Содержание

Предисловие редактора перевода

На современном этапе развития общества особое внимание уделяется развитию систем связи и управления, функционирующих в едином информационном пространстве Российской Федерации.

Определяющей технологией, на базе которой планируется создавать такие системы, является программно-конфигурируемое радио (**Software Defined Radio — SDR**). Все его настройки и параметры задаются программно. Конфигурация SDR определяется с помощью программного обеспечения. Для этой цели используются либо процессоры цифровой обработки DSP, либо матрицы FPGA (Field-Programmable Gate Array — программируемая пользователем вентильная матрица, одна из типов ПЛИС — программируемых логических интегральных схем).

Имеется международный опыт разработки программно-конфигурируемого радио для тактического звена. Так, в США создана **JTRS (Joint Tactical Radio System)** — военная система для связи в американской армии. Изначально JTRS была предназначена для замены 25—30 различных типов военных систем связи (некоторые из них не могли связываться между собой) на одну программную радиосистему, которая могла бы работать во всех диапазонах частот. Помимо военных (WNW-BEAM, OFDM, AJ, LPI и др.) система JTRS должна поддерживать сотовую связь различных стандартов (GSM, CDMA), а также транкинговую связь стандартов TETRA и TETRAPOL.

Концепция американской программы JTRS использовалась для разработки двух европейских программ: ESSOR и SVFuA.

ESSOR (European Secure Software Radio Program) создается усилиями шести стран (Финляндия, Франция, Италия, Польша, Испания, Швеция) и базируется на открытом стандарте SCA (Software Communication Architecture — архитектура программной связи).

SVFuA является немецкой национальной программой и предназначена для применения бундесвером (Федеральными вооруженными силами Германии).

Наличие разработанной в JTRS архитектуры SCA позволило европейским странам ускорить и удешевить программы ESSOR и SVFuA. Поэтому мы надеемся, что перевод книги, посвященный SCA, позволит и отечественным разработчикам за короткие сроки и с меньшими затратами реализовать проекты, аналогичные JTRS, ESSOR и SVFuA.

Книга в первую очередь адресована специалистам по разработке систем радиосвязи двойного назначения. Для работы с книгой требуется базовая подготовка в области радиотехники и прикладного программирования. По причине отсутствия устоявшейся русскоязычной терминологии большинство сокращений дано на английском языке. Их список с русским переводом выделен в отдельное приложение.

Радько Н. М.
Заместитель Генерального директора по науке,
кандидат технических наук,
лауреат Государственной премии в области науки и техники.

Часть I

ОПЕРАЦИОННАЯ СРЕДА

В первой части книги приводится спецификация архитектуры программно определяемых средств связи (SCA, software communications architecture). Цель этой части — дать общую информацию по SCA с пояснениями относительно толкования и функциональных требований к ней. Эта часть книги может использоваться как отдельный справочник по SCA.

ГЛАВА 1

ВВЕДЕНИЕ

Фундаментальной задачей SCA является обеспечение общей инфраструктуры программного обеспечения (ПО) для управления радиосистемами. Несмотря на то, что ПО является ключевым компонентом современных радиосистем, благодаря которому они получают новые функции и возможности, каждый производитель радиосистем использует собственную архитектуру и/или инфраструктуру системы, поэтому для загрузки и управления ПО используются особые, уникальные для каждого производителя механизмы. Как было сказано выше, программно-определяемое радио относится к классу радиосистем, характеристики которых не просто обусловлены программным обеспечением, но используют инфраструктуру, в которой взаимозаменяемы и компоненты, и функции.

В этой главе содержится основная информация об SCA. В спецификации SCA описываются компоненты архитектуры, их структура и объединение их в систему, формирующую радиосигнал. Все эти элементы образуют инфраструктуру для создания программно-определяемой радиосистемы (Software defined radio, SDR).

1.1. Программируемые радиосистемы

Радиосистему можно представить в абстрактном виде, рассматривая форму сигнала и используемый диапазон частот (см. рис. 1.1). Низший уровень схемы занимает комплекс аппаратных средств, который обеспечивает работу ПО, формирующего сигнал и выполняющего вспомогательные задачи. Обработка данных может осуществляться одним из четырех устройств: универсальным центральным процессором (ЦП), цифровым сигнальным процессором (ЦСП), программируемой логической схемой (ПЛИС) или специализированной большой интегральной схемой (СБИС). Строго говоря, такую БИС нельзя использовать как основу программируемой радиосистемы, так как конфигурация этой микросхемы задается при изготовлении и не может быть изменена (перепрограммирована) в процессе работы, что нарушает один из фундаментальных принципов SDR. Однако, по мнению авторов, БИС можно применять в SDR при условии, что она обеспечивает возможность гибкой взаимозаменяемости функциональных компонентов обработки сигналов внутри микросхемы. Например, вызов какого-либо алгоритма внутри БИС должен осуществляться аналогично вызову программных функций.

