# Тестирование проектов и требований при помощи моделирования и симуляции

*Майкл Кароне (Michael Carone), компания MathWorks*

Моделирование является эффективным и малозатратным способом представления реальной системы. Модель может представлять ключевые аспекты системы, включая лежащие в основе требования, компоненты и связи компонентов друг с другом. Можно осуществлять симуляцию модели, что позволяет проектировщикам начинать тестирование до того, как будет готово оборудование. Кроме того, можно тестировать сценарии, которые сложно или дорого воспроизвести в реальном мире. Итерации между моделированием и симуляцией могут улучшить качество проекта системы на ранней стадии, сокращая число ошибок, найденных позже в процессе проектирования.

Несмотря на эти преимущества, проектировщики, которые сильно зависят от написания кода вручную, не всегда используют все возможности моделирования и симуляции. Создание тестов может быть сложным и трудоемким процессом, и, когда для разных инженерных областей используются разные инструменты, получить системное представление о проекте может быть сложно. Как результат, дефекты, которые могли бы быть найдены на стадии моделирования и симуляции, часто находятся во время стадии реализации, когда исправлять ошибки сложнее и дороже.

Эти проблемы адресуются в Simulink®, который является платформой для моделирования и симуляции. Simulink поддерживает не только моделирование многодисциплинарных систем, но также симуляцию с использованием собственных решателей дифференциальных уравнений (ODE). Фундаментальное преимущество использования Simulink заключается в том, что можно представлять различные инженерные области знаний (т.н. домены), включая системы управления, машины состояний и модели окружающей среды – все в одной модели. При этом в Simulink запускаются симуляции для верификации корректности построения модели. Во время симуляции предоставляются возможности по анализу результатов, включая отображение данных, анимацию состояний и условные точки останова. По завершении симуляции можно анализировать любые записанные данные при помощи скриптов MATLAB® и инструментов визуализации.

В этой статье приводится рабочий процесс создания модели компонента на базе требований. Описывается симуляция и тестирование модели компонента, а также подключение его к модели системного уровня для дальнейшей симуляции и тестирования. Чтобы проиллюстрировать этот подход, мы создадим и протестируем компонент, отвечающий за выявление неисправностей, их изоляцию и восстановление (fault detection, isolation and recovery - FDIR) в проекте HL-20 - возвращаемого аппарата, спроектированного в НАСА как дополнение к программе Спейс Шаттла. Мы подключим наш компонент к модели системного уровня,  которая включает модели окружающей среды, систему управления полетом, а также систему наведения, навигации и управления, а затем осуществим симуляцию системной модели для валидации её поведения.

Модель, используемая в этом примере, доступна в Aerospace Blockset™.

## Построение модели компонента на основании требований

На первом шаге осуществляется моделирование логики управления неисправностями исполнительного механизма. Документ с требованиями содержит пять различных режимов работы исполнительного механизма: пассивный (passive), в ожидании (standby), активный (active), изолированный (isolated) и отключенный (off). Для упрощения, мы рассмотрим только первые четыре режима. Мы представим эти режимы, добавив четыре состояния в диаграмму Stateflow® (Рисунок 1).

