Лучшие практики разработки программного обеспечения в соответствии с DO-178 с использованием модельно-ориентированного проектирования

Raymond G. Estrada, Jr.
The MathWorks, Torrance, CA

Eric Dillaber
The MathWorks, Natick, MA

Gen Sasaki
The MathWorks, Torrance, CA

Модельно-ориентированное проектирование и автоматическая генерация кода являются отработанным подходом при разработке встраиваемых систем управления. Сейчас этот подход широко применяется в приложениях, которые должны удовлетворять DO-178B, стандарту программного обеспечения в коммерческой авиации. Модельно-ориентированное проектирование позволяет инженерам создавать продвинутые встраиваемые программные системы, используя исполняемую модель как основу для проектирования, симуляции, верификации и валидации программного обеспечения. Сложность современных авиакосмических систем привела к увеличению использования модельно-ориентированного проектирования для всего жизненного цикла процессов разработки системы — от ранней верификации до реализации и системной интеграции. Для максимизации преимуществ модельно-ориентированного проектирования в контексте удовлетворения целей DO-178B (и DO-178C после принятия его FAA) нужен уровень экспертизы, для достижения которого требуются годы практического опыта.

I. Введение

В этой статье мы делимся набором лучших практик, которые являются критически важными для достижения успеха в организациях, работающих над проектами, требующими сертификации по DO-178B и DO-178C с использованием модельно-ориентированного проектирования. Эти лучшие практики описывают ключевые факторы, методы и фундаментальные возможности модельно-ориентированного проектирования, охватывающие процесс разработки программного обеспечения от моделирования и симуляции до генерации кода, верификации и валидации, и реализации. Эти лучшие практики являются следствием многолетнего взаимодействия с заказчиками в авиакосмической промышленности, разрабатывающими программное обеспечение повышенной надежности. Наше намерение заключается в том, чтобы помочь организациям избежать распространенных подводных камней и сократить время, усилия и стоимость разработки программного обеспечения повышенной надежности, удовлетворяющего целям DO-178.

DO-178B и недавно выпущенный DO-178C содержат четкие цели для процессов жизненного цикла программного обеспечения — таких, как разработка требований к программному обеспечению, проектирование программного обеспечения, кодирование, интеграция, верификация и управление конфигурацией. В таблицах от A-1 до A-10 Приложения A стандарта перечисляются выходные документы, требуемые для каждой цели. Эти выходные документы

генерируются на основе уровня программного обеспечения, необходимого для безопасности и требуют верификации перед использованием для сертификации. Эти таблицы и документ DO-178B [7], однако, предоставляют мало руководств по использованию моделей, предоставляя возможность авиакосмическим компаниям самим решать, как лучше применять модельно-ориентированное проектирование для разработки программного обеспечения в соответствии с DO-178. Выпуск DO-178C принес дополнение, RTCA DO-331 «Model-Based Development and Verification Supplement to DO-178C and DO-278A» [6] («Модельно-ориентированное проектирование и верификация. Дополнение к DO-178C и DO-278A»), содержащее изменения и добавления к целям, мероприятиям и данным жизненного цикла программного обеспечения, которые должны обеспечиваться при использовании модельно-ориентированного проектирования. DO-331 предоставляет дополнительные руководства по артефактам, генерируемым при помощи моделей, и соответствующим доказательствам верификации. Несмотря на ценность этих руководств, командам инженеров требуется эффективный рабочий процесс для ранней идентификации ошибок, быстрой генерации выходных документов для целей стандарта, устранения ручных процессов с использованием инструментов, которые могут быть квалифицированы, и сокращения количества программных ошибок во время интеграции системы. Следующие лучшие практики могут быть приняты для внедрения рабочего процесса разработки программного обеспечения, которое удовлетворяет целям DO-178 с использованием модельно-ориентированного проектирования:

- установление стандартизированного окружения модельно-ориентированного проектирования для разработки и верификации;
- определение стратегии управления конфигурацией, включающей артефакты модельно-ориентированного проектирования;
- определение оптимального процесса модельно-ориентированного проектирования для DO-178;
- выявление того, представляют ли модели требования высокого или низкого уровня;
- использование симуляции и анализа для поддержки целей DO-178;
- анализ на более высоком уровне абстракции результатов тестирования относительно требований;
- использование инструментов модельно-ориентированного проектирования, которые могут быть квалифицированы.

II. Лучшие практики

A. Установление стандартизированного окружения модельно-ориентированного проектирования для разработки и верификации

DO-178 требует стандартизированного окружения для разработчиков, которое позволяет им воспроизвести определенную версию программного обеспечения [5]. Несоответствующее окружение для разработки программного обеспечения может производить исходный код и объектный код, которые не полностью соответствуют требованиям. Установление стандартизированного окружения модельно-ориентированного проектирования помогает удостовериться в том, что модели проекта могут использоваться для генерации согласованных, повторяемых результатов симуляции и тестирования, а также исходного и объектного кода, которые трассируются к требованиям.

Несмотря на то, что большинство организаций с опытом использования традиционных методов уже имеют установленные окружения для инструментов, первоначальное установление окружения модельно-ориентированного проектирования может быть сложным. Причина заключается в обширном наборе инструментов и настроек, доступных для поддержки дополнительных возможностей, предоставляемых модельно-ориентированным проектированием. Инструменты больше не ограничены определенными точками в процессе разработки программного обеспечения, а наоборот, используются непрерывным образом. В результате, содержимое окружения модельно-ориентированного проектирования должно быть хорошо продумано для наиболее эффективной поддержки проекта и сокращения технического обслуживания.

