.

Так как варианты использования группируют требования в связанные группы, мы будем определять конечные автоматы для каждого варианта использования по отдельности. Если система в некотором варианте использования в основном ожидает входных событий и затем реагирует на них, то для такого варианта использования подходит спецификация на основе конечного автомата. Данные события могут быть синхронными, асинхронными или связанными со временем. Независимо от их типа, если система ожидает наступления некоторых событий и выполняет определенные действия в ответ, конечный автомат может быть легко использован для спецификации требуемого поведения в данном варианте использования. Для вариантов использования, в которых последовательно выполняются некоторые действия, т.е. выполняются алгоритмы, более подходящим способом описания являются диаграммы деятельности. В следующем разделе мы покажем как диаграммы деятельности могут использоваться для спецификации такого поведения.

Можно определить несколько простых рекомендаций, которым стоит придерживаться при моделировании конечных автоматов для вариантов использования:

- Входящие сообщения или команды соответствуют событиям для конечных автоматов (возможно, с добавлением данных в качестве параметров).
- Исходящие сообщения или команды становятся действиями в конечном автомате.
- Конечный автомат не должен содержать никакой информации об используемых технологиях или деталях реализации системы, поскольку целью создания конечного автомата для варианта использования является спецификация требуемого поведения, а не внутреннего устройства системы.
- Сценарии для варианта использования представляют собой не что иное, как различные "пути" внутри конечного автомата.
    - Каждый сценарий для варианта использования должен определять некий путь внутри конечного автомата для этого варианта использования.
    - Для обеспечения минимального покрытия, каждый переход конечного автомата должен быть представлен как минимум в одном сценарии.
    - Как уже упоминалось ранее, сценарии для вариантов использования являются основой для приемочного тестирования конечной системы.

Сценарии являются очень полезными операционными представлениями для описания поведения системы. Однако, сценарии являются лишь частичным описанием, вернее они содержат только некоторую часть описания. Как правило, для полного описания

поведения системы в рамках одного варианта использования требуется набор (обычно большой) сценариев. Конечный автомат, напротив, является полным описанием, которое содержит полную спецификацию поведения в варианте использования. При необходимости, конечные автоматы могут быть декомпозированы и представлены стандартным образом на нескольких диаграммах; но вне зависимости от того, представлены ли вложенные состояния на той же диаграмме или вынесены на другие, данное описание все равно является одним представлением для поведения системы.

Задание 3.10: Представление спецификации: Сбор сложных требований

Диаграммы состояний являются отличным средством для спецификации поведения системы в некотором варианте использования. В данном упражнении вам предлагается создать две диаграммы состояний. Первую для варианта использования "Вечерний режим с низкой плотностью потока". Проверяйте себя, что вы фиксируете следующие требования к поведению системы - на главной дороге должен мигать желтый, а на второстепенной должен мигать красный. Частота мигания должна составлять 0,5 Гц, а время горения должно составлять 75% от периода мигания. Также добавьте дополнительную временную задержку перед переходом в данный режим, чтобы транспортный поток через перекресток мог завершиться прежде чем система переключится в новый режим.

Вторая диаграмма состояний значительно сложнее первой. Вам предлагается создать диаграмму состояний для варианта использования "Режим с фиксированной длительностью цикла". Я рекомендую сделать это в три приема[4]. Сначала опишите поведение без учета поворотных полос и пешеходов. После этого дабавьте поворотные полосы. Помните, что существует два режима функционирования поворотных полос, первый – при котором сквозной транспортный поток через перекресток ожидает завершения поворота (ОДН - одновременный режим), и второй – при котором зеленый сигнал на поворот включается одновременно с зеленым сигналом для сквозного потока через перекресток (ПОС - последовательный режим). Усложняющим фактором является то, что вы должны быть способны продетектировать возникновение события о наличии транспорта на поворотной полосе и помнить об этом до тех пор пока не придет время обработать его и только после этого вы можете об нем забыть. Используйте паттерн проектирования “защелкивающее состояние” [2]: для каждой поворотной полосы создайте И-состояние с двумя вложенными состояниями: ”ГлПовНетТС” и ”ГлПовЖдетТС”. При получении запроса от поворачивающего транспортного средства, находящегося на данной поворотной полосе, должен выполняться переход в состояние ”ГлПовЖдетТС”. После того, как запрос будет

[4] Такое разбиение задачи на более мелкие составляющие и проверка каждой из них посредством исполнения является частью "наноцикла" в процессе Harmony (ранее известного как ROPES). Этот подход *очень сильно* упрощает задачу получения корректно работающего конечного продукта.