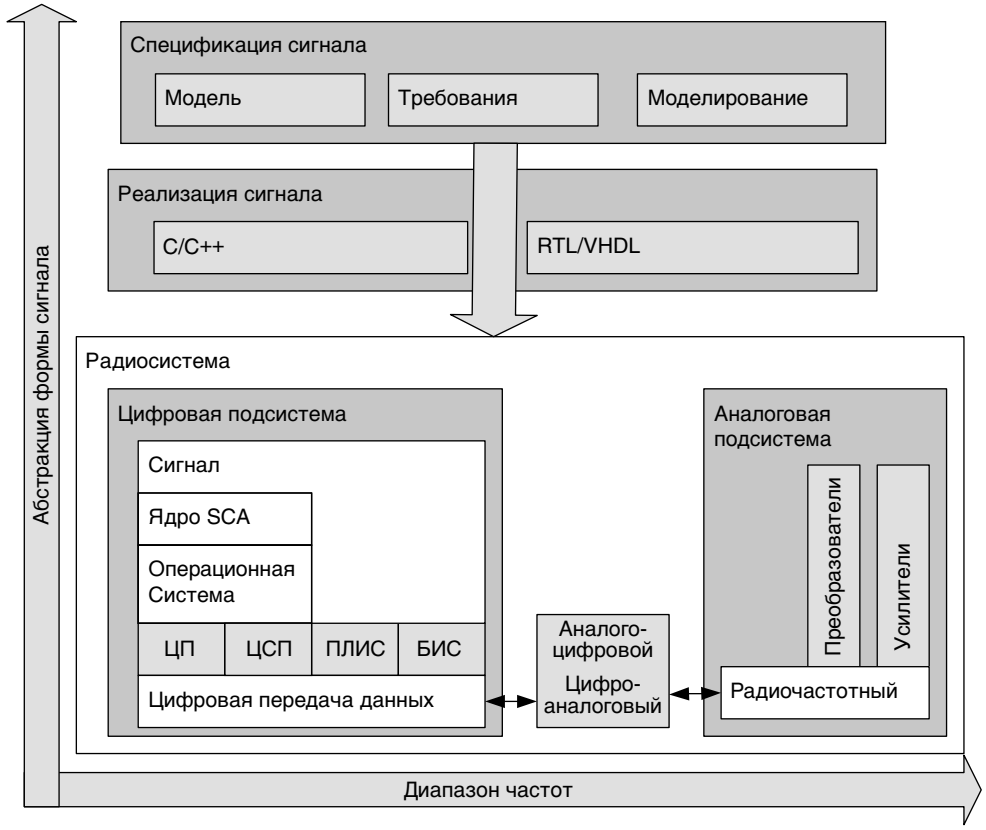

Рис. 1.1. Абстрактное представление системы радиосвязи

Рис. 1.1 демонстрирует две ортогональные проекции процесса разработки SDR. Создание сигнала начинается с разработки комплекса требований, имитационной или математической модели или другого концептуального представления сигнала. По мере разработки формы сигнала параметры его реализации (пропускная способность и мощность) обычно задаются на языке высокого уровня и для их выполнения на ЦП или ЦСП. Для повышения пропускной способности системы необходимо использовать ПЛИС или СБИС.

ЦП, как правило, применяется для управления всей системой. Находящиеся выше процессора ОС с пакетом ПО служат основой динамической структуры радиосистемы. В терминах SCA эта структура называется ядром SCA (SCA Core Framework). Над ней на рисунке находятся приложения, формирующие сигнал и выполняющие прочие задачи.

1.1.1. Аспекты программируемых радиосистем

Систему SDR можно рассматривать с четырех аспектов, каждый из которых образует функциональную группу объектов и служб в радиосистеме (рис. 1.2):

- **аппаратные средства** — физическая структура радиокомплекса, т. е. совокупность входящих в нее устройств;
- **программное обеспечение** — службы и интерфейсы, которые согласуют с основными аппаратными средствами все сигнальные приложения;
- **приложения** — к этому аспекту относятся приложения для создания формы сигнала и службы, которые предоставляет радиосистема;
- **пользователь** — взаимодействует с радиокомплексом. Существуют два основных режима взаимодействия: пользователь либо контролирует систему в целом (например, задает параметры системы), либо управляет работой приложений и передачей данных (например, определяет коэффициент усиления в момент передачи).

SCA можно рассматривать как реализацию аспекта ПО с некоторыми элементами аспекта приложений. Она определяет логическую инфраструктуру и абстрактное представление аппаратных средств и компонентов ПО цифровой обработки, входящих в формирующее сигнал приложение. SCA также характеризует функции системы и интерфейсы управления, загрузки и конфигурирования приложений, формирующих сигнал, внутри системы.