Так же, как на выходы компилятора и компоновщика влияют отдельные конфигурационные файлы, на выходы симуляции и генерации кода влияют такие факторы, как опции конфигурации, файлы настроек и другие зависимости (рис. 1). Эти зависимости часто хранятся в нескольких форматах и обычно находятся на нескольких уровнях иерархии в окружении модельно-ориентированного проектирования.

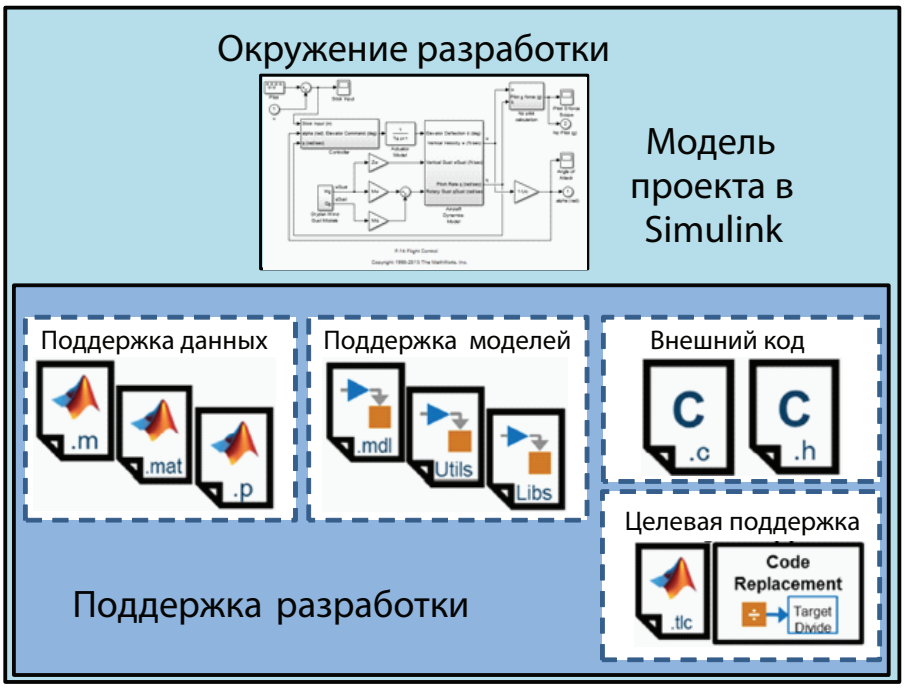

Рис. 1. Окружение модельно-ориентированного проектирования для разработки программного обеспечения

Лучшая практика состоит в автоматизации установки и настройки окружения разработки для всех разработчиков. Для новых пользователей или свежей установки окружения разработки может использоваться командный скрипт для автоматической установки:

1. Корректных версий инструментов разработки
2. Корректной версии компилятора
3. Библиотек для разработки моделей
4. Файлов с данными и скриптов

В дополнение, инициализационные скрипты, которые запускаются автоматически при старте, могут помочь в обеспечении согласованного окружения разработки путем автоматического выполнения таких задач, как:

1. Верификация корректных версий инструментов, библиотек, "заплат", и тому подобного, как указано в плане разработки программного обеспечения
2. Добавление путей ко всем зависимостям модели проекта
3. Выполнение дополнительных скриптов для настройки и поддержки
4. Определение и загрузка данных

Примером этого является ярлык на рабочем столе Windows, выполняющий главный инициализационный скрипт, который настраивает окружение разработки. Автоматическая установка, настройка и конфигурация помогают установить корректный набор используемых зависимостей для модели проекта.

Стандартизированное окружение разработки включает общую, разделяемую библиотеку, которая используется для разработки моделей проекта. Примитивные блоки (например, блоки умножения или сложения) часто имеют несколько опций, доступных разработчикам, которые могут привести к несогласованному использованию опций блока в моделях проекта. Например, блок Gain (умножение на коэффициент) в Simulink имеет несколько доступных опций — таких, как частота дискретизации, атрибуты сигнала и атрибуты параметра. Должна быть создана библиотека, содержащая подмножество доступных примитивов, которая будет использоваться совместно. Настраиваемые проверки Model Advisor могут использоваться для верификации того, что корректные настройки блока используются согласованно во всей модели проекта.

B. Определение стратегии управления конфигурацией, включающей артефакты модельно-ориентированного проектирования

Само по себе стандартизированное окружение модельно-ориентированного проектирования не является достаточным для воспроизведения (тиражирования) согласованных данных жизненного цикла программного обеспечения. Типичные зависимости модели включают в себя поддерживающие библиотеки, файлы данных, скрипты и другое. Эти зависимости требуются для симуляции, тестирования, верификации и генерации кода. DO-178 говорит, что данные жизненного цикла программного обеспечения должны быть помещены под управление конфигурацией (УКПО) [5], как описано в плане сертификации программного обеспечения («План сертификации ПО»). Установление стратегии УКПО, которая поддерживает модельно-ориентированное проектирование и является согласованной с «Планом сертификации ПО» наряду со стандартизированным окружением разработки, позволяют получать согласованные, воспроизводимые результаты симуляции и тестирования, а также исходный и объектный код.

«Золотое правило» процесса управления конфигурацией артефактов модельно-ориентированного проектирования заключается в том, что исходная модель — это источник информации в проекте. Параметры, используемые в модели, также могут считаться частью модели в этом контексте, но должны управляться с использованием отдельных файлов данных или скриптов. Производные артефакты — такие, как C-код и исполняемые файлы, должны всегда генерироваться из исходной модели. Показательные типы артефактов модельно-ориентированного проектирования приведены на рис. 2.