выполнен (например, сигнал на поворот переключится в красный), должен выполняться переход обратно в состояние ”ГлПовНетТС”.

После того как с обработкой поворотных полос будет закончено, добавьте обработку пешеходов.

Подсказка: Не пытайтесь с самого начала создать оптимальную диаграмму состояний, минимизируя число возможных состояний. После того как вы убедитесь, что модель правильно описывает поведение вашей системы, вы при необходимости сможете оптимизировать ее в дальнейшем. Ваша текущая цель – это правильность и простота, даже если в конечном автомате будет иметь место избыточность. Это никак не отразится на проектирование системы, поскольку на данном этапе мы лишь специфицируем требования к системе. Поэтому, если вы еще не обладаете достаточным опытом в создании диаграмм состояний, создайте два или-состояния, одно – для одновременного режима (ОДН), а второе – для последовательного режима (ПОС). Затем для каждого из них подробно опишите поведение сквозного потока через перекресток. После этого добавьте управление пешеходным потоком, используя паттерн проектирования “защелкивающее состояние”.

Задание 3.11: От операционного представления к представлению спецификации: Определение операционных контрактов

В этом последнем упражнении по анализу требований мы возьмем один из вариантов использования БЛА Койот (Оптическое наблюдение) и пройдемся по способу сбора требований, используемому в процессе Harmony®. Этот подход хорошо зарекомендовал себя в проектах по разработке больших систем, в которых выделены отдельные команды системных инженеров (или по крайней мере, где системная инженерия выделена как отдельный вид работ), с последующей спецификацией архитектуры системы и далее распределением работ между различными инженерными дисциплинами – разрабатывающими электронику, программное обеспечение, механические, а также химические компоненты. В проектах по разработке только программного обеспечения, в особенности тогда, когда важны гибкость и способность адаптировать к изменяющимся целям, может использоваться более простой подход. Тем не менее, в данном примере мы с вами предположим, что вначале группа системных инженеров выполняет подробную спецификацию требований и архитектуры системы, а затем на уровне подсистем декомпозирует подсистемы на электронные, программные, механические и химические компоненты, назначая отдельные требования для каждого из них.

Сбор требований в процессе Harmony

Полный процесс Harmony представлен на Рис. 3-2. Это разновидность "V-процесса", в котором сначала производится анализ требований и определяется архитектура системы, за которыми следует спираль, включающая в себя фазы анализа, проектирования, реализации и тестирования (о спирали подробнее будет рассказано в следующих главах).