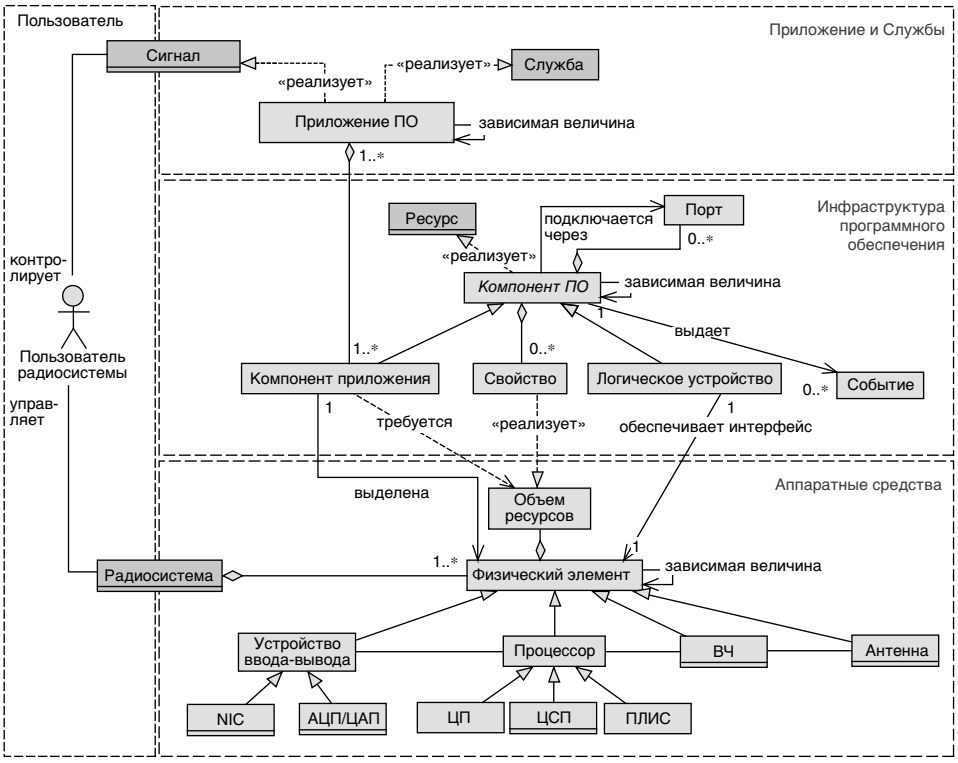

Рис. 1.2. Аспекты программируемого радио

1.2. Архитектура программируемых средств связи

Любая новая концепция или технология требует времени и определенных знаний для ее освоения, и концепция SCA — не исключение. Она определяет инфраструктуру ПО для управления SDR. Она не подчинена какой-либо особой структуре или реализации аппаратных средств и приложений. Перед детальным рассмотрением SCA рекомендуется ознакомиться с тем, что есть и чем не является SCA, историей ее развития и причинами, побуждающими ее использовать или отказаться от нее.

SCA основана на нескольких взаимосвязанных технологиях: объектно-ориентированные (ОО) методы создания ПО, общая архитектура брокера объектных запросов (common object request broker architecture — CORBA) и модель компонентов CORBA (CORBA components model — CCM).

Объектно-ориентированные языки используются в течение нескольких десятилетий, начиная с языка Simula, появившегося в конце 1960-х годов, Smalltalk и Flavors — в начале 1980-х и завершая современными ОО-языками — C++, Python, Ruby и Java. По мере совершенствования систем с распределенной архитектурой и структурой клиент-сервер CORBA стала промышленным стандартом для описания интерфейсов на псевдокоде, т. е. языке описания интерфейсов (interface definition language — IDL). IDL обеспечивает средства для указания доступных интерфейсов и, при помощи «компилятора» IDL, генерирует исходный код, который компилируется для каждого приложения. Сгенерированный код содержит служебные алгоритмы, необходимые для поддержки удаленных вызовов процедур между процессами на одном и том же компьютере и между разными компьютерами, т. е. в распределенной среде. Таким образом, программист был избавлен от рутиной работы по созданию кода межпроцессорной связи низкого уровня, и, что более важно, код CORBA, созданный одним лицом, был совместим с кодом, созданным другим лицом, при единственном условии, что оба автора клиентского и серверного приложения использовали один и тот же язык IDL. Вместе с выделением внутренней логики в самостоятельный элемент (инкапсуляцией) это было важным шагом в направлении совершенствования модульного программного обеспечения: единственным требованием стало использование единого языка IDL разработчиками и клиентского, и серверного приложений.

Несмотря на то, что технология CORBA имела ряд важных преимуществ, стало очевидным, что механизм развертывания системы все еще требовал ручной настройки параметров конфигурации. При загрузке ПО необходимо явно указать требования для его установки, то есть какие ресурсы требуются для этих приложений. Для описания компонентов системы и сопутствующих требований по установке используются XML-файлы. XML (eXtensible Markup Language) — это

текстовый язык, использующий ключевые слова (теги) для определения элементов данных, их атрибутов и значений. На языке XML CCM основан язык XML профиля домена SCA (Domain profile XML).

