Лучшие подходы к реализации руководств по моделированию в Simulink

Дэвид Джефри (David Jaffry), компания MathWorks

Встраиваемые системы становятся всё более масштабными и сложными. В то же время команды проектировщиков становятся более рассредоточенными – как с точки зрения географии, так и с точки зрения навыков и опыта членов команды. С учетом такого сложного окружения для разработки, исключительную важность приобретает внедрение руководств по моделированию, обеспечивающих согласованное моделирование. Руководства устанавливают однородный подход к моделированию внутри команды проектировщиков, упрощая повторное использование моделей для новых проектов. Руководства также помогают новым членам команды освоиться с процессом разработки.

Для некоторых проектов установление руководств по моделированию является не просто лучшей практикой – это является требованием. Программные системы, работающие в приложениях повышенной надежности в авиакосмической и других областях промышленности, должны удовлетворять строгим стандартам по разработке и верификации. Промышленные стандарты, такие, как ISO 26262, EN 50128, IEC 61508 и DO-178C требуют применения руководств по моделированию (Рисунок 1).

В этой статье описываются лучшие методы по созданию, реализации, валидации и внедрению новых руководств по моделированию. Рекомендуемые подходы позволят вам эффективно применить стандарт внутри команды и упростить квалификацию руководств по моделированию.

Table 1 — Topics to be covered by modelling and coding guidelines

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity[a]	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++

[a] An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.

[b] The objectives of method 1b are

— Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.

— Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.

— Exclusion of language constructs which could result in unhandled run-time errors.

[c] The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

Рисунок 1. Руководства по моделированию, требуемые стандартом ISO 26262.

Лучшие подходы к определению правил

Руководства по моделированию могут и должны адресовать несколько требований - включая читаемость, повторное использование и беспроблемный обмен моделями Simulink®. Руководства по моделированию обеспечивают верификацию не только самих моделей, но также и других связанных сущностей – включая свойства объектов в модели и переменных в рабочем пространстве.

По этой причине руководство по моделированию состоит из набора правил. В руководстве перечисляется проверки, применимые к этому руководству и приводятся подробные рекомендации по исправлению проблем.

Важно определять правила, которые понимаются однозначно и могут быть автоматически проверены. Если правило не может быть проверено автоматически, требуется сопоставить важность данного правила со временем, требуемым на его ручную проверку.

В таблице 1 приводятся примеры того, как можно улучшить правила, описанные двусмысленно или субъективно.

Правило	**Проблема**	**Рекомендуемое изменение**
Модель должна быть читаемой	Субъективное	Заменить на “Каждый лист модели должен содержать менее 50 блоков. Максимальная вложенность подсистем равняется 10.”
Модель должна генерировать код	Неполное	Уточнить, что “Модель должна быть совместима с целевым файлом Embedded Coder ert.tlc.”
Модель должна иметь оптимальную производительность	Неясное	Указать, какие характеристики модели должны быть оптимизированы.

Таблица 1. Примеры правил и рекомендуемые улучшения.

Перед тем, как создавать новое правило, рассмотрите возможность использования существующих правил в Simulink, Simulink Verification and Validation™, Embedded Coder®, IEC Certification Kit и DO Qualification Kit. Большинство правил в этих продуктах являются квалифицируемыми и могут использоваться в процессах, подлежащих сертификации. При повторном использовании существующих правил сокращаются усилия на создание ваших собственных руководств по моделированию – не только в краткосрочной, но и в долгосрочной перспективе, поскольку эти правила поддерживаются для каждого нового релиза самими разработчиками MATLAB.

Каждое правило в ваших руководствах по моделированию следует составлять согласно шаблону, который включает следующие элементы:

- Заголовок
- Идентификатор
- Описание
- Обоснование
- Уровень (обязательное или рекомендуемое)

- Примеры применения (хотя бы один пример, который соответствует правилу и хотя бы один пример, нарушающий правило – предпочтительно с вариантом решения нарушения)
- Ссылка на ответственного за реализацию проверки правила

Кроме того, руководства по моделированию должны указывать версию (версии) MATLAB и Simulink, к которым применяются правила. Возможности по верификации меняются от релиза к релизу. Например, в R2012b некоторые инструменты верификации поддерживали шину массивов, но не поддерживали массив шин. Связывая обоснование правила с версией продукта, вы избегаете включения устаревших правил в ваши процессы разработки. Руководство по моделированию должно также включать ссылки на все внешние документы со стандартами.

