Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Высшая школа программной инженерии

Работа допущена к защите
Директор ВШ ПИ
_____П.Д. Дробинцев
"___"_____ 2018г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Моделирование гибридных систем в средах визуального моделирования

По направлению *02.03.02 «Фундаментальная информатика и информационные технологии»*
по образовательной программе
02.03.02_ 02 «Информатика и компьютерные науки»

Выполнил
студент гр. 43504/6 М.В. Сергеев
Руководитель
д.т.н., проф. Ю.Б. Сениченков

Санкт-Петербург
2018

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
Институт компьютерных наук и технологий

Утверждаю
Директор ВШ ПИ
_____П.Д. Дробинцев
"___"_____ 2018г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы
студенту М.В. Сергееву гр. 43504/6

1. Тема: *Моделирование гибридных систем в средах визуального моделирования*
2. Срок сдачи работы 08.06.18.
3. Исходные данные к проекту (работе).
   1) Материалы по RMD: www.mvstudium.com
   2) Материалы по Modelica: www.modelica.org
   3) Материалы по UML: www.uml.org
4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов).
   1) UML как стандарт языков моделирования
   2) Машины состояний в различных средах моделирования
   3) Сравнительный анализ языков Modelica и MVL
   4) Конвертация моделей OpenModelica в RMD
   5) Набор моделей для проверки сконвертированных моделей
5. Перечень графического материала с точным указанием обязательных чертежей.

   Графический материал отсутствует.
6. Консультанты по проекту (с указанием относящегося к ним разделов проекта, работы).

1) Колесов Ю. Б. - гибридные системы в Modelica
2) Григуть Е. В. - моделирование в OpenModelica

Дата выдачи задания: _____ г.
Руководитель ВКР: _____д.т.н., проф.Ю.Б. Сениченков
Задание принял к исполнению _____ г.
Студент _____М.В. Сергеев

Реферат

На 53 с., , рис. 22

В данной выпускной квалификационной работе бакалавра изучены современные среды визуального моделирования и, на основе полученных данных, из-за потребности в создании новых открытых библиотек для среды визуального моделирования RMD, доказана возможность создания конвертора гибридных систем с языка Modelica на язык MVL, опираясь на стандарт UML и особенности реализаций гибридных систем в средах визуального моделирования OpenModelica и RMD.

В работе представлено описание реализации базового конвертора гибридных систем с языка Modelica на язык MVL, соответствующих сред визуального моделирования OpenModelica и RMD, и ряд тестовых моделей, которые показывают корректность сконвертированных моделей.

Ключевые слова: гибридные системы, среды визуальноого моделирования, точки переключения, конвертор, UML, Modelica, MVL, OpenModelica, RMD.

Abstract

53 pages , 22 figures

In this graduate work the modern visual modeling environments were studied and, on the basis of obtained data, because of the need to create new open libraries for the visual modeling environment RMD, the possibility of creating a converter of hybrid systems from the Modelica language to the MVL language, relying on the UML standard and features of hybrid system implementations in the visual modeling environments of OpenModelica and RMD was proved.

In this work the implementation of the basic converter of hybrid systems from the Modelica language to the MVL language in the corresponding visual modeling environments of OpenModelica and RMD, and a several test models that show the correctness of the converted models was described.

Keywords: hybrid systems, visual modeling environments, switching points, converter, UML, Modelica, MVL, OpenModelica, RMD.

Оглавление

Список обозначений

UML	Unified Modeling Language
MVL	Model Vision Language
RMD	Rand Model Designer
ДС	Динамическая система
ГС	Гибридная система
ДК	Динамический компонент
ПО	Программное обеспечение
НУ	Начальные условия

Введение

В современном мире существует потребность в различных устройствах для производства, анализа и управления. Большинство из этих устройств – это сложные технологические системы, состоящие из множества компонент, каждый из которых выполняет свою определенную функцию. Для создания подобных устройств необходимо предварительно доказать правильность и корректность работы будущего устройства и получаемых результатов. Основными подходами к решению данной проблемы являются численное и визуальное моделирование, выполняющееся на различной вычислительной технике и которое позволяет упростить разработку разнообразных устройств, путем создания электронной копии будущей системы и моделирования всех поведенческих аспектов данной системы.

Развитие вычислительной техники и появление различного ПО, стало причиной образования множества подобных друг другу подходов и сред моделирования предназначенных для создания моделей в различных сферах жизнедеятельности. Обилие различных подходов к моделированию стало проблемой для большинства программистов, занимающихся разработкой моделей, так как им потребовалось разбираться в особенностях синтаксиса и возможностей различных языков моделирования, в зависимости от поставленной задачи, что могло вызвать путаницу между подходами и требовало большое количество времени.

Проблема существования большого числа различных подходов к моделированию потребовала их систематизации и создания стандартизированного языка моделирования, объединяющего большинство хорошо зарекомендовавших себя подходов к моделированию. Таким языком стал UML.

Главной проблемой UML является отсутствие непрерывной составляющей моделирования, что не позволяло создавать модели, работающие в режиме реального времени. В связи с этим, параллельно с UML разрабатывались такие языки визуального моделирования как Modelica, Simulink, MVL, AnyLogic и т.д. – некоторые из которых опираются на стандарт UML и могут применяться для моделирования непрерывных, а так же гибридных (событийно-управляемых) систем. Каждый из этих языков имеет свои особенности в построении гибридных систем, характеризующиеся машиной состояний. Возникает проблема соответствия результатов моделирования систем, модели которых разработаны на разных языках моделирования, опираясь на описание гибридных систем в данных языках визуального моделирования.

Среды моделирования, использующие данные языки моделирования, как правило, в зависимости от времени появления содержат разное количество программных библиотек для решения тех или иных задач. Процесс написания и разработки библиотеки под определенную задачу – это трудоемкий и затратный по времени процесс. Поэтому, возникает проблема расширения функционала различных сред моделирования при помощи использования или конвертирования уже готовых открытых библиотек из других сред моделирования.

Глава 1

Обзор литературы и постановка задачи

1.1 Современные языки моделирования

Существует больше количество различных языков моделирования и соответствующих им сред визуального моделирования, часть из которых приведены в табл. 1.1. Каждая среда визуального моделирования, в зависимости от особенностей используемого языка, применяется для описания определенных видов систем [2], представленных в табл. 1.2.

Среда визуального моделирования RMD подходит для моделирования всех видов систем, что является значительным преимуществом

Таблица 1.1. Использование языков моделирования в соответствующих средах визуального моделирования

Языки моделирования	Среды моделирования
Язык среды Simulink	Simulink
Modelica	Dymola, OpenModelica
MVL	RMD, MvStudium
Язык среды AnyLogic	AnyLogic
UML	UML Studio

Таблица 1.2. Виды систем

Компонентные системы с «направленными» связями			Однокомпонентные системы		Компонентные системы с «физическими» связями		
Тип компонента							
динамический	статический		ДС	ГС	статический		динамический
ДК	ДС	ГС			ДС	ГС	ДК
-	Simulink	Simulink, Stateflow	Simulink	Simulink, Stateflow	Simulink, SimMechanics, SimPower systems ***	Simulink, Stateflow, SimPower systems ***	-
-	Dymola, OpenModelica	Dymola, OpenModelica	Dymola, OpenModelica	Dymola, OpenModelica	Dymola, OpenModelica	Dymola, OpenModelica	-
AnyLogic	AnyLogic	AnyLogic	AnyLogic	AnyLogic	-	-	-
RMD	RMD	RMD	RMD	RMD	RMD	RMD	RMD

по отношению к другим средам моделирования. На сегодняшний день, главным недостатком среды визуального моделирования RMD, по причине новизны, является отсутствие в среде достаточного количества открытых для пользователя библиотек компонентов и моделей, из-за чего пользователь вынужден работать в другой, более оснащенной библиотеками, среде моделирования. В связи с этим возникает проблема развития и распространения отечественной среды моделирования RMD путем пополнения среды новыми открытыми библиотеками.