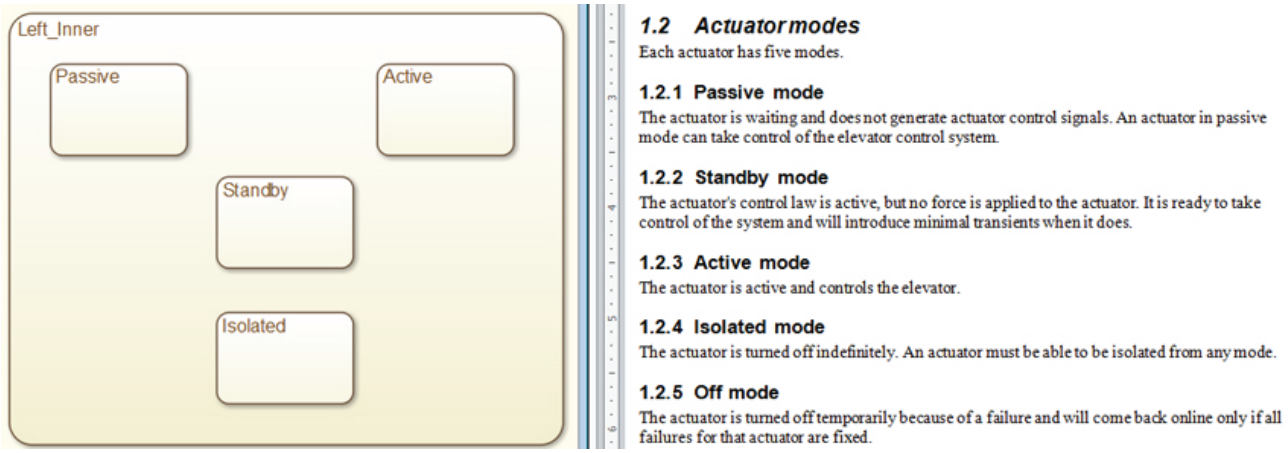


*Рисунок 1. Диаграмма Stateflow, отображающая режимы работы исполнительного механизма, представленные в виде состояний.*

Затем нам требуется определить, как система будет переходить из одного состояния (или режима) в другое. Используя информацию, представленную в документе с требованиями, мы добавляем переходы, соединяющие состояния, и указываем, какие условия должны выполняться, чтобы состояние системы переключилось. Мы также группируем состояния Passive, Active и Standby в одно суперсостояние, поскольку они все переходят в состояние Isolated по одинаковому условию. Такая техника иерархического моделирования помогает нам моделировать сложную логику в простой визуальной форме (Рисунок 2).

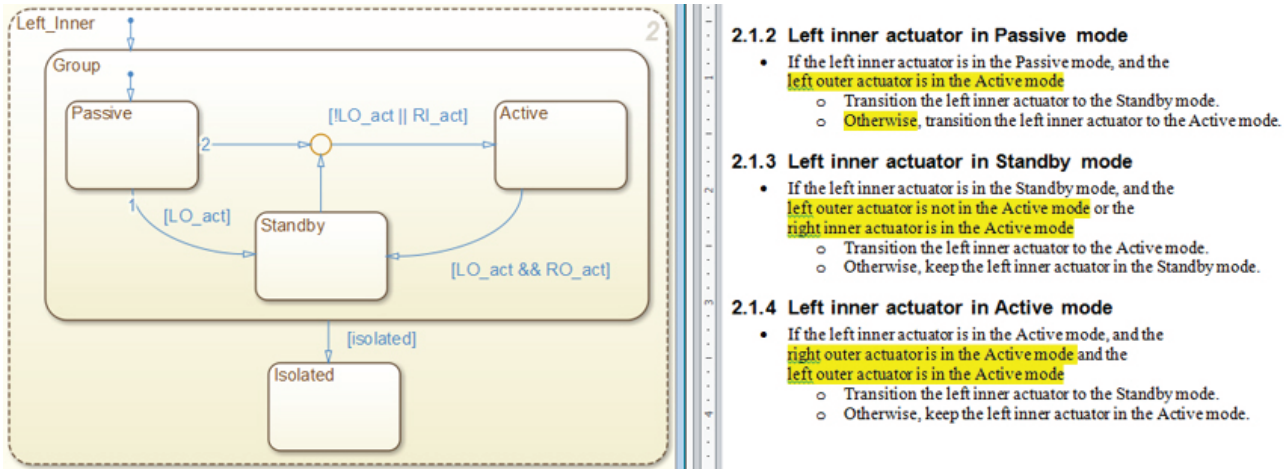


*Рисунок 2. Диаграмма состояний, созданная по требованиям.*

Мы продолжаем строить модель, соединяя каждый элемент с соответствующим требованием к системе (Рисунок 3). Затем мы сможем отследить модель обратно к документу с требованиями, чтобы объяснить, почему была выбрана такая реализация в модели.