1.2.1. Развитие архитектуры программируемых средств связи

Во время проведения военных операций военное ведомство США столкнулось с острой необходимостью обеспечить надежную связь с высоким уровнем совместимости и малыми затратами. Большое количество радиосистем было основано на аппаратной платформе с ограниченными возможностями по формированию сигналов, и это стало одним из главных препятствий на пути создания SCA. Кроме того, было невозможно обновлять существующие или добавлять новые сигналы без значительных затрат на модернизацию аппаратных средств. Вместе с тем, за последние двадцать лет значительно повысилась мощность процессоров, и стали общедоступными специальные процессоры ЦСП и ПЛИС, также непрерывно повышались производительность и разрешающая способность аналогово-цифровых и цифрово-аналоговых систем. Как результат, для обработки большинства сигналов вместо аналоговых систем стали использоваться цифровые, реализованные программно. Ранние эксперименты в области SDR (например, система SpeakEasy) показали, что программно-реализованная архитектура имеет значительные преимущества. Многие производители радиосистем уже начали работы по реализации программных компонентов обработки базового сигнала. Первые многоканальные радиосистемы, разработанные в 1990-х годах — боевой информационный терминал (Joint Combat Information Terminal, JCIT) и цифровая блочная радиостанция (Digital Modular Radio, DMR), — обеспечивали программную инфраструктуру для управления ресурсами радиосистемы.

Из-за необходимости улучшения возможностей перенастройки системы, поддержки совместных действий и снижения затрат на эксплуатацию и обслуживание, для разработки нового семейства программно-реализованных реконфигурируемых радиосистем было создано Управление по разработке совместных программ (Joint Program Office, JPO) Joint Tactical Radio System (JTRS, совместные тактические радиосистемы). Одной из первых задач Управления было определение общей инфраструктуры программного обеспечения для использования в новых семействах радиосистем. Таким образом и возникла SCA.

На рис. 1.3 показаны несколько ключевых этапов развития спецификации SCA. Были разработаны несколько предварительных ее версий, но первой, использованной для разработки первоначальной реализации SCA стала версия 1.0. После следующей версии 1.1 значительная часть спецификации была переработана, и появилась версия 2.0. Она имела некоторые недостатки, для устранения которых нужно было пересмотреть все аспекты инфраструктуры программного

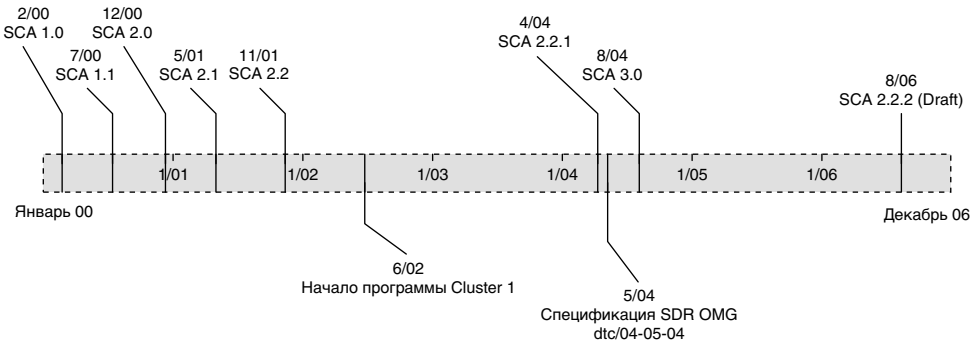

Рис. 1.3. Хронология SCA

обеспечения. Тем не менее, были разработаны несколько реализаций версии 2.0, которые обеспечили важную для разработки следующих версий информацию «обратной связи» (feedback). Следующая версия 2.1 появилась в середине 2001 года, а в ноябре была выпущена версия 2.2. В целом, версия 2.2 считалась достаточно завершенной для реализации и применения в системе SDR[1].

В июне 2002 года фирма Boeing получила от CECOM (Communication and Electronic Command, командование связи и электроники) заказ на разработку первой базовой программы внедрения SCA. Первым проектом, использующим SCA версии 2.2, стал Cluster 1, который позже получил название Ground Mobile Radio (GMR, наземная радиостанция мобильной связи). Позднее стали разрабатываться другие программы JTRS Cluster, и в апреле 2004 года, почти через три года после появления спецификации 2.2, была выпущена версия 2.2.1. В этой версии были устранены многие ошибки и внесены несколько уточнений и улучшений. Наиболее важным изменением в версии 2.2.1 было исключение из спецификации SCA службы Log Service (служба регистрации) и замена ее на службу OMG Lightweight Log (упрощенная регистрация OMG — группы по управлению объектами). В середине 2004 года OMG выпустила свою спецификацию SDR. Спецификация OMG была создана группой специалистов, которые участвовали в разработке SCA. Их первоначальной целью было сделать возможным использование SCA как промышленного стандарта, а не только военного. Но, как только спецификация OMG появилась на свет, стало ясно, что она существенно отличается от спецификации SCA.