Новые открытые библиотеки для среды визуального моделирования RMD можно разрабатывать путем ручного написания библиотек или при помощи программного преобразования исходного кода уже имеющихся открытых библиотек других языков моделирования при помощи конвертора. Основными преимуществами конвертора являются – значительная экономия времени созданиях новых библиотек и возможность пользователю сменить среду моделирования, сконвертировав свои проекты и наработки на язык среды RMD.

UML является языком разработки программных комплексов и, несмотря на то, что является стандартом, не имеет массового практического применения в моделировании, по ряду причин, одними из которых являются неудобность использования UML в сфере моделирования и отсутствие возможности создания непрерывных и событийно-управляемых (гибридных) систем. Следствием из этого является существование малого количества UML библиотек.

Языки моделирования Simulink и Modelica активно используются за рубежом в различных прикладных областях и часто пополняются новыми библиотеками для решения новых видов задач.

Что касается библиотек языка MVL для сред RMD и MvStudium – они разрабатывались отдельно друг от друга. Так, среда MvStudium является частью среды RMD и обладает небольшим количеством закрытых (коммерческих) библиотек, которые в основном используются для создания морских тренажеров фирмой TRANSAS и охранных систем фирмой Пентакон.

Новые заказчики хотят использовать уже разработанные модели и библиотеки (проблема конвертирования) и убедиться на примере уже решенных задач, что предлагаемые RMD технологии более эффективны. В настоящее время поступают заявки на использование RMD в различных областях (энергетика, логистика, социология), где уже

широко применялись другие среды визуального моделирования.

С учетом того, что требуется пополнить среду RMD новыми библиотеками, ставится задача сравнить язык MVL среды RMD с другими языками визуального моделирования по ряду критериев, необходимых для выбора исходного языка, на основе которого, будут получены новые библиотеки для среды RMD.

Критерии сравнения:

1. Соответствие стандарту UML;
2. Наличие объектно-ориентированного подхода;
3. Машины состояний;
4. Гибридные системы;
5. Типы данных;
6. Типы решаемых уравнений и систем уравнений;
7. Алгоритм и точность алгоритма поиска точки переключения в гибридных системах.

После выбора исходного языка – наиболее схожего с языком моделирования MVL, для автоматического преобразования библиотек из одного языка в другой, требуется разработать и протестировать на отдельных примерах простой базовый конвертор гибридных систем.

В главе 2 приведено сравнение между основными языками моделирования по некоторым критериям, на основе которого был выбран исходный язык для конвертирования – Modelica.

1.2 Уточненные требования к работе

Требуется сравнить различные среды моделирования по критериям соответствия стандарту UML, особенностям машин состояний и реализации гибридных систем. По полученным результатам сравнить языки визуального моделирования Modelica и MVL и показать возможность разработки конвертора гибридных систем моделей среды OpenModelica в модели среды RMD и, в случае возможности, реализовать его и протестировать по критерию наличия погрешности вычислений точек переключения в гибридных системах между исходной и сконвертированной моделью.

Глава 2

Моделирование событийно-управляемых систем

2.1 UML как стандарт моделирования

Метод абстракции – один из основных подходов в сфере разработки программного обеспечения, который заключается в описании требований к ПО и разделению программной части и деталей реализации. Данный подход разделяет основную задачу на ряд подзадач и предполагает использование диаграмм Entity-Relationship (Сущность-Связь), для моделирования различных аспектов программных систем, и диаграмм последовательностей для моделирования поведенческих аспектов систем [7].

Главными недостатками метода абстракции являются несогласованность между данными и поведением модели системы и несоответствие между реальным миром и моделью, а также несоответствие между моделью и реализацией [8].

В качестве решения этих недостатков была сформирована база для объектно-ориентированной парадигмы и целого ряда различных объектно-ориентированных языков, использующие различные подходы к моделированию [7].

Отсутствие единого стандарта моделирования стало причиной появления большого числа различных подходов к моделированию, разработчики которых считали, что только их подходы верны. Необходимо было разработать стандарт, на который бы опирались все разработчики программных систем и компонент, предназначенных для моделирования.

В конце 20-го века, Гради Буч, Джим Рамбо и Айвар Якобсон создали новый единый язык моделирования, под названием UML, объединивший большинство хорошо зарекомендовавших себя подходов к моделированию в различных прикладных областях, который в последствии стал общепринятым стандартом в сфере моделирования.

Основными задачами языка UML стали следующие [8]:

1. Замена языков моделирования, специфичных для определенных условий, языком общего назначения;

2. Моделирование и описание всех аспектов системы соответствующим стандарту UML образом;

3. Предоставление визуальных компонентов пользователю для лучшего понимания;

4. Повторное использование удачно зарекомендованных подходов к моделированию уже существующих языков моделирования, подходящих для моделирования определенных аспектов системы, и преобразование их в стандарт;

5. Использование стандарта UML для сравнения различных подходов к моделированию;

6. Согласование формализованного синтаксиса и стандартной нотации для моделирования систем.

Язык UML основывается на концепции абстрактного типа данных и предоставляет различные подъязыки для работы со структурными и поведенческими особенностями системы. Подъязыками являются различные виды диаграмм, которые используются в UML, для представления определенных аспектов систем, что делает UML универсальным средством описания как программных, так и деловых систем. Диаграммы дают возможность представить систему в таком виде, чтобы общая концепция модели была ясна пользователю и ее можно было легко перевести в программный код.

Основные типы UML диаграмм [6]:

1. Диаграмма вариантов использования (use case diagram);
2. Диаграмма активностей (activity diagram);
3. Диаграмма классов (class diagram);
4. Диаграмма объектов (object diagram);
5. Диаграмма состояний (statechart diagram);
6. Диаграмма последовательности (sequence diagram);
7. Диаграмма кооперации (collaboration diagram);
8. Диаграмма компонентов (component diagram);
9. Диаграмма развертывания (deployment diagram).

Большинство сред визуального моделирования основываются на пяти основных типах UML диаграмм – диаграммах активностей, диаграммах состояний, диаграммы компонентов, диаграммы классов и объектов (последние три типа диаграмм применяются для описания структуры моделей и взаимосвязи компонентов в ней, при наличии объектно-ориентированного подхода). В технических системах – структурная схема описания уровней иерархий модели представлена диаграммой компонентов.

Основное внимание в усилиях по стандартизации UML пока было соглашение по общепринятому стандарту обозначение для всех этих типов диаграмм, тогда как соглашение о формально определенной и точной семантике было отложено до следующего этапа стандартизации. Несмотря на это, UML используется как не официальный, но общепринятый стандарт моделирования среди разработчиков моделей различных типов.

2.1.1 Диаграммы классов и объектов

UML предоставляет диаграммы классов [8] (рис. 2.1) и объектов (экземпляров классов) для моделирования всех структурных аспектов системы. Эти диаграммы берутся из диаграмм Entity-Relationship

и предлагают средства для определения структуры объектов и возможной взаимосвязи. Классы представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Структурные отношения между классами могут быть описаны как ассоциации, агрегации, композиции и наследование. Объекты, как экземпляр класса, описываются их атрибутами, а также различными операциями, которые могут изменять состояние объекта [8].