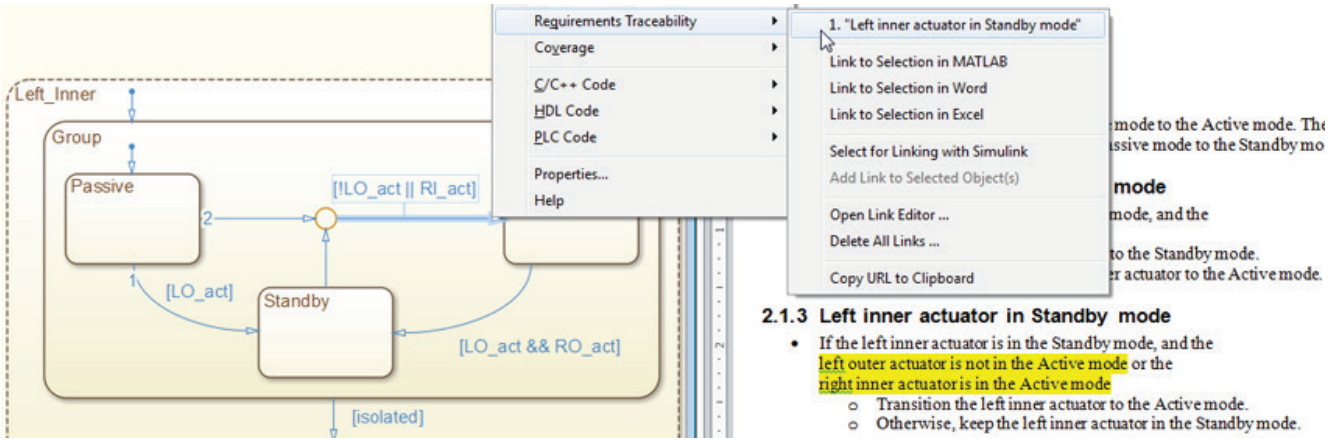


*Рисунок 3. Требования, связанные с определенными частями диаграммы состояний.*

Когда мы построили логику для левого внутреннего исполнительного механизма, мы можем повторно использовать эту логику для правого внутреннего исполнительного механизма, поскольку его структура совершенно такая же. Единственное, что требуется поменять, это условия, которые соответствуют каждому переходу, как описано в документе с требованиями (Рисунок 4).