В то же время в программах JTRS Cluster обнаружились проблемы, связанные с совместимостью сигналов. Проблема заключалась в том, что код, написанный для ЦП, был совместим с различными аппаратными платформами, но код для

[1] Несмотря на это для того, чтобы стать достаточно надежной и эффективной, версия 2.2 требует некоторых корректировок и уточнений. Например, интерфейс IDL, представленный в Приложении C спецификации SCA, не соответствует заданным параметрам из-за конфликта имен с интерфейсом POSIX.

ЦСП и ПЛИС оставался «привязаным» к определенной платформе и архитектуре системы[2].

Эта проблема наиболее остро проявилась в конце 2004 года, что побудило JPO созвать нескольких семинаров для решения проблемы совместимости процессоров ЦСП и ПЛИС. Их результатом стало появление спецификации SCA 3.0. В отличие от предыдущей, новая версия SCA имела незначительные отличия от прошлой. Но в ней появились дополнительные ограничения на системные вызовы, которые могут быть использованы кодом ЦСП, задан предполагаемый набор сигнальных компонентов, предложена высокоуровневая схема передачи данных (получившая название HAL-C) и появился API антенны. Специалисты сошлись во мнении, что необходимо доработать спецификацию — несмотря на наличие потенциально полезных концепций и подходов, для успешной реализации проекта требуется более детальный анализ спецификации.

С конца 2005 до начала 2006 года управление JPO было реорганизовано для более успешного решения проблем с программами Cluster. Отдел по реализации программы был перенесен из Вашингтона в Сан-Диего и передан в подчинение командованию боевых космических и морских систем (Navy SPAWAR). В середине 2006 года была выпущена версия 2.2.2 — развитие версии SCA 2.2.1. При этом на веб-сайте JTRS указано, что версия 3.0 «не поддерживается». Поэтому на момент выхода данной публикации версия 2.2.2 является последней поддерживаемой версией SCA.

1.2.2. Понятие SCA

Основной задачей SCA является определение программного обеспечения операционной среды (ОС) — инфраструктуры, или ядра SCA. Оно используется для управления, развертывания, настройки и регулирования радиосистем и приложений, которые запускаются на платформе радиосистемы. Для того, чтобы получить представление о том, что такое SCA и чем она не является, целесообразно обратиться к введению спецификации SCA. Ниже приводится выдержка из этой части.

Спецификация программируемых средств связи (SCA) опубликована Управлением по разработке совместных программ (JPO) JTRS. Управление было организовано для создания перспективных систем средств связи с учетом передового опыта использования технологии по улучшению совместимости средств связи и снижению затрат на их разработку и развертывание. Основными задачами программы JTRS являются:

---

[2] Совместимость моделей сигналов стала существенной проблемой внутри сообщества SCA. Это ни в коем случае не является фундаментальным требованием SCA. Совместимость формы сигнала можно рассматривать как возможность снизить затраты при создании радиосистем. Но из-за различий в аппаратных средствах, процессоре и архитектуре передачи данных невозможно обеспечить элементарную передачу сигнала от одной системы к другой, не внося в них изменения.

- значительное повышение операционной гибкости и совместимости глобальных систем связи;
- снижение затрат на техническую поддержку;
- возможность модернизации, то есть внедрения новых технологий и улучшения характеристик;
- снижение затрат на приобретение и эксплуатацию систем.

Для успешного решения этих задач спецификация SCA была структурирована с целью:

- обеспечения совместимости программного обеспечения приложений между различными реализациями SCA;
- использования коммерческих стандартов для снижения затрат на разработку;
- снижения сроков разработки новых форм сигналов посредством использования модулей;
- совершенствования промышленной инфраструктуры и архитектуры.
- SCA V2.2, 17 ноября 2001 года, стр. 7.

Ни одна из вышеуказанных задач не связана ни с техническим проектом и процессом разработки, ни с реализацией сигналов в радиосистеме. Поэтому первой фундаментальной целью SCA является совершенствование технико-экономического обоснования проекта (ТЭО) с целью улучшения рабочих характеристик средств связи и их материально-технического обеспечения.

Для успешного решения этой задачи структура SCA призвана обеспечить совместимость программного обеспечения приложений, использование промышленных стандартов, поддержку повторного использования модулей, формирующих сигнал, а также совершенствование промышленной инфраструктуры. Наряду с задачами, обозначенными в предыдущем параграфе, структура SCA фокусируется на процессах проектирования и разработки, обеспечивая повторное использование проектных модулей и применение коммерческого ПО и стандартов.

1.2.3. Общие представления об SCA

В предыдущем разделе было дано краткое описание SCA. Не секрет, что для разных индивидуумов технологии зачастую могут подаваться и пониматься поразному — некоторые представления основаны на реальных фактах, другие же базируются на неполном понимании сути технологии. Ниже будут изложены некоторые из типичных заблуждений относительно SCA.