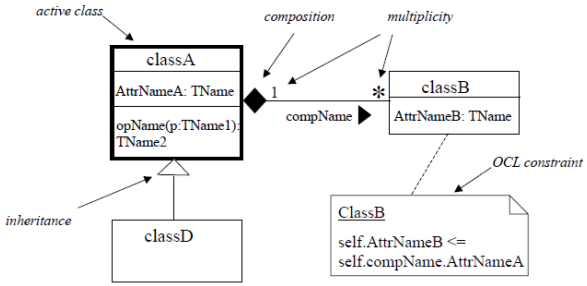

Рис. 2.1. Пример диаграммы классов

Диаграмма объектов - это снимок системы, отображающий взаимосвязь между объектами и их состояния в определенный момент времени.

2.1.2 Диаграммы состояний

Диаграммы состояний [8] (рис. 2.2) используется для описания и отображения машины состояний языка UML, предназначенной для описания поведения отдельных объектов с течением времени. Объекты имеют состояние, которое может меняться из-за реакции на полученные события (инициирующие переходы состояний). Такое событие может быть сигналом или событием вызова от другого объекта или временным сигналом, который заставляет объект изменять свое состояние. Таким образом, машины состояний используются для моделирования жизненных циклов объекта и обеспечения, так называемого, просмотра внутри объекта. Машина состояний в UML основана на графических картах состояний (поведений) Харелла и предлагает

такие элементы, как параллельные и последовательные составные состояния, историю состояний и состояния перехода для моделирования сложных поведений объекта [8].

UML использует графические карты состояний Харелла для описания дискретных систем, в то время как остальные языки моделирования, опирающиеся на стандарт UML, используют расширенный формализм машины состояний для непрерывных систем.

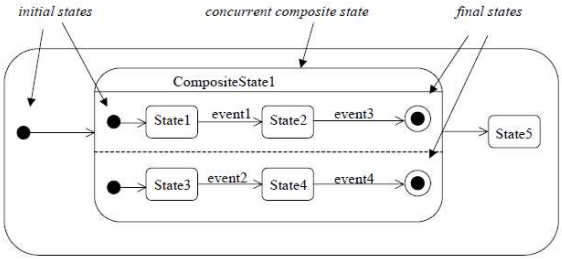

Рис. 2.2. Пример диаграммы состояний

Элементы диаграммы состояний:

- Состояние (state) - период в жизненном цикле объекта, в котором объект удовлетворяет определенному условию;

  Виды псевдосостояний:

  - Начальное состояние (start state) - состояние с которого начинается работа системы;
  - Конечное состояние (final state) - состояние на которого заканчивается работа системы.

- Действие (action) - непрерываемая на переходах между состояниями операция;

  Виды действий:

  - Входное действие (entry action) - действие, выполняющееся в момент входа в состояние;
  - Выходное действие (exit action) - действие, выполняющееся в момент выхода из состояния;

- Деятельность (activity) - выполнение в состоянии различных операций;
- Переход (transition) - отношение между двумя состояниями;
- Событие (event) - явление в системе, вызывающее определенные действия.

2.1.3 Диаграммы активностей

Описание, ориентированное на управление потоком, может быть представлено диаграммой активности [8] UML (рис. 2.3). Синтаксически диаграмма активности представляет собой особую форму конечного автомата, где состояния интерпретируются и помечены действиями. В отличие от обычной машины состояния, изменение состояния автоматически запускается, когда выполнение деятельности было завершено [8].

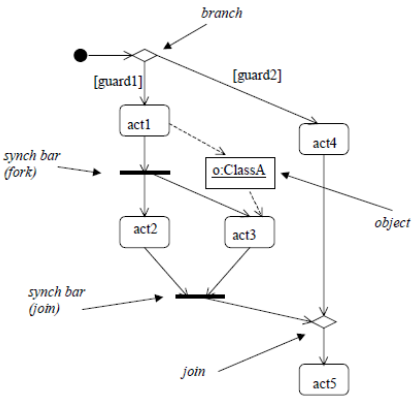

Рис. 2.3. Пример диаграммы активности

Диаграммы активностей не связаны с определенными объектами и представляют в основном процедурный, возможно параллельный поток внутри системы. Объекты могут быть содержать как входные / выходные параметры, между которыми могут быть указаны дополнительные объектные потоки между состояниями [8].

2.2 Машины состояний

Любой язык моделирования, предназначенный для создания событийно-управляемых систем, обладает своими особенностями, а именно различным описанием машин состояний. В данной главе будут рассмотрены значимые различия общеизвестных языков моделирования.

Машина состояний (state machine) — это математическая модель, представляющая из себя конечный автомат и предназначенная для описания реальных или конструируемых объектов, меняющих свое поведение при наступлении определенных событий. Каждая машина состояний представляет собой диаграмму состояний. Основные элементы машины состояний (диаграммы состояний) описаны в разделе 2.1.2.

Сравнительный анализ машин состояний различных языков моделирования представлен в табл. 2.1. Машины состояний, опирающиеся на стандарт UML, как, например, у языка MVL, могут применяться для описания гибридных систем. Отсутствие в языке MVL параллельных состояний связано с особенностями языка, а отсутствие истории состояний – с большими накладными расходами для обработки и хранения данных о значениях переменных в момент выхода из состояния.

Несмотря на то, что язык UML является общепринятым стандартом в моделировании, язык Simulink не опирается на него. В качестве замены классической машины состояния UML, Simulink использует специальное расширение, которое имеет свои особенности и правила в представлении машины состояний, называющийся Stateflow.

2.3 Сравнение языков MVL и Modelica

2.3.1 Соответствие стандарту UML

Процесс конвертации из модели языка Modelica в язык MVL среды RMD должен проходить в два этапа как показано на рис. 2.4, где Этап 1 – это сопоставление исходной модели на языке Modelica со стандартом UML, а Этап 2 – это создание модели языка MVL среды RMD.

Так как структуры моделей среды RMD полностью основаны на основных элементах диаграмм языка UML, то следует поставить в со-

Таблица 2.1. Сравнительный анализ машин состояний

Критерии	UML	MVL	Modelica	Simulink	AnyLogic
Опирается на стандарт UML	да	да	да	нет	да
Машина состояний	да	да	языковой эквивалент	да (Stateflow)	да
Активности в состояниях	дискретные	гибридные	гибридные	гибридные	гибридные
Параллельные состояния	да	нет	да	да	да (дискретные)
История состояний	да	нет	да	да	да

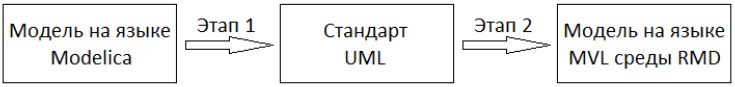

Рис. 2.4. Этапы конвертации модели

ответствие элементы языка Modelica альтернативным элементам языка UML, чтобы определить возможность преобразования библиотек из языка Modelica в язык среды RMD.

В языке Modelica используются следующие особенности языка UML [12]:

- Диаграммы классов

  Элементы диаграммы классов:

  – Типы классов: Модель (Model), Блок (Block), Соединитель (Connector);

  – Атрибуты классов: Переменные (Variables), Параметры (Parameter);

  – Отношения классов: Наследование (Inheritance), Композиция (Composition).

  Пример диаграммы класов [12] изображен на рис. 2.5

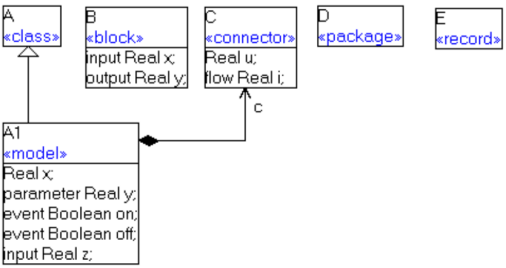

Рис. 2.5. Диаграмма классов в UML

Ниже, в листинге 2.1, представлен языковой эквивалент диаграммы классов рис. 2.5 на языке Modelica:

Листинг 2.1. Языковой эквивалент диаграммы классов

```
package UML_H annotation(UMLH(ClassDiagram=
"<umlhclass><name>...");
class A annotation(UMLH(classPos=[31,53]));
end A;
model A1 annotation(Icon(Text(extent=...,
string="A1"));
annotation(UMLH(classPos=[31,146]));
extends A;
event Boolean on;
event Boolean off;
Real x;
input Real z;
parameter Real y;
C c;
...
end A1;
...
connector C annotation(UMLH(classPos=[192,54]));
Real u;
flow Real i;
end C;
...
end UML_H;
```

- Диаграммы компонентов

  Типы, соединяющие компоненты:

  - Соединяющие переменные;
  - Скалярные переменные;
  - Скалярные входные/выходные переменные;
  - Смешенные соединяющие типы.