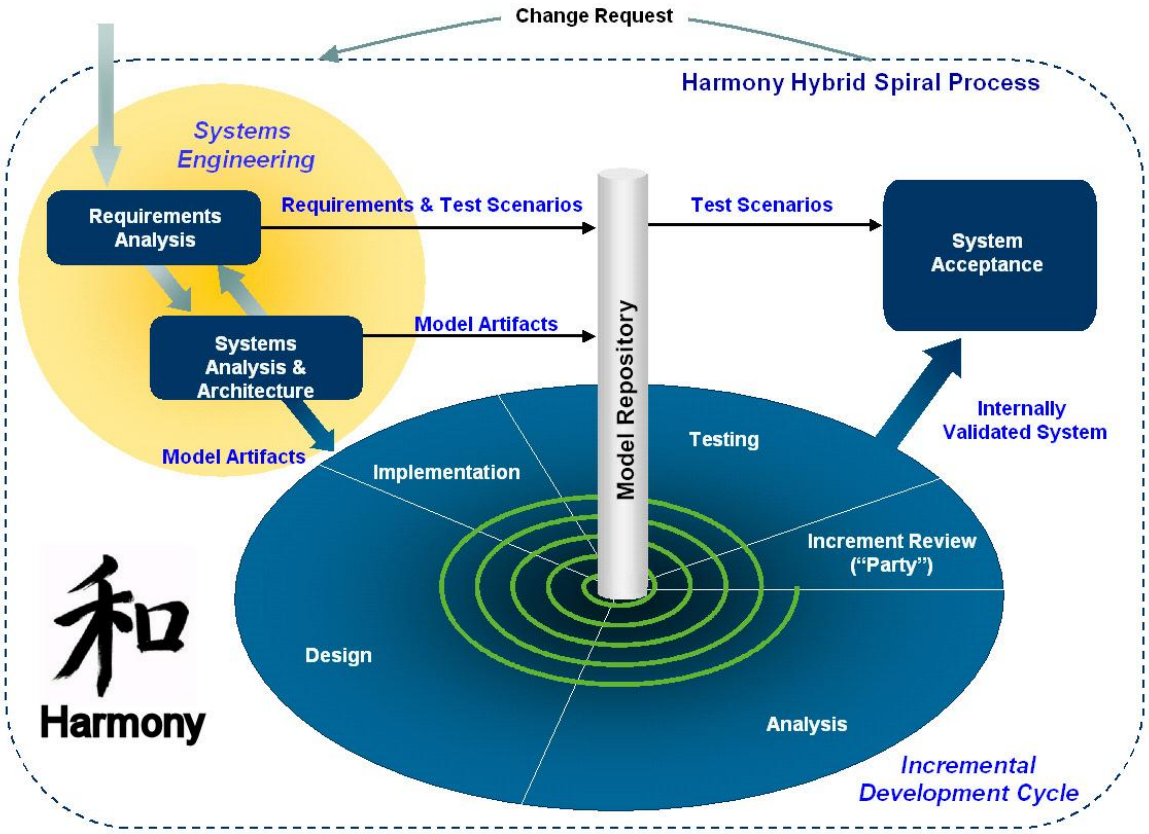

Рис. 3-2: Обзор процесса Harmony

В этой главе, посвященной анализу требований, мы сосредоточимся главным образом на этапах сбора и спецификации требований, относящихся к той части процесса Harmony, в которой выполняется системная инженерия. Последовательность работ, выполняемых при системной инженерии, показан на **Рис. 3-3**

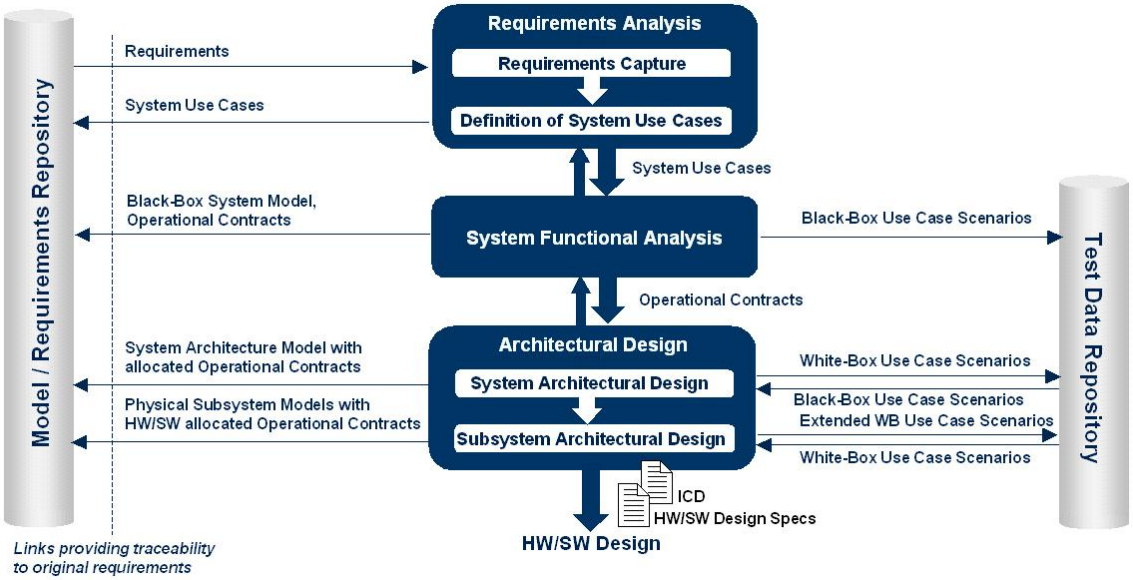

Рис. 3-3: Поток работ в процессе Harmony по разработке системы

Для БЛА Койот мы уже определили множество вариантов использования уровня системы. В этом задании мы проведем анализ одного из этих вариантов использования для системы, выступающей как в качестве "черного" так и “белого” ящика. На этапе функционального анализа системы как "черного ящика" мы определим операционные контракты, которые имеют место при взаимодействии с системой в рамках выбранного варианта использования. Операционные контракты являются наборами сервисов, предоставляемых или запрашиваемых системой, которые используются при взаимодействии системы с ее окружением в процессе работы. К ним относятся отправляемые сообщения, данные, связанные с сообщениями, а также пред- и постусловия для этих сообщений. На следующем шаге мы определим подсистемы для БЛА Койот; а затем, при анализе системы как "белого ящика" мы привяжем операционные контракты к этим подсистемам.