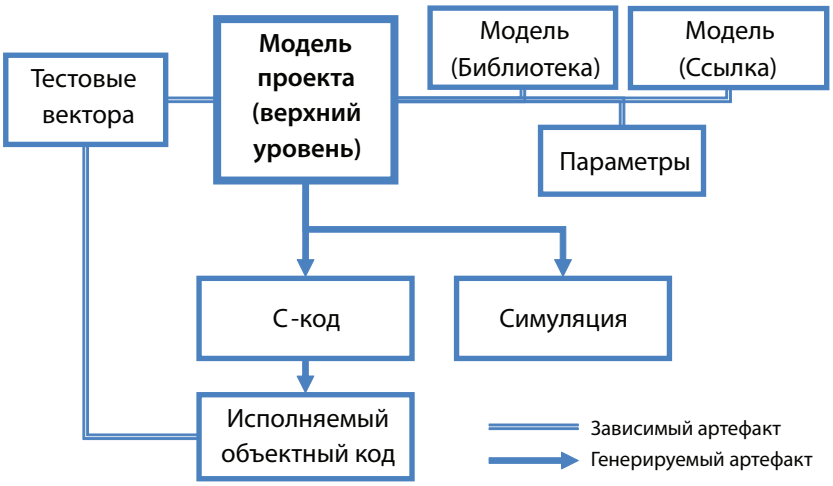

Рис. 2. Показательные артефакты модельно-ориентированного проектирования

С учетом DO-178, в процессе УКПО особенно важно принять во внимание следующее:

- артефакты разработки, хранящиеся в системе УКПО, должны быть синхронизированы;
- артефакты разработки должны быть проверены относительно стандартов до передачи их в систему УКПО.

Следующие параграфы подчеркивают общие проблемы, с которыми сталкиваются инженеры при определении процесса УКПО и некоторые лучшие практики, адресующие эти проблемы.

Синхронизация артефактов

В модельно-ориентированном проектировании многие артефакты, такие, как С-код и исполняемые файлы, могут автоматически генерироваться из модели и связанных с ней параметров. В процессе, который обязан соответствовать DO-178, эти артефакты должны архивироваться, чтобы иметь возможность обратиться к ним в любое время. Полезно идентифицировать изменения в производных артефактах по мере того, как исходные артефакты, из которых они генерируются, обновляются в итерационном процессе разработки.

Для обеспечения согласованности и повторяемости этого действия требуется установить процесс для верификации того, что все артефакты синхронизованы с их исходными файлами до передачи в систему УКПО. Этот процесс должен, например, предотвращать передачу в систему УКПО модифицированных исходных файлов модельно-ориентированного проектирования — таких, как моделей и файлов данных, без соответствующего автоматически сгенерированного кода. В дополнение, этот процесс должен предотвращать передачу в систему УКПО модифицированных автоматически сгенерированных артефактов, таких, как исходные и заголовочные файлы С, с тем, чтобы модель проекта и её зависимости являлись единственным источником всех автоматически сгенерированных файлов. Если рассинхронизованные артефакты попадают в систему УКПО, это может затруднить или помешать отладке, поскольку инженеры не смогут воспроизвести предыдущие конфигурации. Кроме того, назначение каждого сгенерированного артефакта, хранящегося в системе УКПО, должно быть понятным и задокументированным. Например, требуется задокументировать, что отчет «Описание проекта ПО», сгенерированный из модели проекта и её зависимостей, используется для мероприятий по верификации модели и анализу трассируемости.

Проверка артефактов на стандарты до передачи их в систему УКПО

Хотя это не регламентируется DO-178, лучшей практикой является проверка артефактов разработки на соответствие установленным стандартам моделирования и кодирования до передачи их в систему УКПО. Примерами установленных стандартов моделирования в Simulink являются опубликованные NASA (Orion GN&C) и MathWorks Automotive Advisory Board (Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow). Для C-кода, стандарт MISRA C является возможным вариантом. В дополнение, требуется проверять, что модели симулируются и из них можно генерировать код, до передачи в систему УКПО.

Мероприятия по формальной верификации должны осуществляться на настроенной модели и её производных артефактах. Эта лучшая практика помогает и упрощает верификацию, обеспечивая инспекцию корректных артефактов. Например, модель, которая не соответствует стандартам и хранится в системе УКПО, может не симулироваться или не генерировать код. Это увеличивает длительность мероприятий по формальной верификации, потому что модель требуется обновить и повторно сгенерировать артефакты прежде, чем возобновить формальную верификацию. Применение инструментов модельно-ориентированного проектирования, таких, как Model Advisor, может помочь уменьшить или устранить такие задержки путем автоматической систематической проверки артефактов относительно стандартов до передачи в систему УКПО.

C. Определение оптимального процесса модельно-ориентированного проектирования для DO-178

Модели должны использоваться для нескольких целей в течение всего цикла разработки для максимизации эффективности процессов, увеличения возможностей по управлению сложностью и улучшения окупаемости инвестиций. Оптимальный подход к модельно-ориентированному проектированию включает симуляцию и анализ, генерацию кода, связь с требованиями и анализ трассируемости, автоматические рассмотрения кода, автоматическую генерацию документации, покрытие модели и формальные методы верификации.

Уровень требований к программному обеспечению, представленных моделями, должен быть определен до внедрения процесса разработки, специфического для проекта. Дополнительная информация и лучшие практики описаны в разделе «*Выявление того, представляют ли модели требования высокого или низкого уровня*». Один из подходов, используемых в авиакосмической промышленности, заключается в разработке моделей, производных от текстовых требований высокого уровня (ТВУ). Эти модели представляют собой требования низкого уровня (ТНУ), как описано в «MB Example 1» в таблице MB.1-1 (Model Usage Examples — Примеры использования моделей) из DO-331 [6] и являются моделями проекта (design models).