  Пример диаграммы компонентов [12] изображен на рис. 2.6

  Ниже, в листинге 2.2, представлен языковой эквивалент диаграммы компонентов рис. 2.6 на языке Modelica:

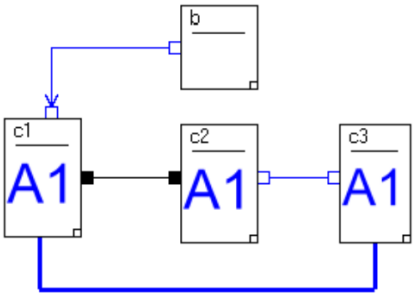

Рис. 2.6. Диаграмма компонентов в UML

Листинг 2.2. Языковой эквивалент диаграммы компонентов

```
model System
annotation(CompConnectors(CompConn(label=
"label2", points=
[-81,52; -81,43; -24,43; -24,51])));
UML_H.A1 c1 annotation(extent=[-87,72; -74,52]);
UML_H.A1 c2 annotation(extent=[-57,71; -44,51]);
UML_H.A1 c3 annotation(extent=[-30,71; -18,51]);
UML_H.B b annotation(extent=[-57,91; -44,77]);
equation
// connection type 1:
connect(c1.c,c2.c)annotation(points=
[-74,62;-57,62]);
// connection type 2:
c2.y=c3.y annotation(points=
[-44,62; -30,62]);
// connection type 3:
b.y=c1.z annotation(points=
[-57,84; -79,84; -79,72]);
// connection type 4 (mixture of type 1 and 2):
connect(c1.c,c3.c) annotation(label="label2");
c1.x=c3.x annotation(label="label2");
end System;
```

- Диаграммы состояний

  Типы состояний:

  - Начальное состояние (Initial states);
  - Конечное состояние (Final states);
  - Атомарное состояние (Atomic states);
  - Нормальное состояние (Normal states).

  Пример диаграммы состояний [12] изображен на рис. 2.7

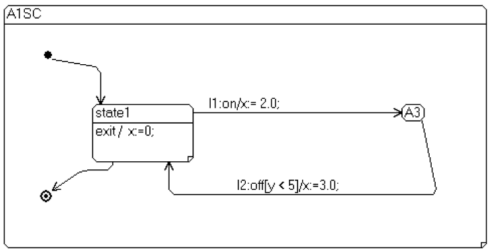

Рис. 2.7. Диаграмма состояний в UML

Ниже, в листинге 2.3, представлен языковой эквивалент диаграммы состояний рис. 2.7 на языке Modelica:

Листинг 2.3. Языковой эквивалент диаграммы состояний

```
model A1
statechart
state A1SC extends State annotation(extent=
[-88,86; 32,27]);
state State1
extends State;
exit action x:=0; end exit;
end State1;
State1 state1 annotation(extent=
[-66,62; -41,48]);
State A3 annotation(extent= ...);
State I5(isInitial=true);
State F7(isFinal=true);
```

```
transition I5->state1 end transition
annotation(points=[-76,73;-64,71; -64,62]);
transition l1:state1->A3 event on action
x:= 2.0;
end transition annotation(points= ...);
transition l2:A3->state1 event off guard y < 5
action x:=3.0;
end transition annotation ...;
transition state1->F7
end transition annotation ..;
end A1SC;
end A1;
```

2.3.2 Объектно-ориентированный подход

Оба языка моделирования являются объектно-ориентированными, что позволяет создавать многокомпонентные модели, основываясь на диаграммах классов и объектов.

Подробное о сравнение представлений объктно-ориентированного подхода языков Modelica и MVL приведено в статье [5].

2.3.3 Гибридные системы

Гибридная (событийно-управляемая или кусочно-непрерывная) система — это математическая модель, схожая с машиной состояний, но расширенная для непрерывных систем, предназначенная для описания объектов, меняющих свое непрерывное поведение при наступлении определенных событий [2].

Факторы, обусловливающие гибридное поведение:

1. Совместное функционирование непрерывных и дискретных объектов

2. Мгновенные качественные изменения в непрерывном объекте

3. Изменение состава системы

Для описания гибридных систем, RMD использует расширенные, для использования в непрерывном времени, диаграммы состояний

языка UML [2, 9]. В то же время, язык Modelica использует собственную концепцию, которая в явном виде не соответствует стандарту UML, и использует упрощенный подход в описании гибридных систем, а именно использует текстовый эквивалент записи карт состояний состояний UML, расширенных для непрерывных систем, через специальный оператор “if” [9]. На рис. 2.8 изображены визуальные представления гибридных (кусочно-непрерывных) систем языков RMD и Modelica.

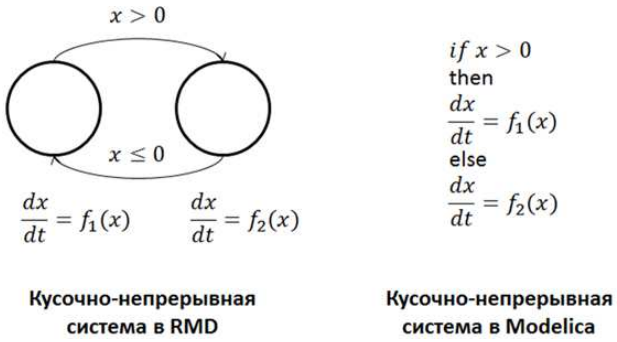

Рис. 2.8. Гибридные системы в RMD и Modleica

Особенностью языка Modelica является то, что процесс анализа гибридных систем (диаграмм состояний) происходит на этапе компиляции, в то время как в языке MVL среды RMD на этапе выполнения [2].

В случае гибридного времени, необходимо правильно найти точку переключения, при нахождении которой, модель завершает вычисление в текущем состоянии и переходит в следующее. Алгоритмы нахождения точки переключения описаны в разделе 2.3.6.

2.3.4 Типы данных

В табл. 2.2 показано сопоставление между основными типами данных языка Modelica и RMD.

Также в обоих языках моделирования присутствуют пользовательские типы. В рамках однокомпонентных систем, пользовательские типы можно легко представить в виде стандартных типов.

Таблица 2.2. Сопоставление типов данных

Modelica	Альтернатива в RMD
Real	double
Integer	integer
String	string
Boolean	boolean
Real[] Integer[]	vector[]
Boolean[]	array of boolean
String[]	array of string

2.3.5 Типы решаемых уравнений и систем уравнений

Язык MVL среды визуального моделирования RMD берет из языка Modelica физическую составляющую моделирования непрерывных систем [3], а именно оба языка позволяют решать различные виды уравнений и их систем.

Виды решаемых систем уравнений [1]:

1. Системы линейных алгебраических уравнений;
2. Системы нелинейных алгебраических уравнений;
3. Системы обыкновенных дифференциальных уравнений;
4. Системы алгебро-дифференциальных уравнений.

В обоих языках MVL и Modelica для решения различных видов уравнений по умолчанию используется математический пакет DDASL.