На рисунке 2 показан пример руководства.

hisf_0002: User-specified state/transition execution order

ID: Title	hisf_0002: User-specified state/transition execution order	
Description	Do the following to explicitly set the execution order for active states and valid transitions in Stateflow charts:	
	A	In the Chart Properties dialog box, select **User specified state/transition execution order**.
	B	In the Stateflow Editor **View** menu, select **Show Transition Execution Order**.
	C	Set default transition to evaluate last.
Note	Selecting **User specified state/transition execution order** restricts the dependency of a Stateflow chart semantics on the geometric position of parallel states and transitions. Specifying the execution order of states and transitions allows you to enforce determinism in the search order for active states and valid transitions. You have control of the order in which parallel states are executed and transitions originating from a source are tested for execution. If you do not explicitly set the execution order, the Stateflow software determines the execution order following a deterministic algorithm. Selecting **Show Transition Execution Order** displays the transition testing order.	
Rationale	A, B, C	Promote an unambiguous modeling style.
Model Advisor Checks	▪ **By Task** > **Modeling Standards for DO-178C/DO-331** > **Check Stateflow charts for ordering of states and transitions** ▪ **By Task** > **Modeling Standards for IEC 61508** > **Check usage of Stateflow constructs** ▪ **By Task** > **Modeling Standards for ISO 26262** > **Check usage of Stateflow constructs** ▪ **By Task** > **Modeling Standards for EN 50128** > **Check usage of Stateflow constructs**	
References	This guideline supports adhering to: ▪ IEC 61508-3, Table A.3 (3) 'Language subset' ▪ ISO 26262-6, Table 1 (b) 'Use of language subsets' ISO 26262-6, Table 1 (f) 'Use of unambiguous graphical representation' ▪ EN 50128, Table A.4 (11) 'Language Subset' ▪ DO-331, Section MB.6.3.3.b 'Software architecture is consistent' DO-331, Section MB.6.3.3.e 'Software architecture conform to standards '	
See Also	The following topics in the Stateflow documentation ▪ Transition Testing Order in Multilevel State Hierarchy ▪ Execution Order for Parallel States	
Last Changed	R2013b	

Рисунок 2. Пример руководства по моделированию (одно из правил руководства "Моделирование систем повышенной надежности", High Integrity Systems Modeling).

Реализация руководств по моделированию

Инструмент Model Advisor из Simulink Verification and Validation обеспечивает основу для установления стандартов моделирования. Model Advisor проверяет модель Simulink на несоответствия и на объекты модели, которые не удовлетворяют руководствам. Каждое руководство перечисляет проверки, применимые к этому руководству и приводит подробные рекомендации по исправлению проблем.

Инструмент Model Advisor предлагает много проверок, которые могут быть повторно использованы. Можно также создавать собственные проверки, используя API – программный интерфейс, доступный в Simulink Verification and Validation. С проверкой

можно связать действие "Исправить" (Fix) для автоматического исправления найденных нарушений.

Для всех правил, представленных в Model Advisor, следует использовать одинаковый шаблон. Для каждого нарушенного правила надо указать местоположение, саму проблему и предлагаемое исправление. Также полезно включить параграф "См. также" (See also), который ссылается непосредственно на документацию к руководствам по моделированию.

Обычно каждое правило, определенное в руководствах по моделированию, соответствует отдельной проверке в Model Advisor. Каждое правило также реализуется в отдельном файле MATLAB. Model Advisor API упрощает создание проверок, позволяя указать функции MATLAB, используемые для проверок, а также предлагая возможности по форматированию результатов - включая изображения, таблицы и ссылки. На рисунке 3 показан пример проверки с форматированием результатов.