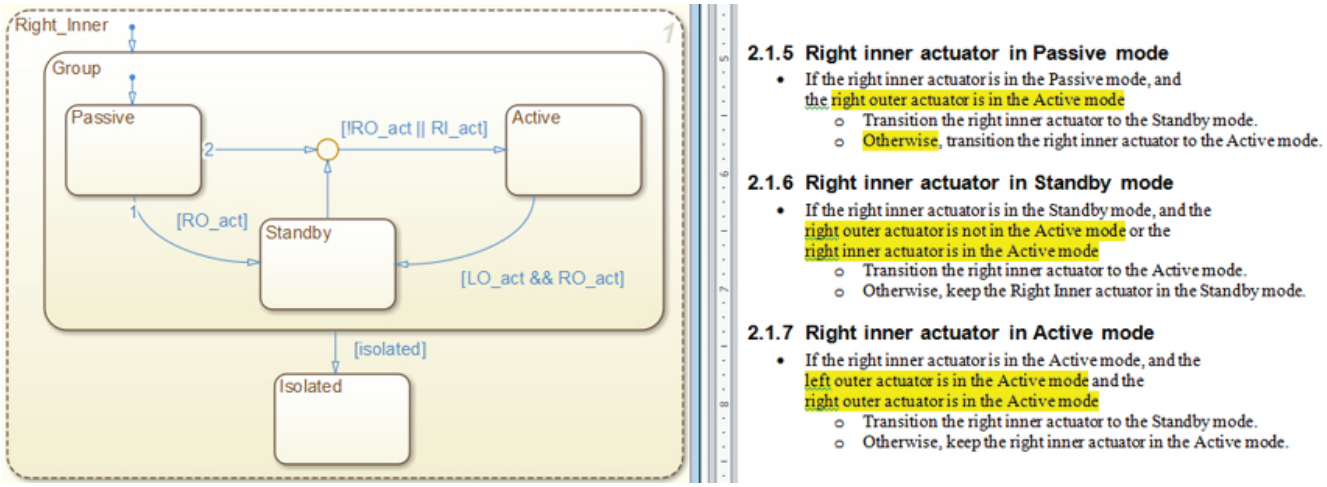


*Рисунок 4. Управляющая логика для правого внутреннего исполнительного механизма.*

## Тестирование компонента посредством симуляции

Теперь, когда компонент частично собран, мы готовы запускать симуляции для верификации корректного поведения. Чтобы сделать это, мы создадим простую тестовую обвязку, которая подает входные сигналы на компонент, используя комбинацию блоков switch (переключатель) и constant (постоянная) (Рисунок 5).