1.2.3.1. SCA определяет техническую архитектуру программируемого радио

Спецификация SCA не содержит информацию о технических характеристиках или инструкции по проектированию и реализации программируемого радио. Спецификация SCA, основанная на компонентах модели CORBA, определяет

архитектуру для развертывания приложений. В SDR в качестве приложений используются модели сигналов.

1.2.3.2. SCA дает возможность повторного использования модулей

SCA повышает возможность повторного использования программных модулей с двух точек зрения. Во-первых, спецификация SCA определяет единый пакет интерфейсов для базовой начальной конфигурации и для управления приложениями. Таким образом, со стороны пользовательских интерфейсов и внешнего управления системой для загрузки, запуска и остановки сигналов например, стандарта SINCGARS, используются те же интерфейсные вызовы, что и для сигналов стандарта FM3TR. Во-вторых, дополнение API к спецификации обеспечивает повторное использование компонентов программного обеспечения через общие интерфейсы сигналов. Данная тема требует дальнейшего обсуждения, поскольку все разработчики радиосистем имеют собственные представления о том, какие интерфейсы должны быть у разработанных ими сигналов.

1.2.3.3. Сигнал может быть перемещен с одной платформы SCA на другую без преобразования

Многие понимают понятие переносимости сигналов SCA как возможность многократного их применения без внесения каких-либо изменений. Спецификация SCA определяет общие интерфейсы высокого уровня для развертывания, настройки, управления и контроля аппаратных систем и программных приложений внутри радиосистемы. Это облегчает процесс переноса приложений, поскольку интерфейсы не меняются. Но возможность использования одних и тех же сигналов в различных радиосистемах без внесения в них изменений никогда не рассматривалась как непременное требование.

1.2.3.4. Архитектура SCA влияет на параметры сигнала в системе

После того как SCA загрузит сигнал в радиосистему, ядро SCA переходит в режим ожидания и использует лишь малую часть времени процессора. Кроме того, SCA исключает влияние на сигналы, реализованные на процессорах ПЛИС или ЦСП. SCA может оказывать некоторое влияние на объем используемой памяти, так как она требуется для работы ядра SCA. Тем не менее, рабочие группы SDR Forum, NASA и некоторые другие исследуют возможности уменьшения объема занимаемой памяти. В случае, когда ядро SCA запускается на ЦП, который используется и для обработки сигналов, возможно некоторое ухудшение производительности. В этом случае для определения допустимого диапазона нагрузок необходимо провести стандартный системный анализ

1.2.3.5. Для передачи данных необходимо использовать CORBA

Об использовании CORBA стоит думать в первую очередь, однако это не является обязательным условием в том случае, если применение CORBA ухудшает производительность. Стоит отметить, что пользователи часто ошибаются, приписывая

задержки передачи данных CORBA, хотя в действительности они происходят на уровне протокола TCP/IP. CORBA же — это протокол, скорее похожий на протокол передачи гипертекста HTTP и находится на верхнем уровне механизма передачи данных. Большинство современных ORB поддерживают подключаемые интерфейсы транспорта данных, обеспечивая гибкую настройку и оптимизацию фактического потока передачи данных.

1.2.3.6. SCA пригодна только для небольших радиосистем

SCA изначально не разрабатывалась для применения в радиосистеме какого-либо определенного класса. При создании SCA-совместимой системы малого размера и с ограниченными ресурсами возникает больше сложностей, связанных с ограничениями по весу, размеру и мощностью, чем при разработке переносной или ранцевой радиостанции. Обычно это связано с тем, что ЦП малой радиостанции активно используется для обработки сигналов и если ЦП будет обрабатывать и ядро SCA, это может оказать значительное влияние на производительность системы.

1.2.3.7. SCA и/или CORBA не годятся для больших, сложных систем

SCA основана на программе JTRS, целью которой была разработка систем тактического радио. Существует множество модификаций таких систем, начиная от небольших переносных радиостанций и заканчивая системами для монтажа в стойку (для автомобилей, кораблей и самолетов). Хотя такие системы не оснащены сложными терминальными системами, SCA может использоваться и в больших системах. В этом случае стоит уделить больше внимания архитектуре системы. Возможет вариант, когда SCA управляет базовым набором радиоаппаратуры и загрузкой сигналов, управляясь командами от системы высшего уровня. Ключевым аспектом является то, что SCA управляет аппаратным и программным обеспечением, которое используется для реализации и поддержки комплексного приложения.

Если говорить о применении CORBA в крупных системах, то она широко используется во всех отраслях промышленности. Фактически крупные, распределенные приложения, основанные на Java и использующие удаленные вызовы процедур CORBA и Java, используют протокол CORBA. В системе управления спутниковой сети связи Iridium COTS-системы OS/COMET объединены комплексной инфраструктурой CORBA. Конечная система запускалась на более чем 50 компьютерах и одновременно выполняла несколько сотен процессов.

