На правах рукописи

Новиков Евгений Михайлович

РАЗВИТИЕ МЕТОДА КОНТРАКТНЫХ СПЕЦИФИКАЦИЙ ДЛЯ ВЕРИФИКАЦИИ МОДУЛЕЙ ЯДРА ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX

Специальность 05.13.11 –
математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва
2013

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институте системного программирования Российской академии наук.

Научный руководитель:	доктор физико-математических наук, профессор Петренко Александр Константинович
Официальные оппоненты:	Горбунов-Посадов Михаил Михайлович, доктор физико-математических наук, старший научный сотрудник, заведующий отделом информационных технологий Федерального государственного бюджетного учреждения науки Института прикладной математики им. М.В. Келдыша Российской академии наук
	Гринкруг Ефим Михайлович, кандидат технических наук, доцент кафедры управления разработкой программного обеспечения Федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский университет «Высшая школа экономики»
Ведущая организация:	Федеральное государственное бюджетное учреждение науки Научно-исследовательский институт системных исследований Российской академии наук

Защита диссертации состоится 31 октября 2013 г. в 16:00 на заседании диссертационного совета Д 002.087.01 при Федеральном государственном бюджетном учреждении науки Институте системного программирования Российской академии наук по адресу: 109004, Москва, ул. Александра Солженицына, д. 25, конференц-зал (комната 110).

С диссертацией можно ознакомиться в библиотеке Федерального государственного бюджетного учреждения науки Института системного программирования Российской академии наук.

Автореферат разослан 26 сентября 2013 г.

Ученый секретарь
диссертационного совета
канд. физ.-мат. наук /Прохоров С.П./

Общая характеристика работы

Актуальность темы

На сегодняшний день ядро операционной системы (ОС) Linux является одной из самых востребованных, больших и динамично развивающихся программных систем. Ядро ОС Linux используется повсеместно на большом количестве различных аппаратных платформ, начиная от встроенных систем и персональных компьютеров и заканчивая интернет-серверами и суперкомпьютерами. Размер ядра составляет более 16 млн строк кода. Процесс разработки ядра ОС Linux обладает уникальными особенностями. В подготовке новых версий ядра принимают участие более 1 000 разработчиков из более 200 организаций, рассредоточенных по всему миру. Каждая новая версия ядра ОС Linux включает около 10 тыс. изменений.

По своей архитектуре ядро ОС Linux является монолитным. Сердцевина ядра состоит из подсистем управления процессами, памятью и межпроцессорным взаимодействием, подсистем поддержки различных аппаратных платформ и т.д. Сердцевина ядра допускает динамическую загрузку модулей, что позволяет расширить набор ее функций. В настоящее время в состав ядра ОС Linux входит около 4 тыс. модулей, из которых большая часть является драйверами устройств. Помимо драйверов устройств, к числу модулей ядра относятся реализации различных файловых систем, сетевых протоколов, аудиокодеков и т.д.

Высокая практическая значимость ядра ОС Linux определяет строгие требования к его надежности и производительности. Ввиду того, что ядро имеет достаточно большой размер и развивается высокими темпами, контролировать выполнение этих требований вручную достаточно затруднительно.

Исследования показали, что более 85% ошибок в ядре ОС Linux сосредоточены в модулях. Это обусловлено тем, что, во-первых, модули составляют большую часть всего ядра. Во-вторых, сердцевина ядра активно используется разными модулями, а потому ошибки в ней выявляются достаточно быстро. Кроме того, разработчики модулей ядра ОС Linux, как правило, не являются экспертами в устройстве ядра.

Данная работа нацелена на автоматизацию обнаружения нарушений правил корректного использования программного интерфейса сердцевины ядра ОС Linux в модулях. Эти нарушения являются критическими, поскольку они могут привести к ненадежной работе и снижению производительности всего ядра, а также и ОС в целом. Результаты предшествующего анализа изменений в ядре ОС Linux показали, что данные нарушения достаточно частые. Они являются источником 15% от числа всех ошибок, или 50% от числа ошибок, которые не связаны с нарушениями спецификаций аппаратного обеспечения, сетевых протоколов, аудиокодеков и т.д.

Для поиска нарушений правил корректного использования программного интерфейса сердцевины ядра ОС Linux в модулях в данной работе предлагается применять передовые методы и инструменты верификации, которые в настоящее время активно развиваются в университетах и научно-исследовательских институтах по всему миру. Среди этих инструментов нет однозначного лидера, так как они используют разные техники и нацелены на разные классы ошибок. Поэтому в долгосрочной перспективе важно иметь возможность использовать различные инструменты верификации.

Сами по себе инструменты верификации не способны искать ошибки в модулях ядра ОС Linux – они позволяют решать задачу достижимости для программ на языке Си. Для того, чтобы свести задачу обнаружения точек нарушения правил корректного использования программного интерфейса сердцевины ядра в модулях к задаче достижимости, нужно:

- разработать контрактные спецификации, которые формальным образом описывают обязательства между сердцевиной ядра и модулями, то есть каким образом должны быть реализованы элементы программного интерфейса сердцевины ядра и какое использование данных элементов в модулях является корректным;

- инструментировать исходный код верифицируемых модулей таким образом, чтобы контрактные спецификации стали частью Си-программы, для которой можно решать задачу достижимости при помощи соответствующих инструментов.