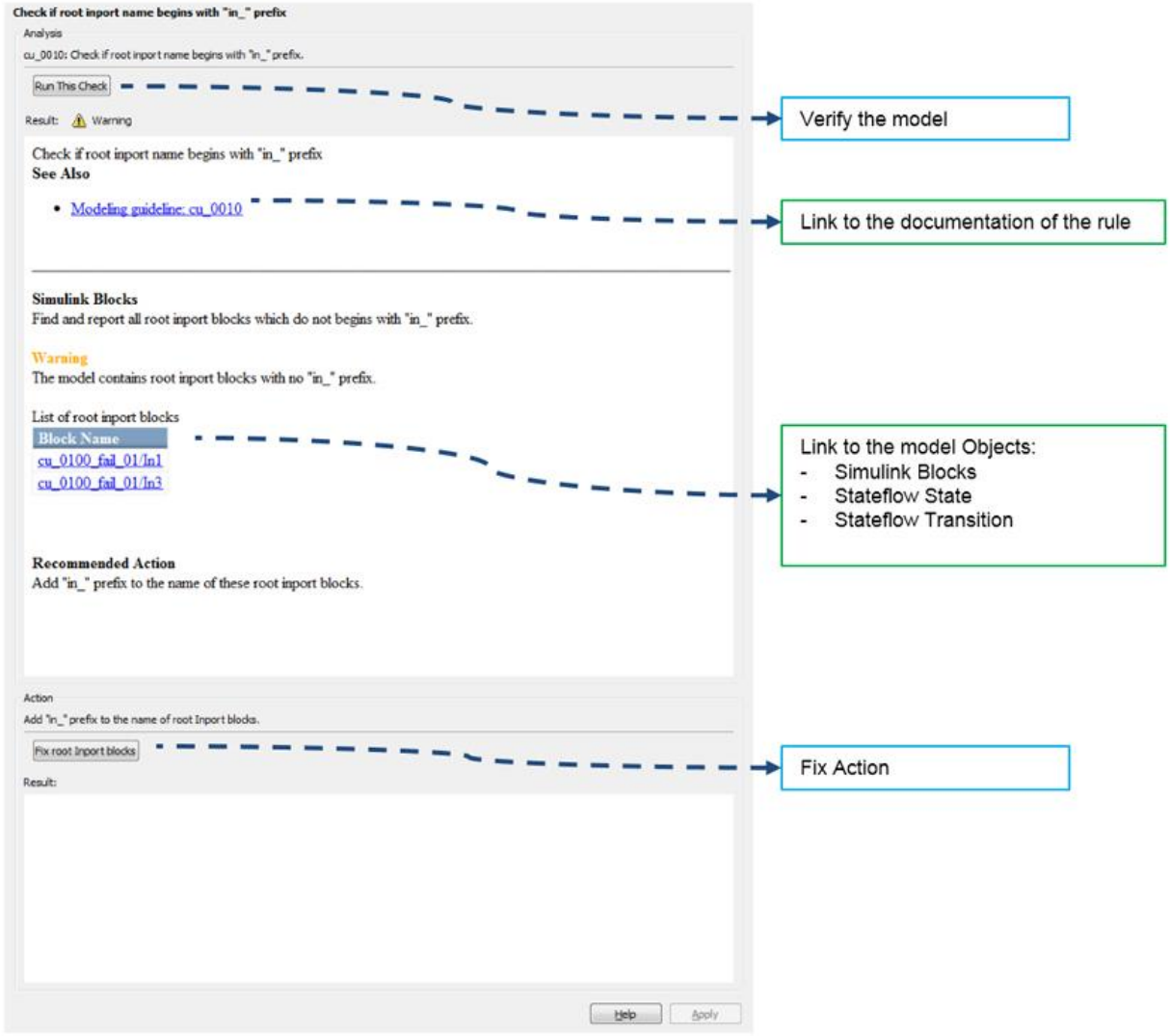

Рисунок 3. Результат проверки с использованием Model Advisor API. (Верификация модели. Ссылка на документацию по правилу. Ссылка на объекты в модели: блоки Simulink, состояния Stateflow, переходы Stateflow. Действие "Исправить").