2.3.6 Алгоритм поиска точки переключения

Язык среды RMD использует метод деления отрезка пополам для нахождения точки переключения [2] (корня уравнения $f(x) = 0$, где $f(x)$ - непрерывная функция на отрезке $[a, b]$ и $f(a) * f(b) < 0$) с заданной точностью ϵ

Функция $f(x)$ основывается на условных выражениях в блоках "if" и "when", которые характеризуют наличие гибридного времени. Пример: Условное выражение $x < 2$ представляется в виде $x - 2 < 0$ после чего решается уравнение $f(x) = x - 2$. Исходным отрезком $[a, b]$ может являться отрезок $[0, 3]$.

Алгоритм нахождения точки переключения в MVL [14]:

1) Вычислить $c = \frac{a+b}{2}$;

2) a) Если $f(a) * f(c) > 0$, то $a = c$ (рис. 2.9, а);
   b) Если $f(a) * f(c) < 0$, то $b = c$ (рис. 2.9, б);

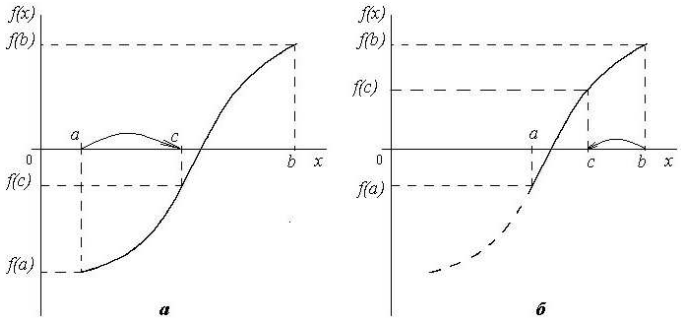

Рис. 2.9. Метод деления отрезка пополам

3) Продолжать, начиная с шага 1, до тех пор, пока не $|b - a| < \epsilon$, и тогда любое значение из отрезка $[a, b]$ можно будет считать корнем с погрешностью ϵ.

Modelica использует схожий подход к поиску точки переключения и использует Illinois алгоритм для решения дифференциально-алгебраических уравнений [10] (программа DASRT, входящая в пакет DDASL).

В языке Modelica функция $f(y)$ (рис. 2.10) называется функцией пересечения нуля, которая, так же как и в языке MVL, представляет непрерывные условные события вида $y > 53$ в виде функции $f(y) = y - 53$ и ищет корень с заданной точностью ϵ, который является временем возникновения переключения и, соответственно, события $y > 53$ [11].

Алгоритм нахождения точки переключения в Modelica [13]:

1) Вычислить $c = \frac{a*f(b)-b*f(a)}{f(b)-f(a)}$;

   a) Если $f(b) * f(c) < 0$, то $a = c$;

   b) Если $f(b) * f(c) > 0$, то $b = c$;

2) Продолжать, начиная с шага 1, до тех пор, пока не $f(c) < \epsilon$, и тогда c можно будет считать корнем с погрешностью ϵ.

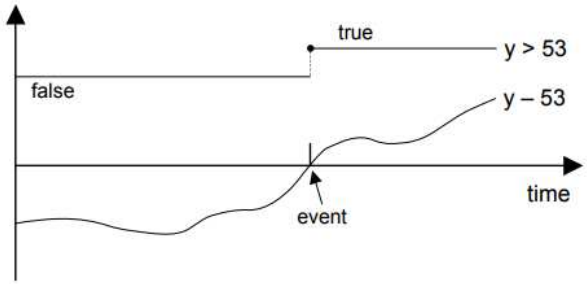

Рис. 2.10. Функция пересечения

На примере модели прыгающего мячика можно сравнить результаты алгоритмов поиска точки переключения в языках Modelica и MVL в момент первых нескольких касаний мячика земли, где y – координата мячика по вертикали, а Vy – скорость мячика и $\epsilon = 1e - 6$. В табл. 4.1 главы 4 представлены результаты поиска точки переключения в языках Modelica и MVL соответствующих сред визуального моделирования OpenModelica и RMD, а также погрешность результатов относительно друг друга.

Также, помимо модели прыгающего мяча, в главе 4 представлены примеры различных гибридных систем, взятых из книги [4], и приведен сравнительный анализ результатов конвертирования моделей с языка Modelica среды OpenModelica в язык MVL среды RMD с выявлением погрешности (разности результатов) вычислений в точках переключения, при том, что НУ не являются точками переключения. При тестировании используется, общий для двух языков моделирования, пакет решения алгебро-дифференциальных уравнений DDASL с фиксированным шагом интегрирования 1e-2 и погрешностью вычис-

лений 1e-6. Реализация исходных моделей для конвертирования на языке Modelica в среде OpenModelica представлена в разделе 3.2.

Как видно из результатов тестирования конвертора гибридных систем (глава 4), погрешность вычислений в точках переключения сред моделирования OpenModelica и RMD крайне мала, что говорит о корректности сконвертированных моделей и о том, что каждую модель среды OpenModelica можно преобразовать в соответствующую модель RMD.

2.4 Вывод

Получив результаты по всем критериям сравнения между языком Modelica и языком MVL среды RMD, можно сделать вывод о возможности преобразования библиотек и создания конвертора гибридных систем, метод конвертирования которого будет заключаться в построчном переводе исходного кода языка Modelica в исходный код MVL среды RMD с дополнением недостающих конструкций, присутствующих в языке Modelica.

Глава 3

Конвертор моделей с языка Modelica на язык MVL

3.1 Разработка конвертора

3.1.1 Архитектура системы

Разработанный на языке C# базовый конвертор гибридных систем использует в себе методы построчного анализа и преобразования исходного кода языка Modelica среды визуального моделирования OpenModelica в язык MVL среды RMD.

Структура конвертора состоит из пяти основных классов:

1. Main

   Функции класса Main:

   1) Считывание исходного файла с расширением ".mo" и преобразование его в список строк;
   2) Преобразования имен функций и других элементов языка Modelica в имеющийся эквивалент языка среды RMD, приведенны в табл. 3.1;
   3) Формирование списка исходных данных по табл. 2.2;

Таблица 3.1. Список доступных для конвертирования функций и друих элементов языка Modelica

Modelica	эквивалент MVL
acos	arccos
asin	arcsin
atan	arctg
log	ln
log10	lg
sinh	sh
cosh	ch
tanh	th
transpose	transp
Комментарии к строкам //comment	–comment
Производные вида der(x)= 0	$\frac{dx}{dt} = 0$
Производные вида der($a + b$)= 0	$x = a + b$; $\frac{dx}{dt} = 0$
reinit(a, b)	$a = b$

4) Формирование списка уравнений;

5) Формирование списка дискретных событий блоков “if” и “when”.

2. ArrayConv

   Функции класса ArrayConv: Обработка массивов и векторов, которые встретились на этапе формирования списка исходных данных в классе Main.

3. WhenStatement и IfStatement

   Функции классов WhenStatement и IfStatement: Сопоставление списка уравнений с определенными дискретными событиями.

4. Converter

   Функции класса Converter: Упорядочивает по шаблону все списки данных, полученных из других классов, и создает конечный файл с расширением “.mvl”.

3.1.2 Интерфейс системы

Интерфейс пользователя представляет собой простое программное окно (рис. 3.1), в котором, по нажатию кнопки “Convert”, можно выбрать исходный файл с расширением “.mo” на языке Modelica. После выбора файла произойдет автоматическое конвертирование файла в файл с тем же названием, но с расширением “.mvl”. Полученный файл создается в той же директории, где лежит исходный файл, и его следует скомпилировать средствами языка MVL среды RMD, а затем проверить работоспособность модели.