Кроме того, следует учитывать особенности процесса разработки ядра ОС Linux по сравнению с ядрами других ОС. Разработчики постоянно модифицируют программный интерфейс сердцевины ядра и его реализацию. Эти изменения могут привести к потере согласованности контрактных спецификаций. В большинстве случаев при этом падает качество верификации, в частности, могут быть пропущены ошибки. Поэтому в случае ядра ОС Linux поддержка согласованности контрактных спецификаций является критически важной задачей.

Существующие методы контрактных спецификаций не учитывают особенности процесса разработки ядра ОС Linux, что ограничивает возможности их использования при проведении верификации модулей ядра ОС Linux. Таким образом, тема развития метода контрактных спецификаций, которой посвящена диссертация, является актуальной.

Цель и задачи работы

Цель работы — развитие метода контрактных спецификаций, который учитывает особенности процесса разработки ядра ОС Linux и обеспечивает возможность проверки правил корректного использования программного интерфейса сердцевины ядра в модулях посредством различных инструментов верификации.

Для достижения цели работы были поставлены следующие задачи:

а) Провести анализ существующих методов контрактных спецификаций.

б) Провести анализ и составить каталог наиболее критичных правил корректного использования программного интерфейса сердцевины ядра ОС Linux.

в) Разработать метод контрактных спецификаций, обеспечивающий следующие возможности:

1) описание в виде спецификаций всех правил из составленного каталога;

2) применение различных инструментов верификации для автоматизированной проверки модулей ядра ОС Linux на предмет выполнения требований спецификаций;

3) автоматизацию контроля корректности и согласованности спецификаций в условиях изменения программного интерфейса.

г) Разработать инструментарий, реализующий предлагаемый метод.

д) Оценить реализацию метода на практике, дать оценку области применимости метода.

Научная новизна работы

Научной новизной обладают следующие результаты работы:

- *новый метод* описания в виде контрактных спецификаций правил корректного использования программного интерфейса сердцевины ядра ОС Linux, основанный на предложенном *новом аспектно-ориентированном расширении языка Си*;

- *новый метод* автоматизированного инструментирования исходного кода модулей ядра ОС Linux на основе контрактных спецификаций, позволяющий применять для его проверки различные инструменты верификации;

- *новый метод* автоматизированного контроля корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса сердцевины ядра ОС Linux.

Практическая значимость

На основе предлагаемого в работе метода контрактных спецификаций были разработаны и использованы на практике спецификации для 34 правил корректного использования программного интерфейса сердцевины ядра ОС Linux.

Инструментарий, реализующий предлагаемый метод контрактных спецификаций, был включен в состав системы верификации модулей ядра ОС Linux, разработанной в рамках проекта Linux Driver Verification. По результатам верификации модулей нескольких версий ядра ОС Linux было выявлено более 125 ошибок, признанных разработчиками ядра. Коллектив проекта Linux Driver Verification постоянно осуществляет мониторинг новых версий ядра ОС Linux, что позволяет как дорабатывать контрактные спецификации, так и находить новые ошибки по мере развития ядра.

Разработанные контрактные спецификации и инструментарий были использованы в процессе подготовки эталонного набора задач достижимости для измерения сравнительных характеристик инструментов верификации. Данный набор используется в рамках ежегодных мероприятий ETAPS/Competition on Software Verification. В будущем планируется обновлять данный набор на основе результатов верификации модулей новых версий ядра ОС Linux.

Результаты работы могут быть полезны для решения задач поддержки крупных развивающихся систем, прежде всего при разработке контрактных спецификаций для формального описания требований к реализации и использованию программных интерфейсов, которые изменяются с течением времени. Такие контрактные спецификации могут быть использованы, например, в ходе проведения тестирования и верификации программ для обеспечения их надежности и производительности.

Доклады и публикации

Основные положения работы докладывались на 8 конференциях и семинарах:

- Научно-практическая конференция «Актуальные проблемы программной инженерии» (г. Москва, 2009 г.);
- 52-я научная конференция МФТИ «Современные проблемы фундаментальных и прикладных наук» (г. Долгопрудный, 2009 г.);
- 5-й весенний/летний коллоквиум молодых исследователей в области программной инженерии (SYRCoSE, г. Екатеринбург, 2011 г.);
- 6-я летняя школа Microsoft Research (г. Кембридж, Великобритания, 2011 г.);
- 8-я конференция разработчиков свободных программ (г. Обнинск, 2011 г.);
- 2-й международный семинар Linux Driver Verification в рамках 5-го международного симпозиума по внедрению формальных методов, верификации и валидации (LDV Workshop, ISoLA, г. Ираклион, о. Крит, Греция, 2012 г.);
- семинар ИСП РАН (г. Москва, 2012 г.);
- Европейская конференция по программной инженерии, совмещенная с Симпозиумом по основам программной инженерии ACM SIGSOFT (ESEC/FSE, г. Санкт-Петербург, 2013 г.).