Программные интерфейсы Simulink или Stateflow используются в-основном при создании проверок, осуществляющих верификацию модели Simulink. Например:

```
% Искать все блоки 'Inport' при помощи функции find_system
inportBlockList = find_system(bdroot,'BlockType','Inport');
% Получить имена всех блоков 'Inport' при помощи функции get_param
inportBlockName = get_param(inportBlockList,'Name');
```

В таблице 2 приводятся примеры функций из Simulink API, позволяющих создавать собственные проверки и верифицировать некоторые свойства модели Simulink.

Функция	**Описание**
find_system	Поиск систем, блоков, линий, портов и аннотаций
get_param	Получение значений параметров систем и блоков
bdroot	Возвращает имя корневой системы Simulink (полезно при написании проверки)
gcb	Получение имени пути текущего блока (полезно при написании проверки)

Таблица 2. Примеры функций API.

Обратите внимание, что некоторые проверки требуют обновления (Update) модели Simulink, поскольку им требуется управлять типами данных или размерностями сигналов или портов. Скомпилированная модель полезна, поскольку в ней указаны типы данных всех портов и размерности всех сигналов.

Следующая команда выполняет фазу компиляции модели:

```
try
   %% Скомпилировать модель
   feval(bdroot(systemToCheck), [], [], [], 'compile');
catch ME
   %% ОШИБКА КОМПИЛЯЦИИ
   ...
end
```

Имя проверки, которая требует обновления модели, начинается с символа “^”. Для упрощения идентификации лучше отделить такие проверки от других. Обратите внимание, что эти проверки могут быть более медленными; для повышения эффективности лучше собрать все проверки, требующие обновления модели, в одну проверку, содержащую вложенные проверки.

Валидация руководств по моделированию

Для каждого правила требуется предоставить примеры моделей, удовлетворяющих и нарушающих проверки. Эти модели должны быть максимально простыми, чтобы подчеркнуть конкретное нарушение или, наоборот, соответствие правилу.

Эти модели могут использоваться для валидации или сертификации проверок в руководствах по моделированию и для осуществления тестирования самих правил при их обновлении.

На рисунке 4 приводится пример иерархии тестовых моделей.