Процесс разработки программного обеспечения с использованием модельно-ориентированного проектирования идеально реализуется при установлении начального набора ТВУ к программному обеспечению. Следующие разделы описывают некоторые ключевые области, требующие больших усилий, в которых модельно-ориентированное проектирование дает преимущество относительно традиционных методов.

Ранняя верификация проекта

Разработка программного обеспечения с использованием модельно-ориентированного проектирования позволяет осуществлять раннюю верификацию проекта. Верификация на уровне модели помогает сократить число ошибок в требованиях и ошибок при разработке, которые находятся во время аппаратного и программного тестирования и даже позже. Симуляция моделей проекта на рабочих станциях используется для осуществления симуляции (не в

режиме реального времени) с использованием тестовых векторов, основанных на требованиях. Покрытие модели, требуемое для уровней программного обеспечения A, B и C в DO-178C, используется совместно с симуляцией для идентификации не протестированных требований, выраженных в моделях проекта.

Верификация согласованности ТНУ с ТВУ и соответствия стандартам обычно достигается путем ручного рассмотрения проекта программного обеспечения. С использованием модельно-ориентированного проектирования, некоторые из мероприятий, осуществляемых во время традиционных рассмотрений, могут быть сокращены или автоматизированы. Инструмент Model Advisor, например, осуществляет автоматизированное рассмотрение модели для анализа соответствия стандартам моделирования. Требуется ручное рассмотрение элементов, не покрытых проверками Model Advisor. Соответствие модели требованиям высокого уровня может быть оценено посредством симуляции — подхода, который может быть более эффективным, чем традиционные рассмотрения, и позволяющим находить настоящие проблемы в проекте.

Сокращение или устранение ручных рассмотрений кода

Simulink Code Inspector может использоваться для удовлетворения целей из таблицы MB.A-5 в DO-331, которые обычно удовлетворяются путем ручных рассмотрений кода. Этот инструмент использует формальные методы для систематического рассмотрения блоков, параметров и настроек модели, чтобы убедиться, что они структурно эквиваленты операциям, операторам и данным в сгенерированном коде. Подробный двусторонний анализ трассируемости между моделью и кодом может идентифицировать проблемы, пропущенные при ручных рассмотрениях кода. Результирующие отчеты по структурной эквивалентности и отчеты по трассируемости архивируются и представляются в сертифицирующие органы для аудита, как доказательство соответствия целям из таблицы MB.A-5 в DO-331. Для максимизации эффективности Simulink Code Inspector модели проекта должны разрабатываться с использованием подмножества поддерживаемых примитивных блоков.

Верификация исполняемого объектного кода

Тестирование Процессор-в-контуре (Processor-In-the-Loop, PIL) используется для верификации требований высокого уровня, низкого уровня и производных требований путем выполнения тестовых векторов, основанных на требованиях, на реальном целевом оборудовании (рис. 3). Для этого мероприятия повторно используется набор существующих тестов, основанных на требованиях. Это может устранить необходимость разработки и рассмотрения дополнительных тестовых векторов и кода, часто требующихся для верификации объектного кода традиционными методами. Инструмент анализа покрытия кода используется совместно с PIL тестированием для оценки покрытия и создания отчета о покрытии кода. Инструменты Polyspace для статического анализа кода используются для доказательства отсутствия определенных ошибок времени выполнения в исходном коде и производят требуемую документацию. Результаты тестирования, покрытие кода и отчеты по ошибкам времени выполнения архивируются и представляются в сертифицирующие органы как доказательство соответствия целям из таблицы MB.A-6 и MB.A-7 в DO-331.

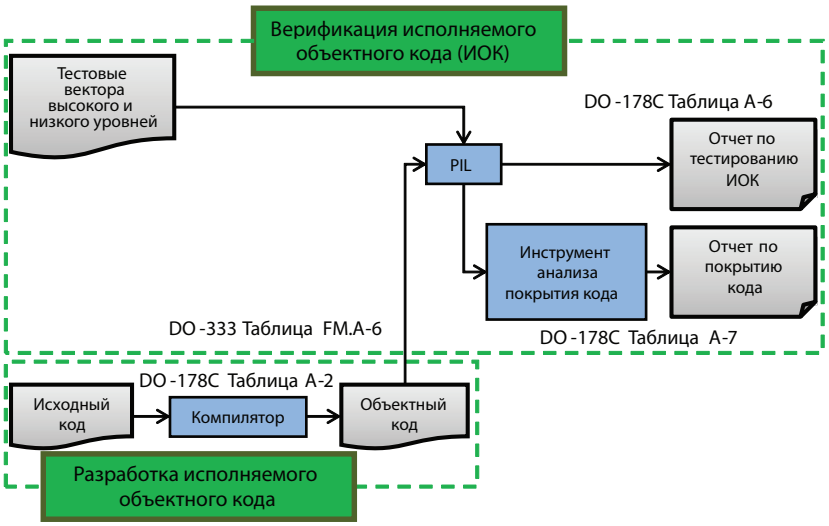

Рис. 3. Верификация исполняемого объектного кода

Генерация доказательств соответствия целям DO-178C

Доказательство соответствия должно быть сгенерировано и представлено в сертифицирующие органы для целей из таблиц A1-A10 Приложения A в DO-178C. С использованием традиционных методов, требуемые артефакты часто разрабатываются вручную. Этот процесс длительный и подверженный ошибкам.

С использованием оптимального процесса модельно-ориентированного проектирования для разработки программного обеспечения, набор требуемых артефактов создается при помощи автоматической генерации отчетов. В таблице 1 приводится список документов, создаваемых для различных мероприятий процесса.