По теме диссертации автором опубликовано 15 работ и получено 2 свидетельства о государственной регистрации программы для ЭВМ. Полный список работ приведен в конце автореферата. Первые 7 из них опубликованы в изданиях из перечня ВАК.

Структура и объем диссертации

Диссертация состоит из введения, 4 глав, заключения, списка литературы (132 наименования) и 2 приложений. Основной текст диссертации (без приложений и списка литературы) занимает 123 страницы.

Краткое содержание диссертации

Во **введении** обосновывается актуальность темы работы, определяются цель и задачи работы, раскрывается ее практическая значимость.

В **первой главе** приводится обзор существующих методов контрактных спецификаций. Рассматриваются возможности различных методов контрактных спецификаций в отношении способности описания требований к реализации и использованию программных интерфейсов, а также их проверки с помощью различных инструментов динамического и статического анализа. Показывается, что классические контрактные спецификации применяются для верификации реализации программных интерфейсов, тогда как в данной работе необходимо разработать метод для проверки правил корректного использования программного интерфейса сердцевины ядра ОС Linux в модулях посредством различных инструментов верификации. По результатам обзора формулируются требования к новому методу контрактных спецификаций.

Во **второй главе** предлагается новый метод контрактных спецификаций.

Первая часть посвящена анализу наиболее критичных правил корректного использования программного интерфейса сердцевины ядра ОС Linux. По результатам данного анализа составлен каталог правил, который приводится в приложении А. В данном каталоге представлена классификация правил корректного использования программного интерфейса по типам (выделение и освобождение ресурсов, работа в атомарном контексте, использование различных примитивов синхронизации в одном потоке и т.п.) и по видам, определяемым по логической взаимосвязанности элементов программного интерфейса. Для каждого вида правил в каталоге представлены:

- список выявленных нарушений;

- описание релевантных элементов программного интерфейса сердцевины ядра ОС Linux;

- неформальные формулировки правил.

Во второй части рассматривается подход аспектно-ориентированного программирования (АОП) для языка Си, который лежит в основе предлагаемого в работе метода контрактных спецификаций. Даются определения понятий АОП, которые используются в дальнейшем в работе:

- *аспект* — специальный модуль, который позволяет описывать дополнительную, так называемую ортогональную функциональность программ отдельно от модулей, в которых описана основная функциональность;

- *аспектная компоновка* — процесс связывания аспектов с целевой программой;

- *аспектный компоновщик* — инструмент, который позволяет выполнить аспектную компоновку целевой программы.

Также во второй части представлены способы описания аспектов, которые используются для различных языков программирования; влияние особенностей языка программирования Си и процесса сборки Си-программ на реализацию АОП; особенности практического применения реализации АОП для языка Си, в том числе для задания контрактных спецификаций; существующие реализации АОП для языка Си.

Показывается, что существующие реализации АОП для языка Си предоставляют не все средства, которые необходимы для задания контрактных спецификаций для верификации модулей ядра ОС Linux с помощью разных инструментов верификации:

- они не учитывают особенности языка программирования Си и процесса сборки Си-программ;

- предоставляемые средства являются недостаточно выразительными для

задания спецификаций многих правил корректного использования программного интерфейса сердцевины ядра ОС Linux;

- использование существующих реализаций АОП на практике затруднено, поскольку они либо не поддерживают язык программирования Си с расширениями компилятора GCC, который используется для разработки ядра ОС Linux, либо не выдают на выходе инструментированный исходный код на Си, что требуется для последующего применения инструментов верификации.

Поэтому в данной работе предлагается новая реализация АОП для языка программирования Си, которая рассматривается в третьей главе.

В третьей части предлагается метод описания в виде контрактных спецификаций правил корректного использования программного интерфейса сердцевины ядра ОС Linux. Суть предлагаемого метода состоит в том, что для элементов программного интерфейса сердцевины ядра, которые релевантны специфицируемым правилам, разрабатывается модель, и на основании состояния этой модели и этих правил задаются предусловия к использованию данных элементов программного интерфейса.

Далее рассматриваются шаги предлагаемого метода.

Шаг 1. Разработка модели для тех элементов программного интерфейса, для правил корректного использования которых необходимо разработать спецификацию. Данная модель включает в себя модельное состояние (набор переменных) и модельную реализацию элементов программного интерфейса (набор функций).

Шаг 2. Описание множества точек использования рассматриваемых элементов программного интерфейса сердцевины ядра ОС Linux.

Шаг 3. Привязка вызовов модельных функций к точкам использования элементов программного интерфейса.

Шаг 4. Задание предусловий использования элементов программного интерфейса на основании тех требований, которые накладывают правила

корректного использования рассматриваемых элементов программного интерфейса сердцевины ядра ОС Linux.

Шаг 5. Объединение частей контрактной спецификации, полученных на шагах с первого по четвертый, и модели обработчиков событий в итоговую контрактную спецификацию, которая представляется в виде аспекта.