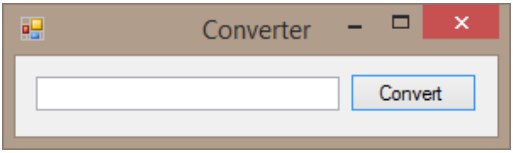

Рис. 3.1. Интерфейс конвертора

3.2 Разработка тестов

Для тестирования работы конвертора гибридных систем, на языке Modelica в среде визуального моделирования OpenModelica, были разработаны примеры гибридных систем. Модель, описанная в подразделе 3.2.1 является базовым и простым демонстрационным примером работы гибридных систем. В подразделах 3.2.1, 3.2.2 и 3.2.3 приводятся примеры гибридных систем, описание которых приводится в книге [4], характеризующие различные переходные процессы в некоторых видах систем.

В данных моделях на языке Modelica, условные выражения “if” описывают условия вхождения в определенное состояние и закрепляет за данным состоянием свою систему уравнений, в то время как выражения “when” отвечают за дискретные изменения переменных на переходах между состояниями.

Описание моделей и результаты конвертирования и вычислений погрешности описаны в соответствующих названиям моделей разделах главы 4. Также, в главе 4, приведены графические сравнения результатов моделирования двух сред моделирования.

3.2.1 Модель прыгающего мяча

```
model BouncingBall
 parameter Real Vx=10;
 Real x(start=0);
 Real y(start=0);
 Real Vy(start=5);
equation
 der(x) = Vx;
 der(y) = Vy;
 der(Vy)=-9.81;
 when  y<=0 and Vy<0 then
  reinit(Vy,-Vy*0.8);
 end when;
end BouncingBall;
```

3.2.2 Модель релейной системы

```
model PerProc
 parameter Real b1=3;
 parameter Real b2=5;
 parameter Real c=4;
 parameter Real k=5;
 parameter Real T=2;
 Boolean s1, s2, s3;
 Real x(start=-4);
 Real y(start=2);
equation
 s1= (x<-b1) and (y>0) or (x<-b2) and (y<0);
 s2= (-b1<x) and (x<b2) and (y>0) or (-b2<x)
 and (x<b1) and (y<0);
 s3= (x>b2) and (y>0) or (x>b1) and (y<0);
 if s1 then
  der(x) = y;
  der(y) = (-y/T)-(k/T)*(-c);
 elseif s2 then
  der(x) = y;
  der(y) = (-y/T);
 elseif s3 then
```

```
   der(x) = y;
   der(y) = (-y/T)-(k/T)*c;
   else
   der(x)=0;
   der(y)=0;
  end if;
end PerProc;
```

3.2.3 Модель с логическим управлением

```
model LogUpr
 parameter Real b1=3;
 parameter Real b2=5;
 parameter Real c=1;
 Real f(start=4);
 Real w(start=4);
 Boolean s1,s2,s3,s4,s5;
equation
 s1=-b1<f and f<b1;
 s2=f>0 and w>0;
 s3=f<0 and w<0;
 s4=f>0 and w<0;
 s5=f<0 and w>0;
 when s4 then
  reinit(w,-b2);
 elsewhen s5 then
  reinit(w,b2);
 end when;
 if s1 or s4 or s5  then
  der(f)=w;
  der(w)=0;
 elseif s2 then
  der(f)=w;
  der(w)=-c;
 elseif s3 then
  der(f)=w;
  der(w)=c;
 else
  der(f)=0;
```

```
   der(w)=0;
  end if;
end LogUpr;
```

3.2.4 Модель с переменной структурой

```
model SysPeremStruct
 parameter Real c=2;
 parameter Real k=1;
 parameter Real k1=3;
 Real x(start=1);
 Real x1(start=0);
 Real y(start=-1);
equation
 der(x)=y;
 x1=y+c*x;
 if x1*x>0 then
  der(y)+k1*k*x=0;
 elseif x1*x<0 then
  der(y)-k1*k*x=0;
 else
  y=0;
 end if;
end SysPeremStruct;
```

Глава 4

Тестирование конвертора

4.1 Модель прыгающего мяча

Дана гибридная модель прыгающего мяча, которого кинули с определенной высоты с заданной горизонтальной и вертикальной составляющей скорости (НУ).

Численные результаты и погрешности вычислений при помощи языков Modelica и MVL представлены в табл 4.1. Графическое сравнение результатов моделирования в средах моделирования OpenModelica и RMD приведено на рисунках 4.1 и 4.2.

Таблица 4.1. Анализ результатов поиска точки переключения

Точка переключения		Modelica	RMD	Разность результатов
1 (НУ)	y	9.9931e-11	-7.1100675e-7	7.11106681e-7
	Vy	-5	-5.0000014	0.0000014
2	y	9.98447e-11	-3.8632853e-7	3.86428375e-7
	Vy	-4	-4.0000003	0.0000003
3	y	9.99234e-11	-6.0513862e-7	6.05238543e-7
	Vy	-3.2	-3.2000009	0.0000009
4	y	9.99786e-11	-4.7690257e-7	4.77002549e-7
	Vy	-2.56	-2.5600002	0.0000002

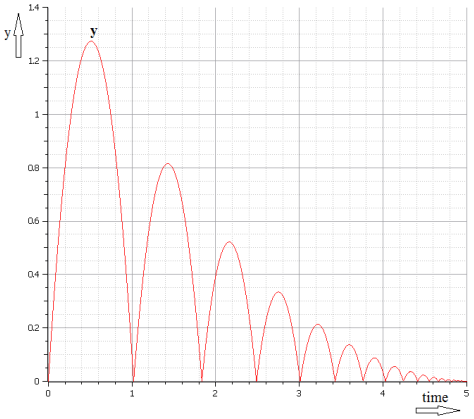

Рис. 4.1. Модель прыгающего мяча в OpenModelica

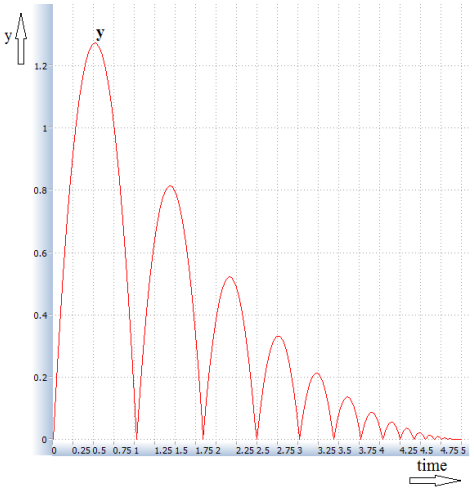

Рис. 4.2. Модель прыгающего мяча в RMD

4.2 Модель релейной системы

Общая система уравнений динамики системы представлена в виде

$$\frac{dx}{dt} = y, \frac{dy}{dt} = -\frac{y}{T} - \frac{k}{T}F(x),$$

где $F(x)$ - релейная характеристика [4], которую по рис. 4.3 можно описать следующим образом:

если $y = \frac{dx}{dt} > 0$, то

$$F(x) = \begin{cases} -c & \text{при } x < -b_1, \\ 0 & \text{при } -b_1 < x < b_2, \\ +c & \text{при } x > b_2; \end{cases}$$

если $y = \frac{dx}{dt} < 0$, то

$$F(x) = \begin{cases} +c & \text{при } x > b_1, \\ 0 & \text{при } -b_2 < x < b_1, \\ -c & \text{при } x > -b_2; \end{cases}$$

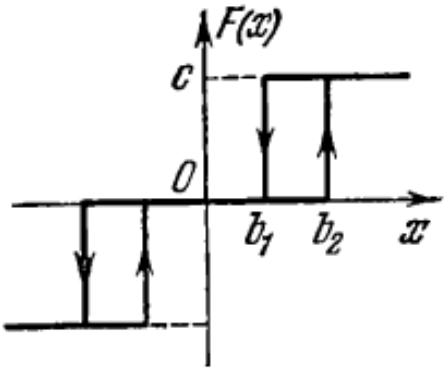

Рис. 4.3. Характеристика $F(x)$

Численные результаты и погрешности вычислений при помощи языков Modelica и MVL представлены в табл 4.2. Графическое сравнение результатов моделирования в средах моделирования OpenModelica и RMD приведено на рисунках 4.4 и 4.5.