Таблица 1: Генерируемые артефакты

Мероприятие процесса	Документ
Трассируемость модели	Отчет «Описание проекта ПО»
Соответствие модели	Отчет «Проверка на стандарты моделирования»
Верификация модели	Отчет «Результаты тестирования в симуляции» Отчет «Описание проекта ПО»
Трассируемость кода	Отчет «Рассмотрение трассируемости кода»
Соответствие кода	Отчет «Проверка на стандарты кодирования»
Верификация кода	Отчет «Инспекция кода»
Верификация низкого уровня	Отчет «Результаты тестирования исполняемого объектного кода»
Верификация высокого уровня	Отчет «Результаты тестирования исполняемого объектного кода»
Трассируемость объектного кода	Отчет «Генерация кода» Листинг объектного кода (сторонняя среда разработки или компилятор)

Автоматическая генерация требуемых артефактов переносит фокус усилий с разработки артефактов на их рассмотрение. Должны применяться инструменты, которые могут быть квалифицированы — такие, как Simulink Code Inspector, поскольку рассмотрение сгенерированных артефактов в таком случае существенно сокращается.

D. Выявление того, представляют ли модели требования высокого или низкого уровня

Модели могут использоваться для представления как требований верхнего уровня (ТВУ), так и требований низкого уровня (ТНУ). Представление и разбиение требований должны быть тщательно заданы, чтобы определить, как должны быть удовлетворены цели DO-178 из таблиц в Приложении. Очень важно выстроить ясную и четкую трассируемость между ТВУ, ТНУ и сгенерированным исходным и объектным кодом. Хотя DO-178C и его дополнения предлагают некоторые руководства, авиакосмическим компаниям приходится выбирать стратегию определения и реализации требований. Они могут, например, решить использовать текст или моделирование для описания требований. DO-178B, DO-178C и его дополнения разрешают такие стратегии, при условии, что соответствующие таблицы из Приложения удовлетворяются. В этом разделе приводятся различные подходы к использованию моделей для каждого типа требований, а также проблемы, которые нужно принимать во внимание при выборе определенного подхода.

В соответствии с разъяснениями, приведенными в Position Paper CAST-15 (Certification Authorities Software Team) [15], ТВУ обычно представляют, ЧТО должно быть разработано, а ТНУ представляют, КАК это должно быть разработано. Модели, используемые для описания ТВУ, должны содержать данные, приведенные в разделе 11.9 стандарта DO-178C. Эти данные фокусируются на определении высокоуровневого функционала, а также высокоуровневых интерфейсов. С другой стороны, модели, используемые для описания ТНУ, должны содержать данные, приведенные в разделе 11.10. Эти данные определяют детальные аспекты проекта, включая архитектуру программного обеспечения, потоки данных и потоки управления, а также планировщик.

Существуют несколько подходов к разбиению ТВУ и ТНУ между одной или двумя моделями, и каждый подход имеет преимущества и недостатки. В таблице 2 приводится обзор пяти потенциальных вариантов использования моделей для представления требований.

Таблица 2. Варианты представления требований в модели

№	Подход	Преимущества	Недостатки
1	ТНУ определены в моделях, а ТВУ заданы текстовым описанием	• Ясное разделение данных, представляющих ТНУ и ТВУ • Можно применять аспекты модельно-ориентированного проектирования — такие, как генерация кода и анализ покрытия	• Верификация может усложняться из-за необходимости сравнивать текстовые требования с результатами симуляции и тестами на целевом оборудовании
2	ТНУ и ТВУ определены с использованием разных моделей	• Ясное разделение данных, представляющих ТНУ и ТВУ • Можно применять аспекты модельно-ориентированного проектирования — такие, как генерация кода и анализ покрытия • Модель спецификации может использоваться для верификации функционала модели проекта при помощи симуляции	• Требуется поддерживать и верифицировать две модели • Необходимы два разных стандарта моделирования • Вряд ли все ТВУ можно будет представить в модели

№	Подход	Преимущества	Недостатки
3	ТНУ заданы текстовым описанием, а ТВУ определены в моделях	• Ясное разделение ТНУ и ТВУ • Можно применять аспекты модельно-ориентированного проектирования — такие, как анализ покрытия тестами	• Верификация может усложняться из-за необходимости сравнивать текстовые требования с результатами симуляции и тестами на целевом оборудовании • Нельзя применить автоматическую генерацию кода • Вряд ли все ТВУ можно будет представить в модели
4	Один набор моделей используется для определения ТНУ и ТВУ, но не используется для системных требований и в процессах системного проектирования	• Минимальное число артефактов моделирования • Можно применять аспекты модельно-ориентированного проектирования — такие, как генерация кода и анализ покрытия	• Системные требования должны быть достаточно подробными для трассирования к модели проекта
5	Один набор моделей используется для определения ТНУ и ТВУ, и также используется для системных требований и в процессах системного проектирования	• Минимальное число артефактов моделирования • Можно применять аспекты модельно-ориентированного проектирования — такие, как генерация кода и анализ покрытия	• Системные требования должны быть достаточно подробными для трассирования к модели проекта

Приложение MB.B.17 содержит примеры, которые более подробно поясняют, как модели могут использоваться для представления различных уровней требований на разных этапах жизненного цикла разработки.

При использовании модели проекта при разработке программного обеспечения требуется определить, представляет ли эта модель ТНУ, архитектуру программного обеспечения, или и то, и другое. Обычно модели включают некоторые, но не все, аспекты архитектуры программного обеспечения. Например, иерархия ссылок на модели (model reference) описывает архитектуру программного обеспечения, но при этом сгенерированный из моделей код может интегрироваться с драйверами ввода-вывода и операционной системой реального времени. Эта часть архитектуры программного обеспечения определяется независимо от моделей.