Замечание по нотации

Можно определять конечные автоматы непосредственно для элементов модели “вариант использования”, и именно так мы сделали в этой главе. Однако, на данном этапе уже полезно начинать определять интерфейсы и это то, что невозможно отобразить явным образом на диаграмме использования. По этой причине мы будем моделировать вариант использования с использованием класса. Будучи представленным с помощью класса, я смогу для него добавить порты и точно специфицировать интерфейсы (включая сервисы и данные), которые определяют операционные контракты, предоставляемые портами. Это пригодится нам и в дальнейшем, когда мы определим внутреннюю структуру для кооперации элементов,

реализующей данный вариант использования – мы сможем определить экземпляры объектов в виде составных частей класса, моделирующего вариант использования. Чтобы явно показать, что данный класс представляет собой вариант использования, я буду использовать для его имени префикс "ВИ". Поскольку в UML отсутствует нотация для отображения экземпляров действующих лиц, а также для отображения интерфейсов, которые они могу запрашивать или предоставлять (что необходимо при разработке системы), я также буду моделировать действующих лиц с помощью отдельных классов. В именах классов, моделирующих действующих лиц, я буду использовать префикс "ДЛ". Например, на Рис. 3-4 показана обычная диаграмма использования, представленная в виде классов на Рис. 3-5. Последний рисунок предоставляет больше информации относительно точек соединения и их интерфейсов, чем первый.

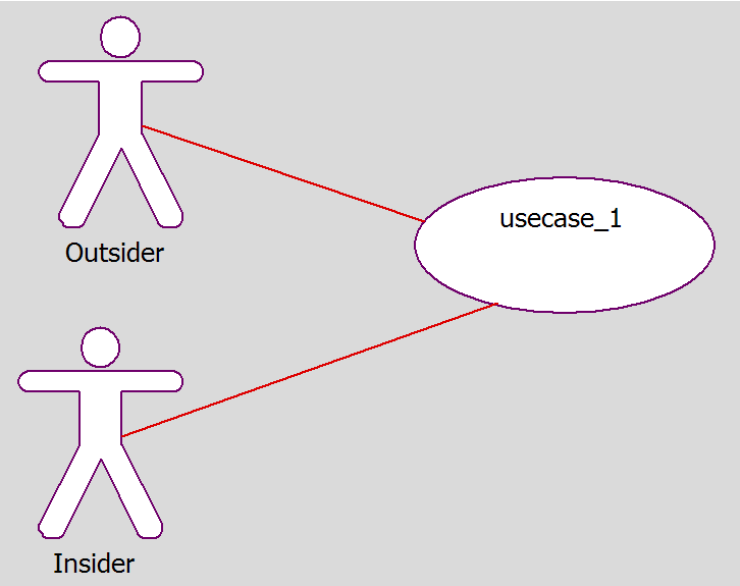

Рис. 3-4: Диаграмма использования (Стандартная)