Контрактные спецификации для правил корректного использования программного интерфейса сердцевины ядра ОС Linux приведены в приложении А для каждого вида правил из представленного там каталога.

Для демонстрации места контрактных спецификаций в ядре ОС Linux представлены схема обработки событий в ядре ОС Linux (рисунок 1) и схема контрактных спецификаций (рисунок 2).

На схеме обработки событий в ядре ОС Linux показано, что в ядро приходят события от пользовательских приложений (системные вызовы) и от оборудования (прерывания). Данные события диспетчиризуются сердцевиной ядра, в результате чего вызываются функции-обработчики соответствующих модулей. В процессе работы модули через программный интерфейс обращаются к сердцевине ядра, например, с целью выделения и освобождения ресурсов, захвата и освобождения примитивов синхронизации, чтения и записи данных на диск, передачи данных от пользовательских приложений и обратно и т.д.

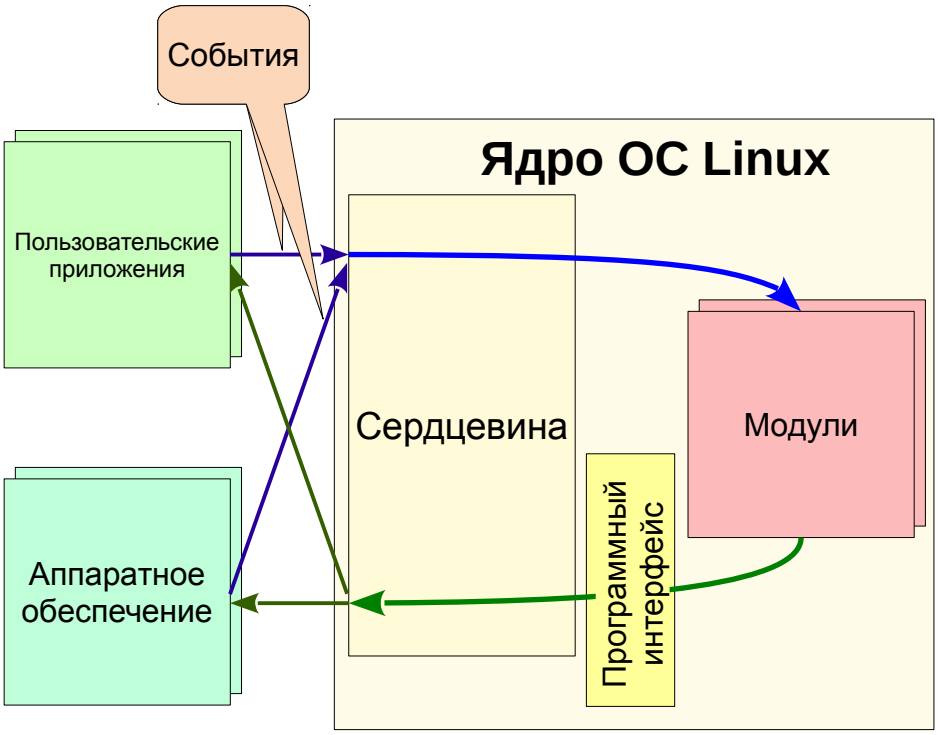

Рисунок 1. Схема обработки событий в ядре ОС Linux.

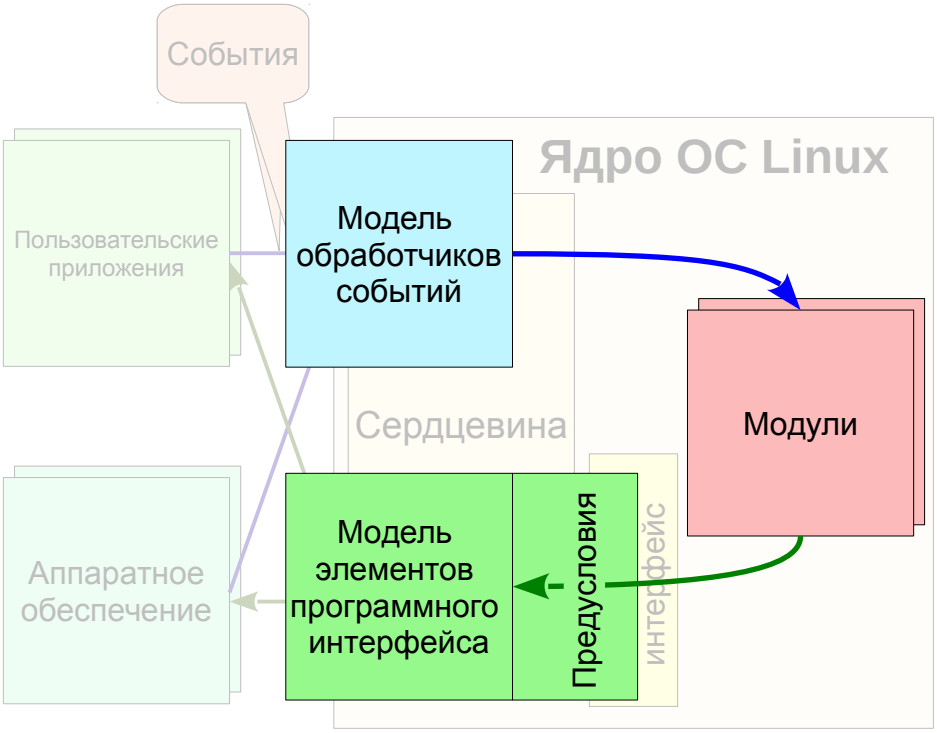

Рисунок 2. Схема контрактных спецификаций.