1.2.3.8. SCA не годится для систем, работающих на частотах выше 2 ГГц

Обычно эта причина обосновывается тем, что разработанные в рамках программы JTRS тактические радиостанции работали на частотах ниже 2 ГГц. Но спецификация SCA не содержит каких-либо элементов, которые могли бы в явной или скрытой форме ограничить способности SCA работать на частотах более 2 ГГц.

1.2.4. Цели применения SCA

Учитывая, что SCA не является технической архитектурой программируемой радиосистемы, возникает вопрос о необходимости ее применения. Как было сказано выше, частично это объясняется тем, что задачей SCA является достижение коммерческой эффективности при снижении затрат на модернизацию, обновление и обслуживание. Это те преимущества, которые главным образом реализуются заказчиком или пользователем системы SCA. Таким образом, основной причиной использования SCA является то, что этого требует проект. Для обеспечения долговечности и достаточной приемлемости проекта разработчики должны иметь обоснованные коммерческие причины ее использования.

Далее перечислены некоторые причины, которые необходимо учитывать при рассмотрении вопроса об использовании архитектуры.

1.2.4.1. SCA обеспечивает общую инфраструктуру для развертывания распределенного приложения

Хотя SCA была разработана для систем радиосвязи, базовую инфраструктуру для размещения компонентов приложения можно использовать практически в любой системе. Это касается тех элементов системы, которые имеют отношение к обработке радиосигналов, либо же это могут быть узлы, не имеющие отношения к радио — например, процессорное управление устройствами и ПО на транспортном средстве.

1.2.4.2. SCA обеспечивает быструю интеграцию внешних приложений

Компактность пакета интерфейсов внутри SCA позволяет легко интегрировать внешние приложения с помощью языка IDL, предназначенного для загрузки, пуска, остановки и управления приложениями внутри системы SCA. В качестве примера можно привести интеграцию системы SCA внутри сети систем. Общее управление узлами радиосистемы часто осуществляется с помощью сетевого управления (network management и network control). Такие системы управления часто используют протокол администрирования сети (SNMP) или язык Java, которые имеют собственную поддержку CORBA. Таким образом, в случае, когда с помощью ПО управления сетью требуется загрузить сигнал или коммуникационное ПО в SCA, для этого достаточно использовать запуск удаленных процедур Java (Java RPC). При использовании протокола SNMP можно использовать прокси-сервер SNMP, который служит интерфейсом SNMP в сетевой системе и обеспечивает Java RPC в SCA-радиосистеме.

1.2.4.3. SCA обеспечивает быстрое внедрение новых технологий

Поскольку SCA задает интерфейсы для развертывания, настройки, управления и мониторинга аппаратного и программного обеспечения в SCA-системе, новые технологии могут внедряться с меньшими затратами и без ухудшения рабочих па-

раметров. Например, часть описания приложения в SCA определяет программный компонент, который выполняет некоторую функцию. В этом описании могут присутствовать несколько различных программных реализаций (например, одна для запуска на ЦСП, другая — для ЦП Intel с ОС WxWorks и т.д.). Таким образом, благодаря современным возможностям совместимости приложение, которое можно было запускать только на ЦСП, теперь может работать и на универсальном ЦП. (Однако это не значит, что для переноса элемента с одной системы на другую не потребуется определенных усилий. Это лишь означает, что разработка, предназначенная для определенной системы, может быть размещена и сконфигурирована на этой системе без перенастройки всего приложения.) Поскольку SCA позволяет использовать многократные реализации приложений, новая реализация может быть развернута на уже имеющейся системе, которая оснащена ЦП, но не имеет ЦСП, при этом не требуется изменения остальных аппаратных компонентов. Такая же концепция применяется и к новому оборудованию.

1.2.4.4. SCA обеспечивает инфраструктуру для расширения и интеграции

SCA является базой для проектирования и разработки комплексных радиосистем. Она определяет пакет требуемых интерфейсов и алгоритмы, заложенные в инфраструктуре, и дает возможность проектировать и создавать приложения с улучшенными рабочими параметрами и характеристиками. С системой SCA, обеспечивающей высокоуровневое управление и контроль радиоресурсов, могут быть интегрированы технологии типа когнитивного радио. Например, несколько SCA-радиосистем, имеющих когнитивную карту их окружения и временной логики, могут согласовывать взаимную конфигурацию, повышая тем самым надежность связи. Это может достигаться путем изменения параметров существующих сигналов внутри системы или загрузки новых сигналов, согласованных с остальными радиосистемами.

1.2.4.5. SCA снижает объем единовременных затрат при проектировании системы

Благодаря наличию единого пакета управляющих программ и инструментов стандартизация на единой инфраструктуре ПО снижает затраты, связанные с разработкой систем в будущем. Это способствует оптимизации графика выполнения работ и снижению затрат на модернизацию. Важно разработать стандартную инфраструктуру, которая применима для различных систем, поддерживающих SCA. Последний пункт является основным для организации принципов разработки систем и использования спецификации SCA.