Таблица 4.2. Анализ результатов поиска точки переключения

Точка переключения		Modelica	RMD	Разность результатов
1 (НУ)	t	0	0	0
	x	-4	-4	0
	y	2	2	0
2	t	0.303210	0.305	-0.00179
	x	-2.999999	-2.9919	-0.008099
	y	4.532106	4.54595	-0.013844
3	t	4.5874	4.5875916	-0.000192
	x	5	5.00022	-0.00022
	y	0.532106	0.532057	0.000049
4	t	5.309984	5.3099988	-0.000015
	x	2.999999	2.99878	0.001219
	y	-5.693729	-5.69535	0.001621
5	t	7.734848	7.73453	0.000318
	x	-5	-5.000002	0.000002
	y	-1.693729	-1.693858	0.000129
6	t	8.588631	8.588431	0.0002
	x	-2.999999	-2.99942	-0.000579
	y	5.844105	5.84491	-0.000805
7	t	10.895508	10.89511	0.000398
	x	5	5.000012	-0.000012
	y	1.844105	1.84422	-0.000115
8	t	11.767278	11.767611	-0.000333
	x	2.999999	2.99585	0.004149
	y	-5.873600	-5.8787	0.0051

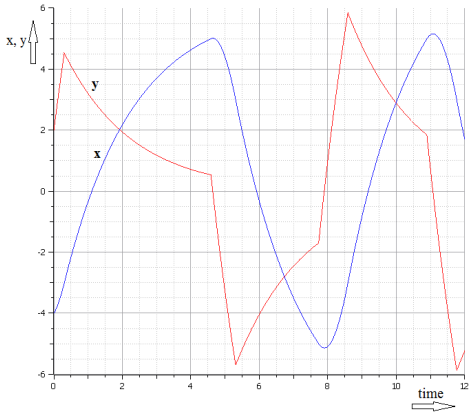

Рис. 4.4. Модель релейной системы в OpenModelica

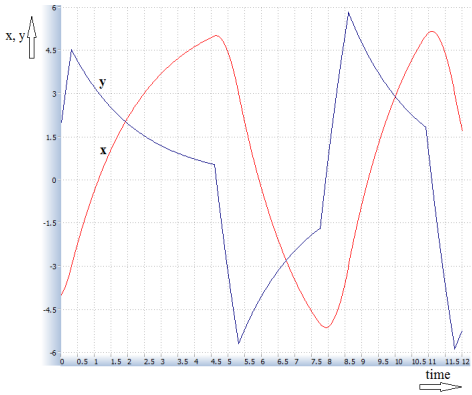

Рис. 4.5. Модель релейной системы в RMD

4.3 Модель с логическим управлением

Общая система уравнений идеальной системы управления представлена в виде

$$\frac{d\varphi}{dt} = \omega, \frac{d\omega}{dt} = cF(\varphi, \omega),$$

где $F(\varphi, \omega)$ - логическая управляющая функция [4], которую по рис. 4.6 можно описать следующим образом:

$$F(\varphi,\omega) = \begin{cases} -1 & \text{при } \varphi > 0 \text{ и } \omega > 0, \\ 0 & \text{при } -b_1 < \varphi < b_1, \\ 0 & \text{при } \varphi < 0 \text{ и } \omega > 0, \\ 0 & \text{при } \varphi > 0 \text{ и } \omega < 0, \\ 1 & \text{при } \varphi < 0 \text{ и } \omega < 0; \end{cases}$$

Существует зависимость ω от φ и ω, представленная в виде:

$$\omega = \begin{cases} -b_2 & \text{при } \varphi > 0 \text{ и } \omega < 0, \\ b_2 & \text{при } \varphi < 0 \text{ и } \omega > 0; \end{cases}$$

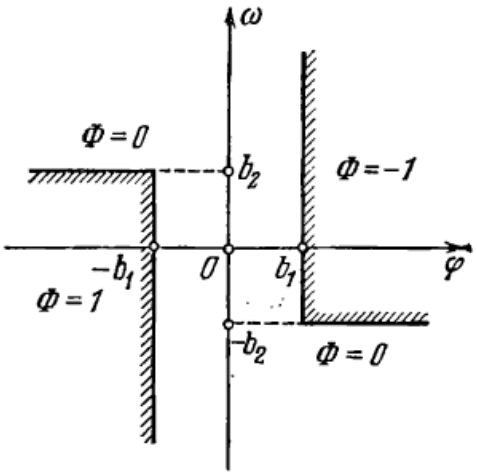

Рис. 4.6. Логическая управляющая функция $F(\varphi, \omega)$

Численные результаты и погрешности вычислений при помощи языков Modelica и MVL представлены в табл 4.3. Графическое сравнение результатов моделирования в средах моделирования OpenModelica и RMD приведено на рисунках 4.7 и 4.8.

Таблица 4.3. Анализ результатов поиска точки переключения

Точка переключения		Modelica	RMD	Разность результатов
1 (НУ)	t	0	0	0
	f	4	4	0
	w	4	4	0
2	t	4	4	0
	f	11.999999	12	-0.000001
	w	-1.001367e-10	6.10352e-7	-6.104521e-7
3	t	7	7	0
	f	-2.999999	-2.99998	-0.000019
	w	-5	-5	0
4	t	12	12	0
	f	-15.5	-15.5	0
	w	1.002362e-10	6.10351e-7	-6.102508e-7
5	t	15.7	15.7	0
	f	3	3	0
	w	5	5	0
6	t	20.7	20.7	0
	f	15.5	15.5	0
	w	-1.002895e-10	1.19336e-13	-1.004088e-10
7	t	24.4	24.4	0
	f	-3	-3	0
	w	-5	-5	0
8	t	29.4	29.4	0
	f	-15.5	-15.5	0
	w	1.001225e-10	-1.36058e-13	9.998644e-11

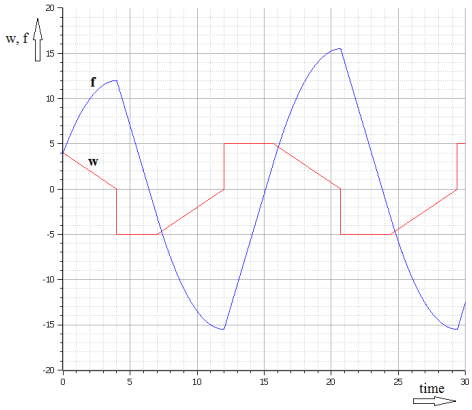

Рис. 4.7. Модель с логическим управлением в OpenModelica

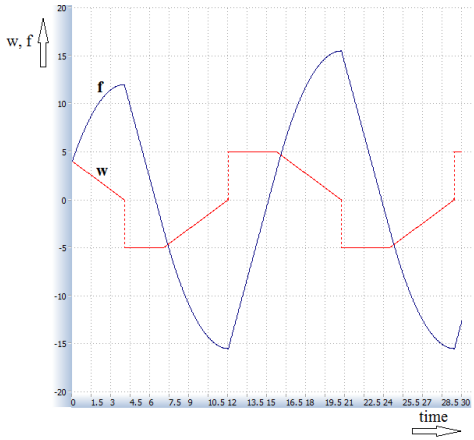

Рис. 4.8. Модель с логическим управлением в RMD

4.4 Модель с переменной структурой

Имеется логическое переключающее устройство [4], представленной на рис. 4.9, которое в зависимости от размеров и знака входной величины x подключает либо звено 1, либо звено 2.