На схеме контрактных спецификаций, показано, что спецификации состоят из трех частей: модель обработчиков событий, модель элементов программного интерфейса сердцевины ядра ОС Linux и предусловия к

использованию данных элементов. Модель обработчиков событий позволяет описать все возможные варианты воздействия на модули со стороны сердцевины ядра аналогично тому, как это происходит при работе ядра в реальном окружении. Модель элементов программного интерфейса позволяет описать те изменения в состоянии подсистем сердцевины ядра, которые необходимы для определения корректности использования программного интерфейса сердцевины модулями. Посредством предусловий к использованию элементов программного интерфейса сердцевины ядра ОС Linux задаются требования правил корректного использования данного интерфейса.

В четвертой части предлагается метод автоматизированного инструментирования исходного кода модулей ядра ОС Linux на основе контрактных спецификаций, который позволяет применять для его проверки различные инструменты верификации. Суть данного метода состоит в том, что аспекты, с помощью которых представлены контрактные спецификации, компонуются с исходным кодом анализируемого модуля ядра ОС Linux (рисунок 3). В результате этого ставится задача достижимости для инструментов верификации.

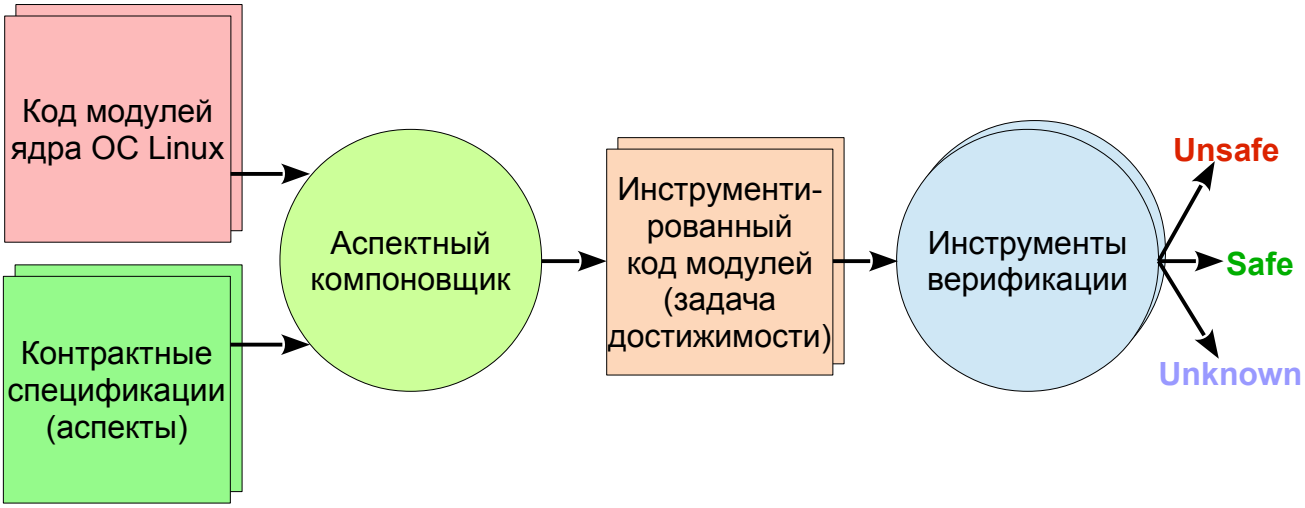

Рисунок 3. Постановка и решение задачи достижимости для инструментов верификации на основе контрактных спецификаций.

По результатам решения данной задачи инструменты верификации могут вынести один из трех вердиктов:

- *Safe* – в анализируемом модуле нет ошибок искомого вида;

- *Unsafe* – выявлено возможное нарушение проверяемых правил корректного использования рассматриваемых элементов программного интерфейса сердцевины ядра ОС Linux;

- *Unknown* – инструмент не смог решить поставленную задачу достижимости (например, из-за нехватки отведенного на решение времени).

Пятая часть посвящена вопросу о соотношении между ядром ОС Linux, контрактными спецификациями и инструментами верификации. В данной части определяется такое понятие, как *избыточные контрактные спецификации* — такие спецификации, которые потенциально позволяют обнаружить все возможные ошибки в коде программ с помощью инструментов верификации. Показывается, что хотя такие спецификации могут привести к ложным срабатываниям, что требует последующего ручного анализа результатов верификации, на практике избыточные контрактные спецификации являются наиболее предпочтительными.