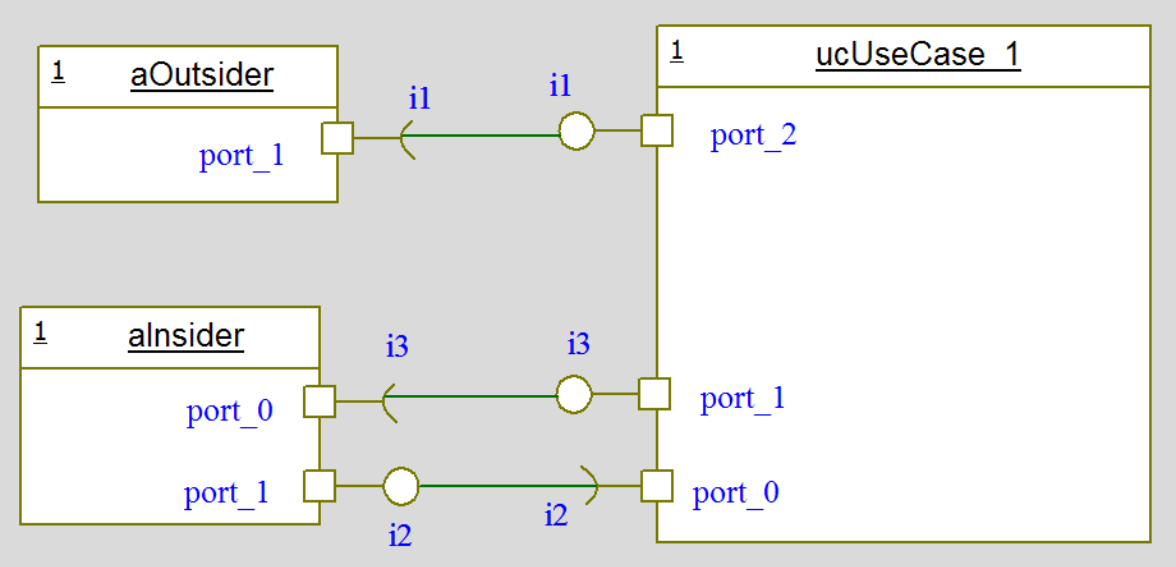

Рис. 3-5: Диаграмма использования (В нотации классов)

Среда разработки Rhapsody предоставляет возможность использования элемента *блока*, который является объектом с единичной множественностью, для которого не определен тип с помощью класса. Блоки могут быть легко преобразованы в обычные объекты, однако многие системные инженеры не хотят отвлекаться от своих задач спецификации требований, уделяя много внимания семантике объектов и классов. При желании, мы можем использовать термин объект или объектная роль всякий раз, когда мы используем термин "блок".

От полетных требований БЛА Койот к операционным контрактам

Ниже представлен базовый подход по спецификации операционных контрактов:

Шаг	Задача	Результирующий продукт	Комментарий
1	Определить границы модели варианта использования	Диаграмма использования Диаграмма структуры для каждого варианта использования	Порты определены, но на данном шаге еще не определены их интерфейсы.
2	Определить операционные контракты уровня системы, используя сценарии варианта использования для системы в представлении "черного ящика"	Диаграммы последовательности для системы в представлении "черного ящика"; Операционные контракты уровня системы	Линии жизни могут быть действующими лицами или блоками вариантов использования
3	Определить последовательность выполнения операций в варианте	Диаграмма деятельности для	Эта задача может выполняться

	использования	варианта использования системы в представлении "черного ящика"	одновременно с задачей №2.
4	Реализовать сообщения в виде операций	Наполненная диаграмма структуры для системы в представлении "черного ящика"	Лучше не добавлять сообщения к модели до тех пор, пока диаграмма последовательности не стабилизируются, после чего наполнить ими модель при помощи "автоматической реализации".
5	Определить интерфейсы; Распределить сообщения по интерфейсам		
6	Определить поведение уровня системы , основанное на состояниях	Диаграмма состояний для варианта использования (для каждого отдельная)	Получается на основе сценариев варианта использования для представления "черного ящика"
7	Выполнить проверку и оценку модели варианта использования в представлении "черного ящика" путем исполнения модели	Проверенная модель вариантов использования уровня системы	

Таблица 3-1: Определение операционных контрактов в процессе Harmony

Для варианта использования БЛА Койот "Оптическое наблюдение " мы уже определили несколько сценариев. Обратите внимание, что сообщения от действующих лиц становятся операционными контрактами для портов, а сообщения самому себе, называемые "рефлексивными сообщениями" – будут использоваться для функциональной декомпозиции требуемой обработки, выполняемой в ответ на задействование операционного контракта.

В этом задании вам предстоит выполнить следующее:

1. Создать диаграмму, на которой с помощью блоков отобразите варианты использования, действующих лиц и определите интерфейсы между ними.
2. Сообщения от действующих лиц необходимо поместить в *предоставляемые* интерфейсы для соответствующего порта в блоке варианта использования и в *используемые* интерфейсы для порта в блоке действующего лица. Сообщения, посылаемые действующему лицу, необходимо поместить в *используемые* интерфейсы для порта в блоке варианта использования и в *предоставляемые* интерфейсы для порта в блоке действующего лица. Сообщения для блока варианта использования должны быть открытыми (видимыми всем), а рефлексивные сообщения – защищенными или закрытыми.
3. Для каждого сервиса (операции), предоставляемых системой, определите пред- и постусловия, а также типы и ограничения на возможные значения параметров, если это необходимо.

4. Создайте диаграмму деятельности для варианта использования, которая будет содержать *все* сценарии, определенные для данного варианта использования ранее.
5. Определите конечный автомат для варианта использования, который будет соответствовать данному набору сценариев.

Несомненно, последний пункт в данном списке будет самый сложный.

Ссылки

[1] Douglass, Bruce Powel Real-Time UML 3rd Edition: Advances in the UML for Real-Time Systems (Addison-Wesley, 2004)

[2] Douglass, Bruce Powel Doing Hard time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns (Addison-Wesley, 1999)