пользователя среда обработки, что является необходимым условием нормального функционирования системы высокой чёткости изображения. Данная технология обладает большей гибкостью, поэтому они подходят для обработки видеосигнала с высоким разрешением. FPGA легко модернизировать и потому нет риска, что система быстро устареет.

Сочетание на одной платформе ARM процессора, предназначенного для выполнения алгоритмов обработки видеосигнала, и матрицы FPGA, для построения связующей логики, является оптимальным решением. Применение FPGA позволяет разгрузить DSP — матрица посчитает SAD в 10 раз быстрее. Это освободит процессор для других задач. На FPGA выполняется процесс фильтрации изображения, т.к. вычисления требуют выполнения многочисленных вычислительных операций в секунду. В итоге мы получаем высокоскоростную обработку и передачу видеосигнала.

Совокупность использования всех вышеперечисленных методов и технологий позволяет нам решить поставленную в самом начале задачу. Достижение поставленной цели происходит максимально рациональным путём, при этом относительно невысокая стоимость, по сравнению с другими технологиями представленными на рынке, и оптимальное расходование ресурсов вычислительной техники, так же является доказательством того, что предложенные выше решения являются оптимальными.

Литература

1. Плотников С.В. Сравнение методов обработки сигналов в триангуляционных измерительных системах // Автометрия № 6 1995 с. 58-63.

2. Р.В. Хемминг. Цифровые фильтры. Перевод с английского В.И. Ермишина / Под ред. Профессора А.М. Трахтмана. – М.: Советское Радио, 1980. – 224 с.

Е.П.Варламова

ВЕРИФИКАЦИЯ В ТЕСТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

(Поволжский государственный университет
телекоммуникаций и информатики)

С момента появления первых программ требовалось проверять их на правильность. Причем не просто удостовериться, что программа работает на конечном числе тестов, а уметь формально доказывать, что ее поведение соответствует спецификации.[1]

В настоящее время программное обеспечение (ПО) все больше реализует сложное поведение. Чем критичнее для бизнеса программа, тем дороже обходятся дефекты в ней.

Основная проблема в развитии программных систем состоит в том, что проверка правильности реализуемой программной системы чрезвычайно сложна. При разработке такой системы обычно основное время уходит не столько на написание кода, сколько на его анализ и отладку.[2]

На практике часто используются методы валидации и верификации, т.е. проверки программного обеспечения на корректность реализации поставленной задачи путем сравнения с требуемыми свойствами.

Поддержка этих процессов даже относительно простыми инструментами, не говоря уже о методах и инструментах с серьезной математической базой, в настоящее время недостаточна.

Ввиду возрастания размеров и сложности систем, становится важным обеспечение процесса верификации систем с использованием методов и инструментов, которые облегчают автоматический анализ корректности, так как, например, ручная верификация может быть настолько же ошибочна, как и сама программа. Одним из таких инструментов является верификатор.

Таким образом, целью данной работы является построение верификатора бизнес-процессов для тестирования ПО.

Одной из задач проекта является изучение тестируемого ПО.

В работе рассматривается представитель небанковской кредитной организации, а именно, депозитарий, участник рынка ценных бумаг, осуществляющий депозитарную деятельность.

Объектом исследования является функционал обработки поручений на участие/отмену участия в корпоративных действиях. Бизнес-процесс представляет собой этапы, которые проходит поручение на пути своего исполнения в системе. Задачей верификатора будет отследить корректность этого пути. Эта задача важна для бизнес-функции тестируемого ПО, так как ошибки здесь критичны и дорого стоят. Предполагается проведение проверок с помощью генерации различных исходных данных работы ПО. Таким образом, будет осуществляться прогон тестовых сценариев.

Для описания требований к ПО вводятся формализмы различных темпоральных логик и описываются алгоритмы проверки моделей для спецификаций, выраженных в этих темпоральных логиках. Исходные данные генерируются в соответствии с требованиями, разработанными специально для программного продукта. Результатом работы верификатора служит заключение о корректности работы ПО. Также предполагается проверка на негативные тестовые сценарии, которые, в свою очередь, не приводят к положительному заключению о работе программы.

В работе используется синтетический метод верификации, который совмещает в себе формальный метод (Model Checking, проверка на модели) и динамический метод (тестирование).

Model Checking - это автоматизированный метод, который для заданной модели поведения системы с конечным числом состояний и логического свойства, записанного в подходящем логическом формализме (обычно в темпоральной логике), проверяет справедливость этого свойства в данной модели. Про-

верка моделей может применяться для верификации как аппаратуры, так и программ.