В шестой части представлен подход к автоматизированному анализу структуры программ на основе высокоуровневых запросов по исходному коду. Механизм запросов используется для построения некоторых частей контрактных спецификаций, а также является основой метода автоматизированной поддержки корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса и подхода к автоматизированному уточнению контрактных спецификаций. В предложенном подходе запросы формулируются в виде аспектов. Аспектный компоновщик в специальном режиме проводит анализ исходного кода целевой программы с учетом свойств искомых фрагментов программного кода, заданных в аспектах, и выдает на выходе необходимую информацию.

В седьмой части обсуждается метод автоматизированной поддержки корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса. Поддержка корректности и согласованности контрактных спецификаций является критически важной задачей для ядра ОС Linux, поскольку программный интерфейс его сердцевины меняется достаточно интенсивно и у ядра существует множество различных конфигураций.

Показывается, что благодаря особенностям процесса разработки ядра ОС Linux выбранный способ задания контрактных спецификаций обеспечивает возможность использовать одни и те же спецификации для разных версий/конфигураций ядра. Если для некоторой версии/конфигурации ядра ОС Linux были разработаны контрактные спецификации, то для другой версии/конфигурации ядра необходимо провести автоматизированную проверку синтаксической совместимости данных спецификаций с программным интерфейсом сердцевины этой версии/конфигурации ядра. При наличии синтаксической совместимости контрактные спецификации могут быть использованы, хотя точность верификации при этом может понизиться. В случае обнаружения несовместимости необходимо внести соответствующие исправления в описания множеств точек использования элементов программного интерфейса и, возможно, в модель элементов программного интерфейса сердцевины ядра и в проверяемые предусловия (рисунок 4).

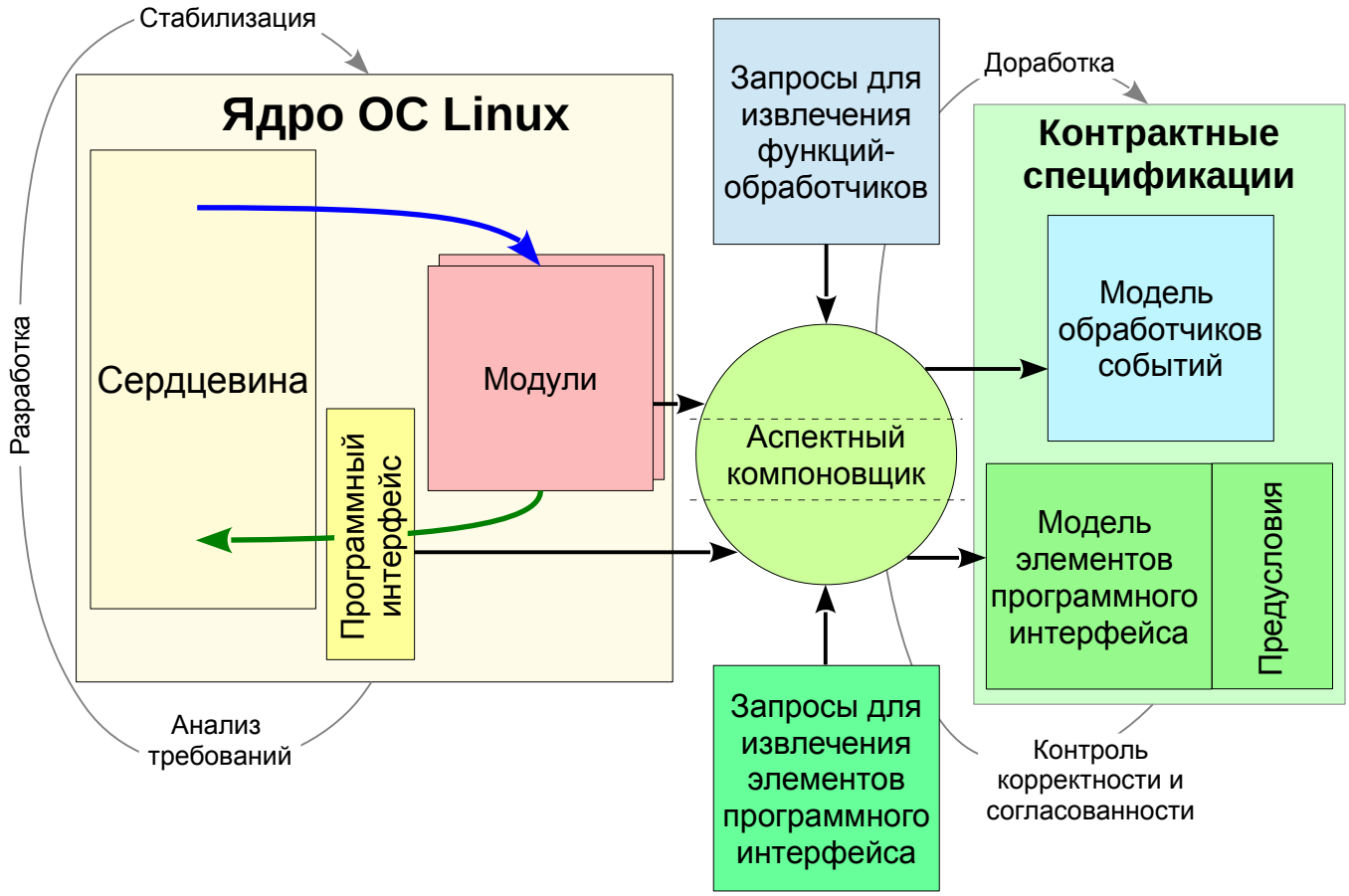

Рисунок 4. Схема контроля корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса сердцевины ядра ОС Linux.