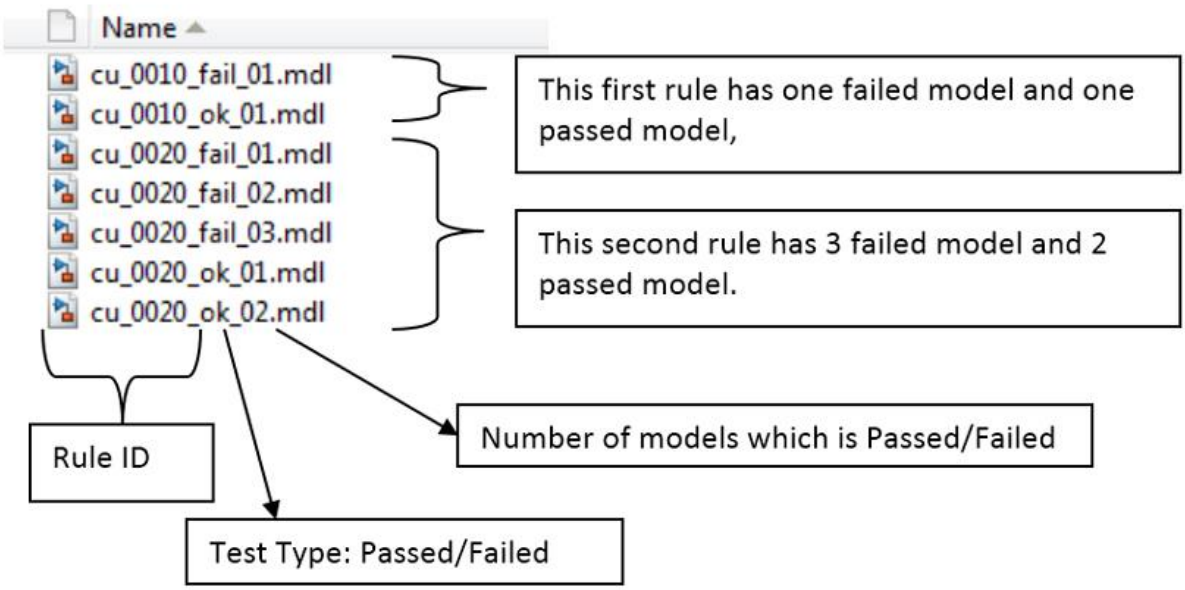

Рисунок 4. Пример иерархии тестовых моделей. (Это первое правило, которое содержит одну модель, нарушающую правило, и одну модель, соответствующую правилу. Это второе правило, которое содержит 3 модели, нарушающих правило и 2 модели, соответствующих правилу. Число моделей, которые Соответствуют/Нарушают правило. Тип теста: Соответствует/Нарушает правило. Идентификатор правила).

Организация и продвижение руководств по моделированию

Редактор конфигурации (Configuration Editor) в Model Advisor позволяет организовать и оставить только те правила, которые вы хотите передать другим пользователям. Конфигурация Model Advisor хранится в MAT файле, который можно распространять на уровне команды разработчиков.

На рисунке 5 показан пример руководств по моделированию, настроенных при помощи редактора конфигурации. Заметьте, что все неиспользуемые проверки были убраны из интерфейса Model Advisor.

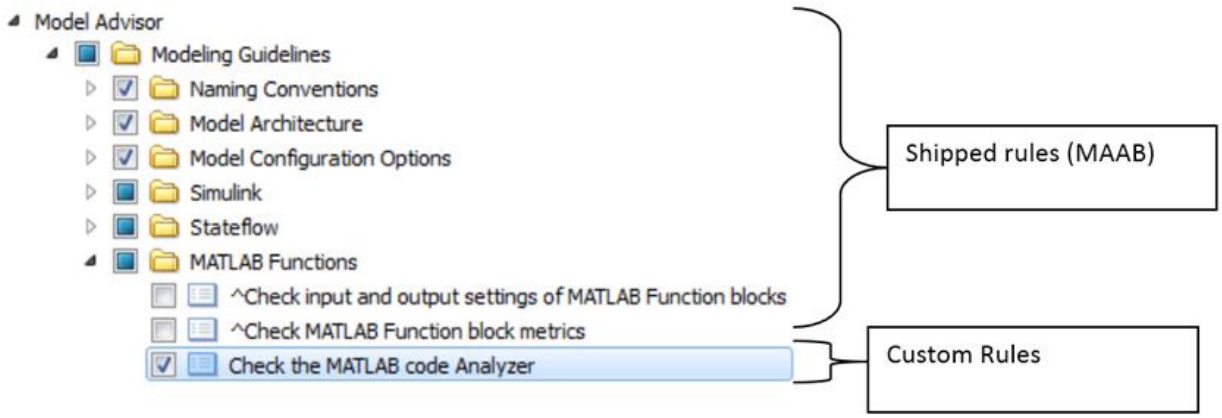

Рисунок 5. Пример настроенной конфигурации. (Встроенные правила (MAAB). Собственные правила)

Теперь вы готовы распространить руководства по моделированию в ваших командах проектировщиков. Для этого требуется создать набор, в котором содержится следующее:

- Текущая версия ваших руководств по моделированию
- Реализация проверок
- Модели Simulink, используемые для осуществления валидации или тестирования самих проверок после их обновления
- Документация
- Настроенная конфигурация в виде MAT файла, созданного редактором конфигурации Model Advisor
- Файл MATLAB sl_customization.m для использования настроенной конфигурации

Хорошая идея применить набор руководств по моделированию в начале проекта. Команде проектировщиков будет проще принять новые руководства, если внедрение будет осуществляться поэтапно. Это также позволит вам выбирать проверки, относящиеся к определенной стадии вашего процесса разработки.

Об авторе

Дэвид Джефри является консультантом в MathWorks. Дэвид специализируется на помощи автомобильным, авиакосмическим, биотехническим и промышленным компаниям при внедрении модельно-ориентированного проектирования для разработки встраиваемых систем. До MathWorks он работал в компаниях Silicomp и Polyspace Technologies над разработкой программного обеспечения и верификации кода для встраиваемых систем. Дэвид David обладает степенью магистра по компьютерной инженерии в университете École Nationale d’Ingénieurs de Brest.