Необходимыми ресурсами для реализации проекта являются: алгоритмы работы ПО (на их основе будет выполняться моделирование), требования к алгоритмам, среда разработки и математический аппарат.

Основная идея метода состоит в запуске алгоритмов, исполняемых компьютером, для проверки корректности систем. При использовании этого подхода пользователь вводит описание модели системы (возможное поведение) и описание спецификации требований (желаемое поведение), а верификацию проводит машина. Если ошибка найдена, инструментальное средство (верификатор) предоставляет контрпример, показывающий при каких обстоятельствах ошибка может быть обнаружена. Контрпример представляет собой сценарий, в котором модель ведет себя нежелательным образом. Это позволяет пользователю обнаружить ошибку и исправить спецификацию модели перед тем, как продолжить верификацию. Укрупненная схема проверки моделей приведена на рис. 1.

Метод проверки моделей имеет такой математический фундамент как логика, теория автоматов, структуры данных, алгоритмы на графах.

В качестве дополнительной функции верификатора предусматривается получение первичного набора тестовых сценариев для проведения тестирования на реальной системе (как негативные, так и положительные тестовые случаи).

Ожидаемыми результатами от практического применения разработки проекта являются:

- внедрение практики создания верификаторов в процесс тестирования ПО поможет улучшить качество выпускаемых продуктов;
- повышение квалификации специалистов по тестированию за счет вовлечения в процесс верификации тербований, составления тестовых сценариев и анализа результатов тестирования;
- применение Model Checking позволяет упростить построение модели программы по ее спецификации по сравнению с традиционными подходами к программированию;
- модель программы, пригодная к верификации, строится уже на этапе проектирования, что позволяет избежать грубых ошибок в начале разработки.

Применение автоматного подхода и созданного верификатора при разработке программ позволяет получить более надежное программное обеспечение по сравнению с традиционным подходом. Запуск программы на определенных входных данных, а также проверка различных сценариев выполнения, позволяют достаточно быстро (по сравнению с другими методами поиска ошибок) убедиться в корректности разработанного ПО. Возможность верификатора создавать тестовые наборы поможет упростить процесс написания тестов и освободит время для других важных задач.

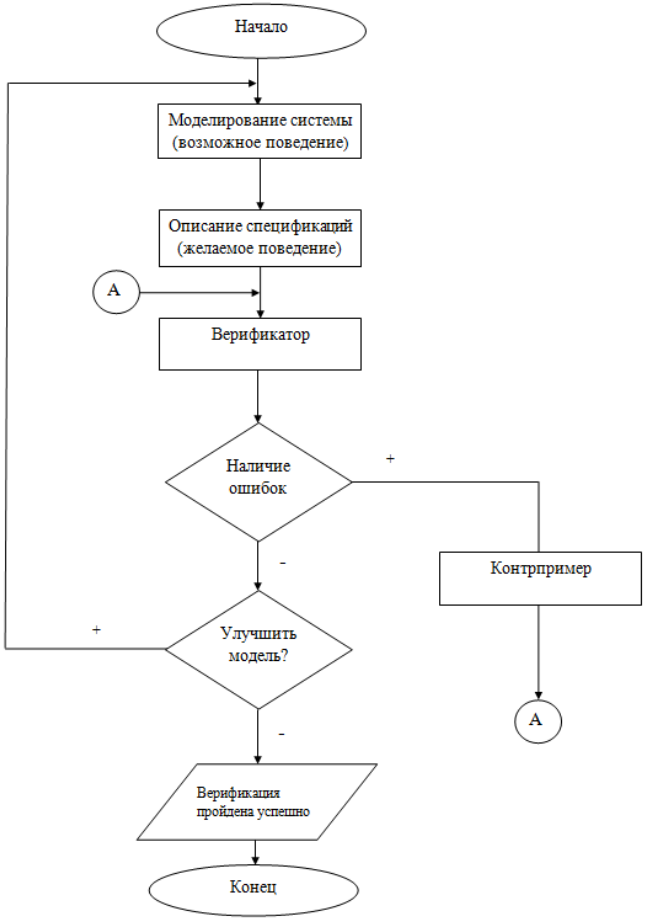

Рис. 1. Укрупненная схема проверки моделей

Литература

1. Егоров К.В, Шалыто А.А. Разработка верификатора автоматных программ: статья-СПб.:Научно-технический вестник, 2008.-177с.

2. Вельдер С.Э, Лукин М.А, Шалыто А.А, Яминов Б.Р. Верификация автоматных программ: учебное пособие.– СПб.: Наука, 2011. – 244 с.