При использовании моделей во всем цикле системного проектирования и разработки программного обеспечения нужно искать возможности для повторного использования моделей. Оптимальным вариантом повторного использования является пример 5 из таблицы 2 (также из таблицы MB.1-1 в DO-331), при котором модель используется при разработке системных требований и проектировании. Хотя такой подход не сокращает число целей DO-178C, которые нужно удовлетворить, он предоставляет возможность разрабатывать модель проекта на более раннем этапе разработки.

В случае, когда модели представляют ТВУ и ТНУ, модель ТВУ обязательно требуется рассматривать отдельно от модели ТНУ. Такие аспекты разработки, как стандарты моделирования, окружение для тестирования и мероприятия верификации должны быть определены отдельно для модели ТВУ и модели ТНУ. Тем не менее, поддержка моделей ТВУ и ТНУ в одном окружении моделирования позволит увеличить эффективность путем повторного использования инструментов и верификации посредством симуляции обеих моделей в едином окружении.

Не рекомендуется совмещать ТВУ и ТНУ в одной модели или единице данных [15]. Это обусловлено различными целями каждого уровня требований; ТВУ описывают, что должно делать программное обеспечение системы, в то время как ТНУ описывают, как это должно делаться и какие компоненты используются для того, чтобы это делать. Должна осуществляться раздельная верификация данных как для ТВУ, так и для ТНУ, и совмещение их обоих может сделать это невозможным. Заметьте, что хотя раздел 5 в DO-178C допускает один уровень требований, обычно это не делается, поскольку для этого нужно, чтобы требования системного уровня содержали дополнительные подробности, позволяющие осуществлять трассируемость к совмещенной единице данных ТВУ/ТНУ.

E. Использование симуляции и анализа для поддержки целей DO-178

Симуляция — это техника верификации и валидации, представленная в DO-331. При использовании модельно-ориентированного проектирования симуляция может заменить традиционные рассмотрения и анализы, обеспечив повторяемые доказательства соответствия определенным целям для требований высокого и низкого уровней. Симуляция позволяет инженерам предсказать динамическое поведение системы, что может оказаться эффективней, чем традиционные рассмотрения и анализы при верификации и валидации проекта.

В случае, когда модель проекта содержит требования низкого уровня, симуляция может применяться для демонстрации соответствия требованиям высокого уровня (DO-331, раздел MB.6.3.2.a). Инструменты модельно-ориентированного проектирования, такие, как Simulink Report Generator, используются для выполнения набора тестов, основанных на требованиях, с применением моделей проекта, симуляции динамического отклика системы и автоматического представления результатов в отчете (рис. 4) для ручного рассмотрения. Требуется разработать поддерживающие скрипты для сравнения выходов симуляции с ожидаемыми результатами. Такие скрипты должны быть квалифицированы как инструмент верификации (критерий 2, согласно DO-330) [8].

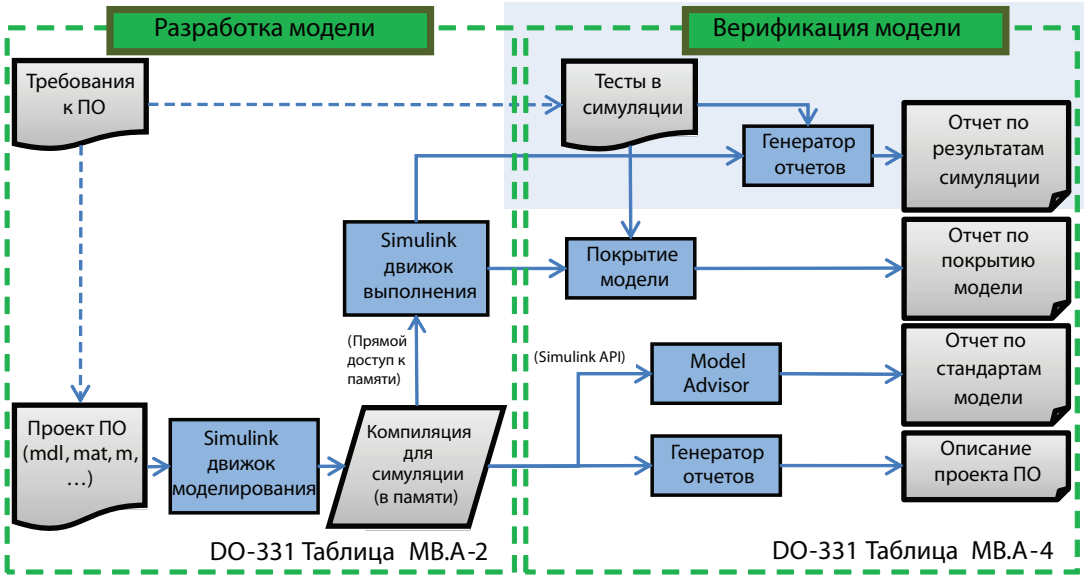

Рис. 4. Симуляция для верификации модели

В соответствии с разделом 6.4.1 в DO-178C [5], платформа для тестирования «Процессор-в-контуре» (Processor-in-the-Loop, PIL) может использоваться для верификации индивидуальных программных компонентов, даже если тестовая платформа не включает финальное целевое оборудование. Для каждой модели проекта требуется разработать тестовую обвязку, которая настроена для связи с отладочной платой, содержащей процессор, идентичный целевой платформе. Каждая тестовая обвязка должна содержать модель на верхнем уровне, которая ссылается на соответствующую модель проекта (посредством механизма model reference). Инструменты модельно-ориентированного проектирования, такие, как Simulink Report Generator, используются для выполнения набора тестов, основанных на требованиях, на целевой платформе и автоматически представляют результаты в отчете (рис. 5). Этот отчет рассматриваются вручную, для осуществления верификации того, что исполняемый объектный код удовлетворяет требованиям высокого уровня (DO-331, таблица MB.A-6).