Таким образом, поддержка корректности и согласованности контрактных спецификаций требует существенно меньших трудозатрат по сравнению с разработкой новых контрактных спецификаций для каждой новой версии/конфигурации ядра ОС Linux.

В восьмой части рассматривается подход к автоматизированному уточнению контрактных спецификаций. Данный подход является важным, поскольку благодаря ему при практическом применении удается частично устранить последствия чрезмерной избыточности контрактных спецификаций, а именно устранить большое количество ложных срабатываний.

В конце второй главы делаются выводы о том, что предложенный метод контрактных спецификаций отвечает всем тем требованиям, которые были сформулированы в первой главе.

Третья глава посвящена описанию инструментария, который реализует предложенный метод контрактных спецификаций.

Сначала рассматривается наиболее важная часть инструментария, а именно новая реализация АОП для языка Си. Данная реализация включает в себя аспектно-ориентированное расширение языка программирования Си с поддержкой всех расширений компилятора GCC и аспектный компоновщик (рисунки 3, 4).

В первой части и приложении Б представлено аспектно-ориентированное расширение языка программирования Си с поддержкой всех расширений компилятора GCC. Данное расширение используется в предложенном методе контрактных спецификаций для задания спецификаций в виде аспектов, а также для написания высокоуровневых запросов по исходному коду, которые используются для автоматизированного анализа структуры программы.

Вторая часть описывает аспектный компоновщик, который используется для выполнения инструментирования исходного кода модулей ядра ОС Linux на основе контрактных спецификаций, а также для выполнения высокоуровневых запросов по исходному коду. Представлена архитектура аспектного компоновщика и основные этапы работы инструмента.

В третье части приводится описание базы контрактных спецификаций, которая содержит описание всех поддерживаемых контрактных спецификаций. Для каждой контрактной спецификации задаются:

- уникальный идентификатор;
- набор файлов, в том числе с аспектом;
- набор дополнительных опций, которые определяют, например, какой инструмент верификации необходимо использовать для проверки.

В четвертой части показано место разработанного инструментария в системе верификации Linux Driver Verification. Показывается, каким образом в

системе верификации используется аспектный компоновщик и какие данные ему подаются на вход. Затем рассматриваются инструменты, которые используют новую реализацию АОП. Данные инструменты реализуют:

- подход к автоматизированному анализу структуры программ на основе высокоуровневых запросов по исходному коду;

- метод автоматизированной поддержки корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса (рисунок 4);

- подход к автоматизированному уточнению контрактных спецификаций.

В **четвертой главе** рассматривается практическое применение предлагаемого метода контрактных спецификаций для решения задач поддержки крупных развивающихся систем и дается оценка его эффективности.

Показывается, что предложенные средства для задания контрактных спецификаций обладают достаточной степенью выразительности, но при этом понятны людям, которые не являются экспертами в области верификации и в устройстве ядра ОС Linux.

Приводятся результаты, которые были получены в рамках практического применения разработанного инструментария в составе системы верификации, разработанной в рамках проекта Linux Driver Verification. В среднем для ядра версий от 2.6.31.6 и до 3.11-rc1 был успешно инструментирован исходный код более 95% модулей (всего их порядка 2 — 4 тыс. в зависимости от версии ядра). Инструментированный код был успешно проверен с помощью нескольких инструментов верификации: BLAST, CPAchecker, UFO и CBMC.

По результатам проведенной верификации модулей нескольких версий ядра ОС Linux к настоящему времени было выявлено более 125 ошибок. Исправления обнаруженных ошибок были направлены разработчикам ядра ОС Linux, которые включили их в состав последних версий ядра. Коллектив

проекта Linux Driver Verification постоянно осуществляет мониторинг новых версий ядра ОС Linux, что позволяет как дорабатывать контрактные спецификации, так и находить новые ошибки по мере развития ядра.

Дается оценка числа модулей, в которых происходят ложные срабатывания вследствие избыточности контрактных спецификаций и неточностей в алгоритмах верификации. В среднем данное число составляет менее 1% от числа всех модулей ядра. Особо отмечается, что благодаря автоматизированному уточнению контрактных спецификаций для ряда спецификаций удалось сократить число ложных срабатываний более чем на 70%.

В последней части показывается, как разработанные контрактные спецификации и инструментарий были использованы в процессе подготовки эталонного набора задач достижимости для измерения сравнительных характеристик инструментов верификации. Показывается, что поскольку данный набор используется в рамках ежегодных мероприятий ETAPS/Competition on Software Verification, то все инструменты верификации, которые участвуют в этих мероприятиях, потенциально могут быть использованы для проверки модулей ядра ОС Linux на предмет выполнения требований контрактных спецификаций автоматизированным образом.