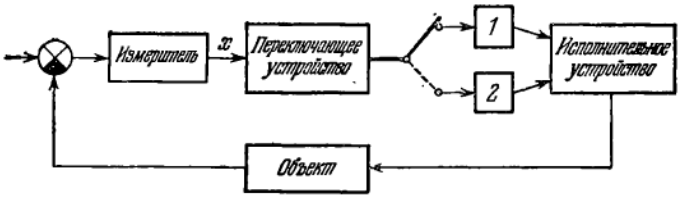

Рис. 4.9. Логическое переключающее устройство

Общая система уравнений динамики замкнутой системы при включенном и выключенном звене представлена в виде

$$\frac{d^2x}{dt^2} + k_1 kx = 0 \text{ при } x_1 x > 0,$$

$$\frac{d^2x}{dt^2} - k_1 kx = 0 \text{ при } x_1 x < 0,$$

$$x_1 = \frac{dx}{dt} + cx.$$

Численные результаты и погрешности вычислений при помощи языков Modelica и MVL представлены в табл 4.4. Графическое сравнение результатов моделирования в средах моделирования OpenModelica и RMD приведено на рисунках 4.10 и 4.11.

Таблица 4.4. Анализ результатов поиска точки переключения

Точка переключения		Modelica	RMD	Разность результатов
1 (НУ)	t	0	0	0
	x	1	1	0
	y	-1	-1	0
2	t	0.192533	0.19250	0.000033
	x	0.755928	0.755976	-0.000048
	y	-1.511857	-1.511787	-0.000070
3	t	0.952878	0.952844	0.000034
	x	-1.3229108e-10	1.3056e-5	-0.000013
	y	-0.75593	-0.755933	-0.000003
4	t	2.354606	2.354658	-0.000052
	x	-0.285714	-0.285707	-0.000007
	y	0.571429	0.571444	-0.000015
5	t	4.516685	4.516658	0.000027
	x	0.107991	0.108005	-0.000014
	y	-0.215982	-0.215959	-0.000023
6	t	6.678767	6.67876	0.000007
	x	-0.040816	-0.040819	0.000003
	y	0.081632	0.081632	0
7	t	8.840844	8.840849	-0.000005
	x	0.015426	0.015430	-0.000004
	y	-0.030853	-0.030852	-0.000001
8	t	11.003076	11.003057	0.000019
	x	-0.00583	-0.005831	0.000001
	y	0.011661	0.011664	-0.000003

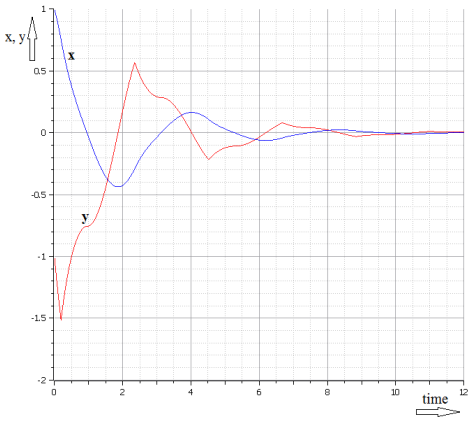

Рис. 4.10. Модель с переменной структурой в OpenModelica

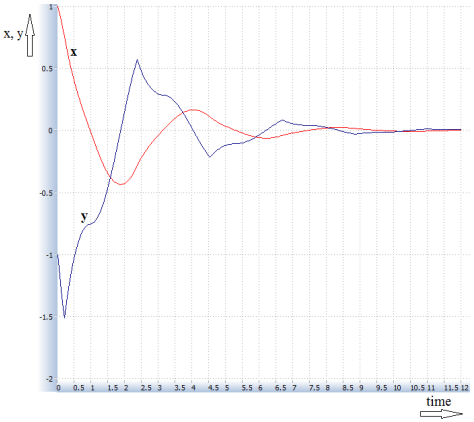

Рис. 4.11. Модель с переменной структурой в RMD

Заключение

В рамках выпускной квалификационной работы бакалавра были произведены исследования среди различных языков и сред визуального моделирования по критериям соответствия стандарту UML, особенностям машин состояний, реализации гибридных систем, с целью описания возможности создания конвертора гибридных систем. На основе данных сравнения языков моделирования Modelica и MVL был разработан базовый конвертор гибридных систем с языка Modelica среды визуального моделирования OpenModelica на язык MVL среды визуального моделирования RMD. Для проверки правильности работы конвертора был разработан ряд тестов, представляющие собой различные гибридные системы на языке Modelica. Полученные результаты сконвертированных моделей среды RMD сравнивались с результатами исходных моделей среды OpenModelica, по критерию величины погрешности вычислений точек переключений в гибридных системах.

Результатом разработки базового конвертора гибридных систем стала возможность автоматически, программным способом, пополнять среду визуального моделирования RMD новыми открытыми библиотеками и моделями, основывающиеся на событийно-управляемых системах (также на дискретных и непрерывных), с целью увеличения возможностей среды RMD.

Для создания полноценного конвертора любых видов систем, необходимо будет объединить полученный конвертор гибридных систем с конвертором объектно-ориентированной составляющей языка Modelica, описанного в [5]. Дополнительно будет разработан единый пользовательский интерфейс для простой и понятной работы с конвертором. В конечном счете, будет разработан программный продукт,

который можно будет использовать в различных компаниях, для конвертирования моделей и увеличения популярности среды визуального моделирования RMD.

Литература

[1] *Колесов Ю. Б., Сениченков Ю. Б.* Моделирование систем. Динамические и гибридные системы. — СПб.: Изд-во БХВ, 2006. — Р. 224 с.

[2] *Колесов Ю. Б., Сениченков Ю. Б.* Математическое моделирование гибридных динамических систем. — СПб.: Изд-во Политехн. ун-та, 2014. — Р. 236 с.

[3] *Колесов Ю. Б., Сениченков Ю. Б.* Объектно-ориентированное моделирование в среде Rand Model Designer 7. — Изд-во Проспект, 2016. — Р. 255 с.

[4] *Попов Е. П.* Теория линейных систем автоматического регулирования и управления. — Москва: Изд-во Наука, 1988. — Р. 301 с.

[5] *Крышин А. А.* Объектно-ориентированный подход в современных средах моделирования: Tech. rep.: Санкт-Петербургский политехнический университет Петра Великого, 2018.

[6] *Booch G., Rumbaugh J., Jacobson I.* The Unified Modeling Language Usere Guide. — Москва: Изд-во ДМК, 2006. — Р. 496 с.

[7] *Engels G., Groenewegen L.* Object-oriented modeling: A roadmap. — 2000.

[8] *Engels G., Heckel R., Sauer S.* Uml - a universal modeling language? // *Application and Theory of Petri Nets 2000.* — 2000.

[9] Hybrid systems. preliminary comparative analysis of modelica and model vision language / Y. B. Kolesov, Y. B. Senichenkov, A. Urquia,

C. Martin-Villalba // *Университетский научный журнал.* — 2014. — no. 8.

[10] *Mao G., Petzold L. R.* Efficient integration over discontinuities for differential-algebraic systems // *Computers and Mathematics with Applications.* — 2002. — no. 43.

[11] *Lundvall H., Fritzson P., Bachmann B.* Event handling in the open-modelica compiler and runtime system: Tech. rep.: Linkoping University, 2008.

[12] *Nytsch-Geusen C.* The use of the uml within the modelling process of modelica-models: Tech. rep.: Technische Universitat Berlin, 2007.

[13] Regula falsi method algorithm and flowchart. https://www.codewithc.com/regula-falsi-method-algorithm-flowchart/.

[14] Метод деления отрезка пополам. http://eco.sutd.ru/Study/Informat/mpd.html.