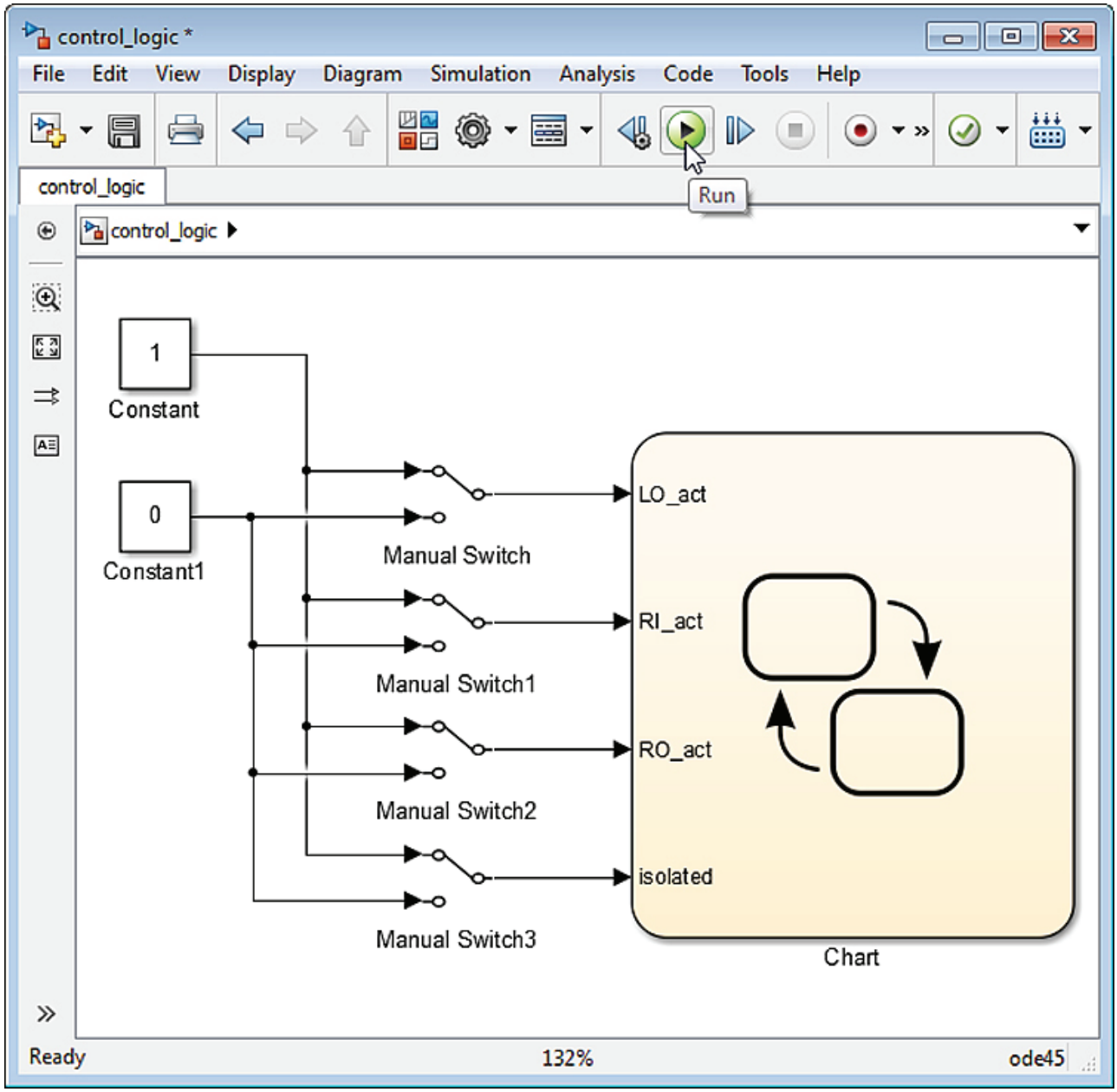


*Рисунок 5. Тестовая обвязка для тестирования управляющей логики.*

В Simulink и Stateflow мы можем запускать симуляцию без необходимости вручную задавать переменные. Когда мы нажимаем кнопку **Play** в первый раз, появляется диалоговое окно с указанием переменных, которые требуется определить до запуска симуляции. Когда мы нажимаем **OK**, эти переменные автоматически создаются (Рисунок 6).

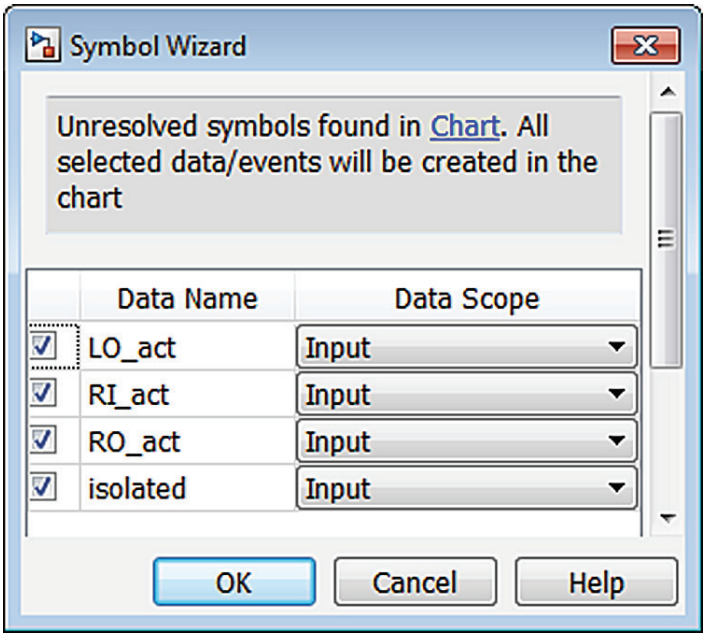


*Рисунок 6. Stateflow Symbol Wizard (мастер символов) для автоматического определения переменных, используемых в логике.*

Во время симуляции диаграмма состояний анимируется, показывая, какое состояние активно в данный момент времени, и как система переходит из одного состояния в другое. В данном случае тестирование путем переключения входных сигналов из включенного в выключенное состояние выявило ошибку в проекте (Рисунок 7). Когда активируется левый внутренний исполнительный механизм, правый внутренний тоже должен активироваться. Тот факт, что нам удалось выставить входные условия таким образом, что это не случилось, указывает на ошибку в нашем проекте.