Рис. 5. Тестирование на целевой платформе с использованием механизма "Процессор-в-контуре" (Processor-In-The-Loop, PIL)

F. Анализ на более высоком уровне абстракции результатов тестирования относительно требований

DO-178C позволяет использовать симуляцию для верификации функциональности вместо — или совместно — с ручными рассмотрениями данных проекта. В дополнение, использование симуляции при тестировании дает другие преимущества — такие, как покрытие модели, показывающее, насколько тщательно протестирован проект.

Без модельно-ориентированного проектирования требования к программному обеспечению традиционно определяются с использованием текстовых документов или статических графических спецификаций, таких, как диаграммы потоков. Такие данные проекта более высокого уровня не могут использоваться для верификации сами по себе. Сначала их требуется преобразовать в исходный код, а затем запустить либо на симуляторе, либо на целевом оборудовании. Тестовые вектора также должны быть разработаны на основании требований. При использовании модельно-ориентированного проектирования такой тип тестирования может осуществляться на более высоком уровне абстракции. Это означает, что модели, представляющие проект, требования, или и то, и другое, могут использоваться напрямую в мероприятиях верификации.

Существуют несколько аспектов этой лучшей практики. Во-первых, требуется выстроить процессы и методы для применения и анализа трассируемости от требований к модели и от требований к тестовым векторам. Во-вторых, требуется выстроить процесс для разработки тестовых векторов в окружении для моделирования и симуляции. В-третьих, такие техники, как анализ покрытия модели, должны применяться для измерения адекватности тестов, разработанных на основании требований, и для выявления дефектов проекта — таких, как мертвая логика. Наконец, могут использоваться формальные методы и соответствующие инструменты для автоматического создания тестов, основанных на требованиях в определенных случаях для определенных сред моделирования или для выявления ошибок проектирования — таких, как целочисленное переполнение или деление на ноль.

Трассируемость требований является одним из фундаментальных столбов DO-178. Установление ясной и четкой трассируемости между требованиями высокого уровня, моделями, исходным кодом, объектным кодом и тестами может быть сложной задачей. Лучшая практика заключается в оценке покрытия модели с использованием тестов, основанных на требованиях, а затем использование результатов этого анализа для выявления пропущенных требований, производных требований и других проблем фазы проектирования. Это помогает избежать траты усилий на генерацию программного кода с известными дефектами и может снизить расходы, связанные с исправлением проблем из-за пропущенных или некорректных требований на последующих стадиях процесса разработки [9]. Многие инструменты для моделирования предлагают возможности по трассируемости внешних требований к модели или к тестам, используемым в окружении для симуляции. Эти возможности часто предлагают анализ трассируемости наряду с установлением трассируемости.

Тестовые вектора и процедуры тестирования, так же, как архитектура программного обеспечения и проект программного обеспечения, должны быть разработаны из ТВУ и ТНУ. Эти тестовые вектора и процедуры могут быть разработаны в окружении для моделирования и использоваться во время симуляции, в поддержку мероприятий формальной верификации. DO-331, раздел MB.6.8.1 описывает мероприятия, которые должны быть осуществлены для этого. Многие из этих мероприятий включают определение того, подходит ли симуляция для верификации требований и полностью ли удовлетворяет симуляция определенные цели рассмотрений или анализов. При использовании симуляции для мероприятий верификации рекомендуется использовать покрытие модели для определения требований, выраженных в модели, которые не были проверены посредством верификации, основанной на требованиях, по которым разрабатывалась модель [6]. Заметьте, что покрытие модели отделено от структурного покрытия в соответствии с DO-178C, раздел 6.4.4.2.

Инструменты формальных методов, такие, как Simulink Design Verifier, могут использоваться для идентификации «мёртвой» логики в модели, а данные покрытия модели могут использоваться для создания тестов для любых требований, которые не были протестированы. Хотя сертификационные зачеты не могут быть получены при использовании этих инструментов на уровне модели, идентификация таких проблем в проекте до генерации исходного кода сокращает последующие усилия.

G. Использование инструментов модельно-ориентированного проектирования, которые могут быть квалифицированы

DO-178C разрешает использование инструментов, которые устраняют, сокращают или автоматизируют процессы DO-178C. Организации могут сократить или устранить рассмотрения кода, например, при использовании инструмента, который автоматизирует цели DO-178C по верификации исходного кода относительно требований низкого уровня. Эти инструменты должны быть квалифицированы, как указано в RTCA DO-330 «Software Tool Qualification Considerations» («Квалификация программных инструментов») [9]. Лучшая практика заключается в использовании и квалификации максимально возможного числа инструментов верификации при модельно-ориентированном проектировании во время жизненного цикла процесса разработки. Квалификация инструментов разработки обычно не осуществляется (например, не существует коммерческих квалифицированных компиляторов), поскольку это дает малый возврат инвестиций по сравнению с автоматизированным подходом, заключающимся в верификации выходов инструментов разработки при помощи квалифицированных инструментов верификации.

Наборы для квалификации инструментов («киты») предоставляют артефакты и руководства для упрощения процесса квалификации инструмента. Наборы для квалификации могут расширяться с учетом особенностей стандартов компании и требований. Однако, для сокращения усилий нужно минимизировать расширение инструментов модельно-ориентированного проектирования и их соответствующих наборов для квалификации. В дополнение, артефакты квалификации инструмента должны быть сгенерированы в том же самом окружении для разработки, которое используется для разработки основного приложения. Традиционно, верификация кода относительно требований является длительным и подверженным ошибкам процессом, который требует ручного рассмотрения кода построчно относительно списка проверок в проекте для удовлетворения целей верификации исходного кода из таблицы A-5 в DO-178C. Автоматическая инспекция кода при помощи Simulink Code Inspector может значительно сократить время, требуемое для удовлетворения этих целей. Модели проекта должны разрабатываться с использованием подмножества блоков, поддерживаемых Simulink Code Inspector (рис. 6).