В **заключении** перечисляются основные результаты работы.

Основные результаты работы

Основные научные результаты, полученные в диссертационной работе и выносимые на защиту, состоят в следующем:

- разработан метод описания в виде контрактных спецификаций правил корректного использования программного интерфейса сердцевины ядра ОС Linux, основанный на предложенном новом аспектно-ориентированном расширении языка Си;
- разработан метод автоматизированного инструментирования исходного кода модулей ядра ОС Linux на основе контрактных спецификаций, позволяющий применять для его проверки различные инструменты верификации;
- разработан метод автоматизированной поддержки корректности и согласованности контрактных спецификаций в условиях изменения программного интерфейса.

На основе предлагаемых методов в рамках проекта Linux Driver Verification при непосредственном участии автора в качестве руководителя, проектировщика и одного из основных разработчиков были разработаны контрактные спецификации для 34 правил корректного использования программного интерфейса сердцевины ядра ОС Linux и инструментарий, который реализует предложенный метод контрактных спецификаций и который был включен в состав системы верификации модулей ядра ОС Linux.

Работы автора по теме диссертации

1 Новиков Е.М. Подход к реализации аспектно-ориентированного программирования для языка Си // Программирование. 2013. №4. С. 47-65.

2 Новиков Е.М. Построение спецификаций программных интерфейсов в

открытой системе покомпонентной верификации ядра Linux // Труды Института системного программирования РАН. 2013. Т. 24. С. 293-316.

3 Новиков Е.М., Хорошилов А.В. Использование аспектно-ориентированного программирования для выполнения запросов по исходному коду программ // Труды Института системного программирования РАН. 2012. Т. 23. С. 371-386.

4 Мандрыкин М.У., Мутилин В.С., Новиков Е.М., Хорошилов А.В., Швед П.Е. Использование драйверов устройств операционной системы Linux для сравнения инструментов статической верификации // Программирование. 2012. №5. С. 54-71.

5 Мандрыкин М.У., Мутилин В.С., Новиков Е.М., Хорошилов А.В. Обзор инструментов статической верификации Си программ в применении к драйверам устройств операционной системы Linux // Труды Института системного программирования РАН. 2012. Т. 22. С. 293-326.

6 Мутилин В.С., Новиков Е.М., Хорошилов А.В. Анализ типовых ошибок в драйверах операционной системы Linux // Труды Института системного программирования РАН. 2012. Т. 22. С. 349-374.

7 Мутилин В.С., Новиков Е.М., Страх А.В., Хорошилов А.В., Швед П.Е. Архитектура Linux Driver Verification // Труды Института системного программирования РАН. 2011. Т. 20. С. 163-187.

8 Новиков Е.М. Использование аспектно-ориентированного подхода программирования в процессе верификации ядра операционной системы Linux // Сборник научных трудов Научно-практической конференции «Актуальные проблемы программной инженерии». Москва, 2009. С. 148-154.

9 Мутилин В.С., Новиков Е.М., Хорошилов А.В. Классификация концепций для аспектно-ориентированного расширения языка программирования С // Труды 52-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук», часть 7 «Управление и прикладная математика». Долгопрудный, 2009. Т. 2. С. 31-33.

10 Новиков Е.М., Хорошилов А.В. Реализация аспектно-

ориентированного программирования для языка Си // Тезисы докладов восьмой конференции разработчиков свободных программ. Обнинск, 2011. С. 23-26.

11 Щепетков И.В., Новиков Е.М. Диагностика синтаксической совместимости правил корректности с ядром ОС Linux при статической верификации драйверов // Сборник научных трудов Научно-практической конференции «Актуальные проблемы системной и программной инженерии». Москва, 2013. С. 192-201.

12 Novikov E. One Approach to Aspect-Oriented Programming Implementation for the C programming language // Proceedings of the 5th Spring/Summer Young Researchers' Colloquium on Software Engineering. Yekaterinburg. 2011. Pp. 74-81.

13 Khoroshilov A., Mutilin V., Novikov E., Shved P., Strakh A. Towards an open framework for C verification tools benchmarking // Perspectives of Systems Informatics, Lecture Notes in Computer Science. 2012. V. 7162. Pp. 179–192.

14 Zakharov I., Mutilin V., Novikov E., Khoroshilov A. Generating Environment Model for Linux Device Drivers // Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering. Kazan. 2013. Pp. 77-83.

15 Beyer D., Löwe S., Novikov E., Stahlbauer A., Wendler P. Precision Reuse for Efficient Regression Verification // Proceedings of the 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. 2013. Pp. 389-399.

16 Новиков Е.М. Свидетельство о государственной регистрации программы для ЭВМ №2012615598 от 20.06.2012 «Система аспектно-ориентированного программирования для языка Си».

17 Мандрыкин М.У., Мутилин В.С., Новиков Е.М., Хорошилов А.В., Швед П.Е. Свидетельство о государственной регистрации программы для ЭВМ №2012615596 от 20.06.2012 «Система проверки выполнения проблемно-ориентированных правил для Си программ».