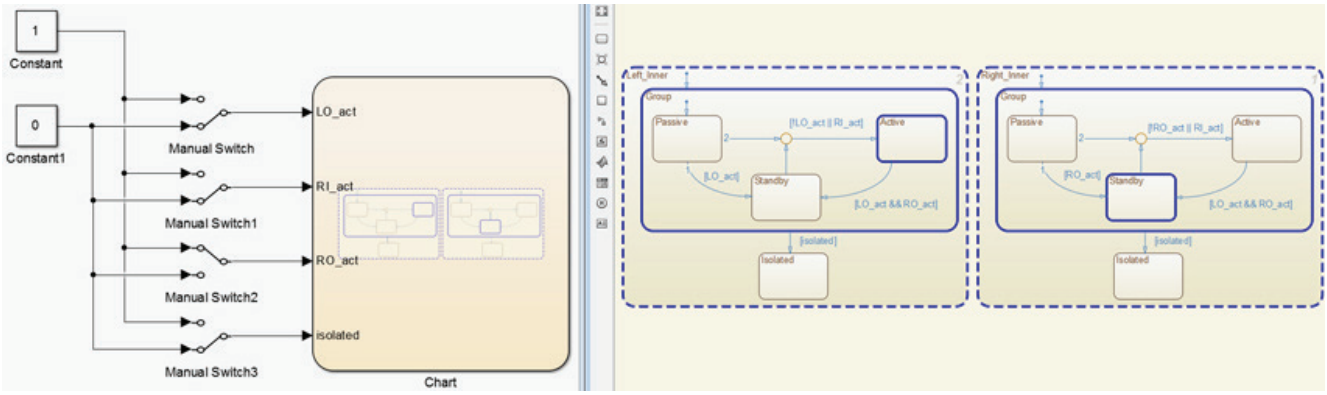


*Рисунок 7. Диаграмма состояний, указывающая на ошибку в проекте.*

Оказывается, что условие на переходе от состояния Active к Standby содержит ошибку. Поскольку мы привязали каждое условие к требованию, мы можем отследить это условие к соответствующему требованию и убедиться, что дефект происходит из документа с требованиями, а не из проекта (Рисунок 8).

**2.1.6 Right inner actuator in Standby mode**

- If the right inner actuator is in the Standby mode, and the right outer actuator is not in the Active mode or the right inner actuator is in the Active mode
  - Transition the right inner actuator to the Active mode.
  - Otherwise, keep the Right Inner actuator in the Standby mode.

*Рисунок 8. Дефект проекта отслеживается к документу с требованиями.*

Последняя строка должна выглядеть как “or the *left* inner actuator is in the Active mode.” (*или левый* внутренний исполнительный механизм находится в режиме Active)

Мы исправляем ошибку в документе с требованиями, изменяем условие, снова осуществляем симуляцию модели и проводим верификацию того, что система теперь ведет себя корректно в ответ на входные сигналы.

## Тестирование системы с новым компонентом

Обратите внимание, что после тестирования модели FDIR как отдельного компонента, мы готовы протестировать его в системной модели. Мы помещаем этот компонент в модель в виде ссылки на модель (Model block) с названием FDIR_application. После интеграции блока в системную модель мы можем продолжать работать над ним независимо от остальной системы, используя механизм ссылочной модели (Model reference) в Simulink (Рисунок 9).