Рис. 6. Simulink Code Inspector поддерживает подмножество доступных блоков

Традиционно, артефакты, полученные вручную, разрабатываются и рассматриваются для удовлетворения целей из таблиц A-3 и A-4 в DO-178C [5]. Списки проверок разрабатываются и используются для таких мероприятий, как верификация соответствия стандартам и трассируемости к требованиям.

С использованием модельно-ориентированного проектирования цели из таблиц MB.A-3 и MB.A-4 в DO-331 [6] могут быть удовлетворены комбинацией следующих мероприятий:

- рассмотрение подробного отчета, описывающего проект системы;
- рассмотрение и симуляция модели проекта;
- проверки Model Advisor.

Simulink Report Generator должен использоваться для автоматической генерации из моделей проекта отчета «Описание проекта ПО» — описывающего подробный проект системы. Генерация отчета «Описание проекта ПО» можно настраивать для включения или исключения определенного содержимого. Окончательный отчет, тем не менее, должен включать ключевые детали, характеризующие проект системы, включая:

- интерфейсы верхнего уровня и уровня подсистем;
- ссылки на требования высокого уровня;
- переменные в проекте;
- параметры блоков;
- параметры настроек модели проекта;
- внешние зависимости.

Возможность генерации отчета «Описание проекта ПО» в инструменте Simulink Report Generator должна быть квалифицирована, чтобы получить зачеты сертификации за удовлетворение целей из таблиц MB.A-3 и MB.A-4 в DO-331 [6]. Для упрощения процесса квалификации инструмента требуется использовать набор для квалификации («кит»).

Инструменты модельно-ориентированного проектирования, такие, как Simulink Report Generator, также должны использоваться для выполнения тестов, основанных на требованиях, с использованием моделей проекта. Хотя рассмотрение численных результатов симуляции и PIL относительно ожидаемых выходов может быть осуществлено вручную, лучшая практика заключается в разработке и квалификации собственных скриптов, которые осуществляют это сравнение. Такие скрипты должны поддерживать разные типы данных (единичной точности, двойной точности, целочисленные), а также реализовывать лучшие практики для сравнения чисел с плавающей точкой.

II. Заключение

Документ DO-178B предоставляет мало руководств по использованию моделей. DO-331, будучи дополнением для недавно выпущенного DO-178C, предоставляет дополнительные руководства по артефактам и доказательствам верификации при использовании моделей. Эти документы фокусируются на процессах и целя[верификации, предоставляя авиакосмическим компаниям самим решать, как наиболее эффективно применять модельно-ориентированное проектирование ко всему жизненному циклу разработки программного обеспечения. Лучшие практики, представленные в этой статье, собраны из знаний, полученных в результате многолетнего взаимодействия с заказчиками в авиакосмической промышленности, разрабатывающих программное обеспечение повышенной надежности. Эти лучшие практики могут быть внедрены для реализации эффективного рабочего процесса модельно-ориентированного проектирования при разработке программного обеспечения, сертифицированного по DO-178.

Ссылки

1. “Design Times at Honeywell Cut by 60 Percent,” W. King, Honeywell Commercial Aviation Systems, Nov. 1999, http://www.mathworks.com/company/user_stories
2. “Flight Control Law Development for F-35 Joint Strike Fighter,” D. Nixon, Lockheed-Martin Aeronautics, Oct. 2004, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
3. “ESA's First-Ever Lunar Mission Satellite Orbits Moon with Automatic Code,” P. Bodin, Swedish Space, Oct. 2005, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
4. “Automatic Code Generation at Northrop Grumman,” R. Miller, Northrop Grumman Corporation, June 2007, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
5. “Software Considerations in Airborne Systems and Equipment Certification,” RTCA/DO-178C, RTCA Inc. Dec. 2011.
6. “Model-Based Development and Verification Supplement to DO-178C and DO-278A,” RTCA/DO-331, RTCA Inc. Dec. 2013.
7. “Software Considerations in Airborne Systems and Equipment Certification,” RTCA/DO-178B, RTCA Inc. Dec. 1992.
8. “Software Tool Qualification Considerations,” RTCA/DO-330, RTCA Inc. Dec. 2011.
9. J.B. Dabney, "Return on Investment of Independent Verification and Validation Study Preliminary Phase 2B Report." Fairmont, W.V.: NASA IV&V Facility, 2003.
10. “Model-Based Design for DO-178B with Qualified Tools,” Tom Erkkinen, Bill Potter, The MathWorks, Inc., AIAA Modeling and Simulation Technologies Conference and Exhibit 2009, AIAA Paper 2009-6233.
11. “Complying with DO-178C and DO-331 using Model-Based Design,” Bill Potter, The MathWorks, Inc., SAE Paper 12AEAS-0090.
12. “Best Practices for Establishing a Model-Based Design Culture,” Paul F. Smith, Sameer M. Prabhu, Jonathan H. Friedman, The MathWorks, Inc., SAE Paper 2007-01-0777.
13. “Pragmatic Strategies for Adopting Model-Based Design for Embedded Applications,” Eric Dillaber, Larry Kendrick, Wensi Jin and Vinod Reddy, The MathWorks, Inc., SAE Paper 2010-01-0935.
14. “Large Scale Modeling for Embedded Applications,” Kerry Grand, Vinod Reddy, Gen Sasaki, Eric Dillaber, The MathWorks, Inc., SAE Paper 2010-01-0938.
15. “Merging High-Level and Low-Level Requirements,” Certification Authorities Software Team (CAST), Position Paper CAST-15 Feb. 2003.