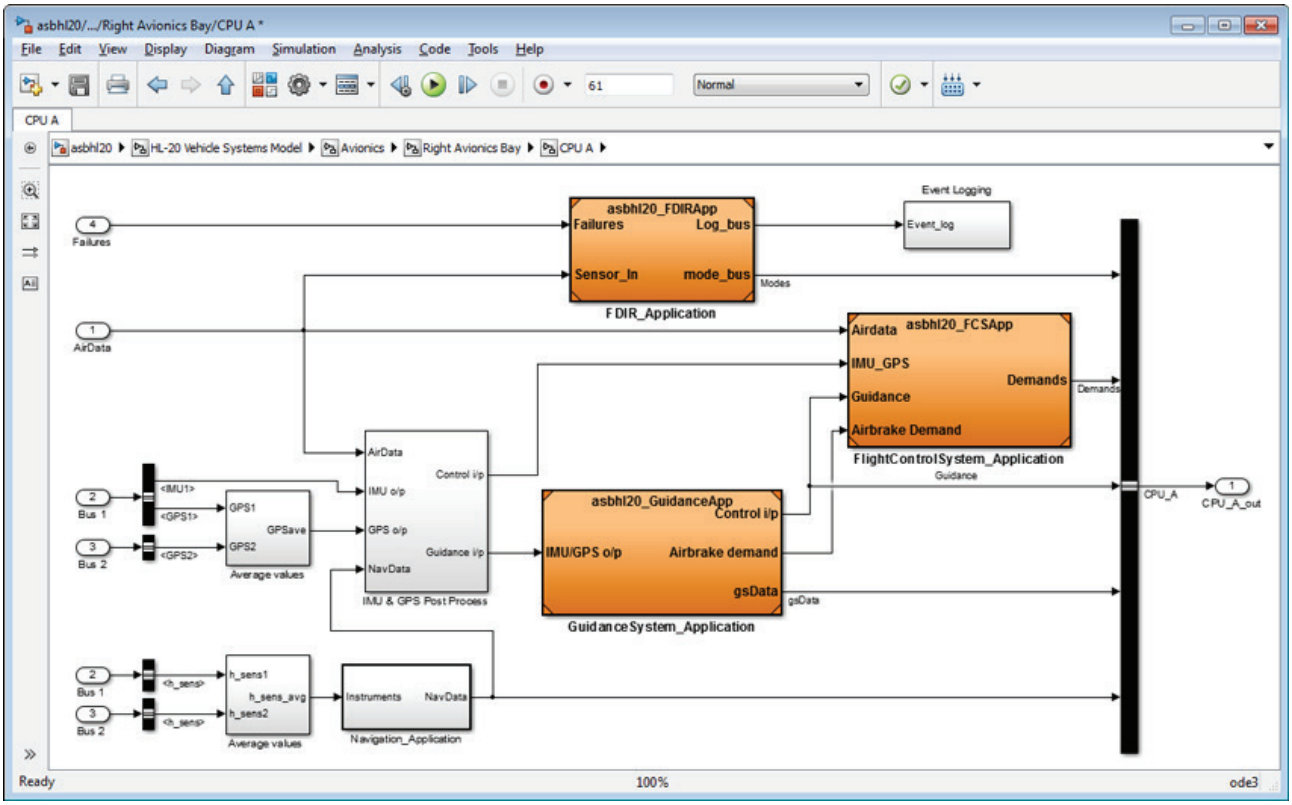


*Рисунок 9. Модель системного уровня, отображающая три ссылочных модели компонентов: систему управления полетом, логику FDIR и систему навигации.*

Мы осуществляем симуляцию модели системного уровня и визуализацию поведения компонента в диаграмме состояний, а также поведения всей системы с использованием FlightGear – инструмента для визуализации с открытым исходным кодом. Для тестирования этой системы мы настроили обвязку, которая вносит неисправности в систему исполнительного механизма, так чтобы мы могли осуществить верификацию того, что и компонент и система в целом ведут себя корректно (Рисунок 10).

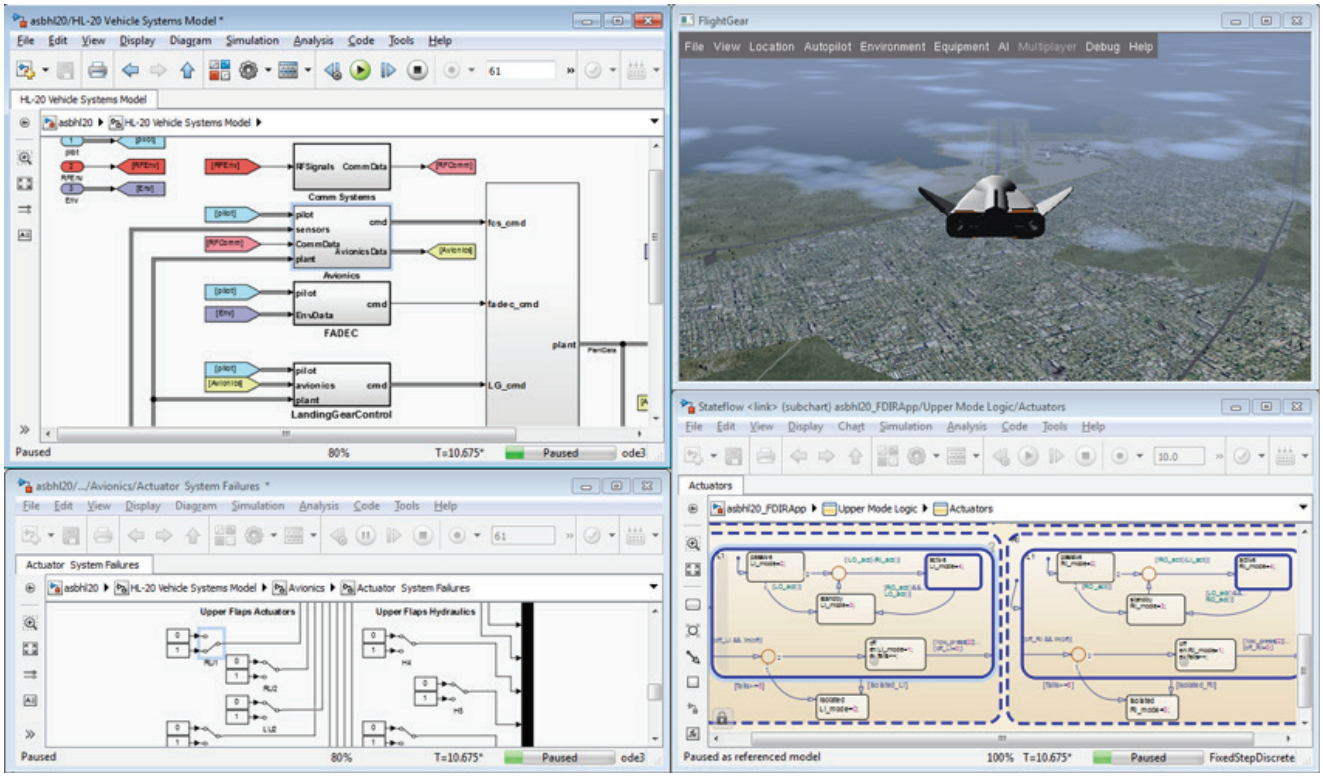


*Рисунок 10. Симуляция многодисциплинарной модели системного уровня.*

## Следующие шаги

На данном этапе мы создали компонент на базе требований, осуществили симуляцию и тестирование компонента, а затем подключили его к модели системного уровня для дополнительной симуляции и тестирования. Дополнительные шаги, которые можно предпринять для усовершенствования рабочего процесса моделирования и симуляции, включают в себя:

- Увеличение производительности симуляции с использованием Performance Advisor в Simulink
- Внедрение процесса формального тестирования и верификации с доказательством свойств системы, анализом покрытия и генерацией тестов
- Замена блоков Simulink подключениями к реальному оборудованию по мере того, как оборудование становится доступным

Какие бы дальнейшие шаги вы ни выбрали, ключевой аспект заключается в моделировании, симуляции и тестировании системы так часто и так рано, как это только возможно, для поиска и исправления дефектов на ранней стадии и сокращения общей стоимости разработки.

# ДЛЯ ЗАМЕТОК

## Дополнительная информация и контакты

**Информация о продуктах**
matlab.ru/products

**Пробная версия**
matlab.ru/trial

**Запрос цены**
matlab.ru/price

**Техническая поддержка**
matlab.ru/support

**Тренинги**
matlab.ru/training

**Контакты**
matlab.ru
**E-mail:** matlab@sl-matlab.ru
**Тел.:** +7 (495) 232-00-23, **доб.** 0609
**Адрес:** 115114 Москва,
Дербеневская наб., д. 7, стр. 8