Теперь, после рассмотрения «плюсов» и «минусов» использования SCA, можно продолжить краткий обзор технических аспектов SCA. Остаток этой главы посвящен вопросам операционной среды SCA и структуры ее спецификации и станет основой для последующих глав.

1.3. Операционная среда

Архитектура программируемых средств связи JTRS определяет требования для единой операционной среды программируемого радио. Операционная среда включает в себя ядро SCA, брокер объектных запросов CORBA и операционную систему. Операционная среда определяет интерфейсы, правила, ограничения и процедуры, которые необходимо соблюдать для реализации SCA-совместимой радиосистемы.

Ядро SCA обеспечивает:

- пакет общих служб, используемых сигналами и другими приложениями;
- ПО для установки, настройки, управления и контроля сигналов;
- интегрированную файловую систему для выполнения общих операций с файлами, а также обеспечения доступа к множественным вычислительным платформам;
- интерфейсы устройств, которые обеспечивают общую абстракцию компонентов аппаратного обеспечения.

На рис. 1.4 ядро SCA, представленное как уровень управления системой, изображено по вертикальной оси. Сигналы и компоненты, образующие уровень приложений, представлены на горизонтальной оси.

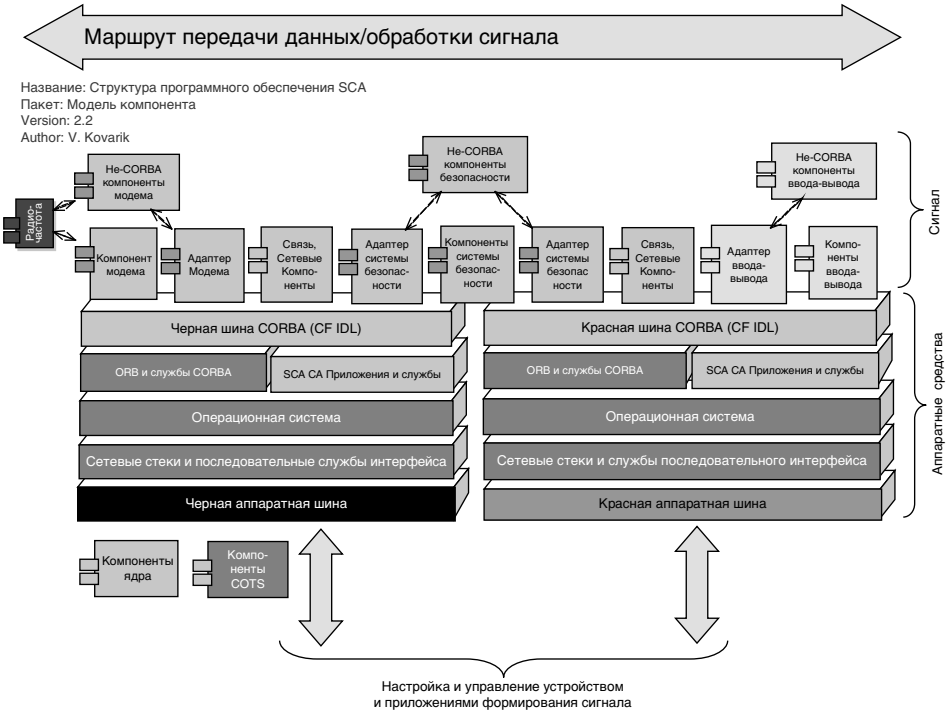

Рис. 1.4. Структура компонента формы сигнала SCA

1.3.1. Структурная организация

По своей структуре радиосистема SCA состоит из трех элементов: загрузка модели сигнала (Waveform Deployment), ядро (Core Framework) и профиль домена (Domain Profile). В свою очередь, каждый из трех указанных элементов можно рассматривать как физический и логический элементы. С точки зрения загрузки модели сигнала аппаратные средства являются физической структурой радиосистемы. Но сигналы реализуются с помощью ПО, которое выполняется на физических элементах.

Существуют два уровня логической структуры радиосистемы. Первый уровень включает в себя пакет компонентов, которые образуют приложение, реализующее сигнал или какую-либо другую службу в системе. Второй уровень — это приложение, которое обеспечивает интерфейс высшего уровня и управление пакетом компонентов.

Ядро SCA включает в себя все программные средства, необходимые для управления радиосистемой и развертывания приложений. Его так же можно рассматривать как физическую и логическую структуры. Физическая структура ядра обеспечивает высокоуровневое управление физическими устройствами в радиосистеме. Логическая структура обеспечивает ту же функцию для приложений, формирующих сигнал и других служб.

Профиль домена включает в себя XML-файлы, которые содержат описание аппаратных средств радиосистемы, структуру приложения, формирующего сиг-

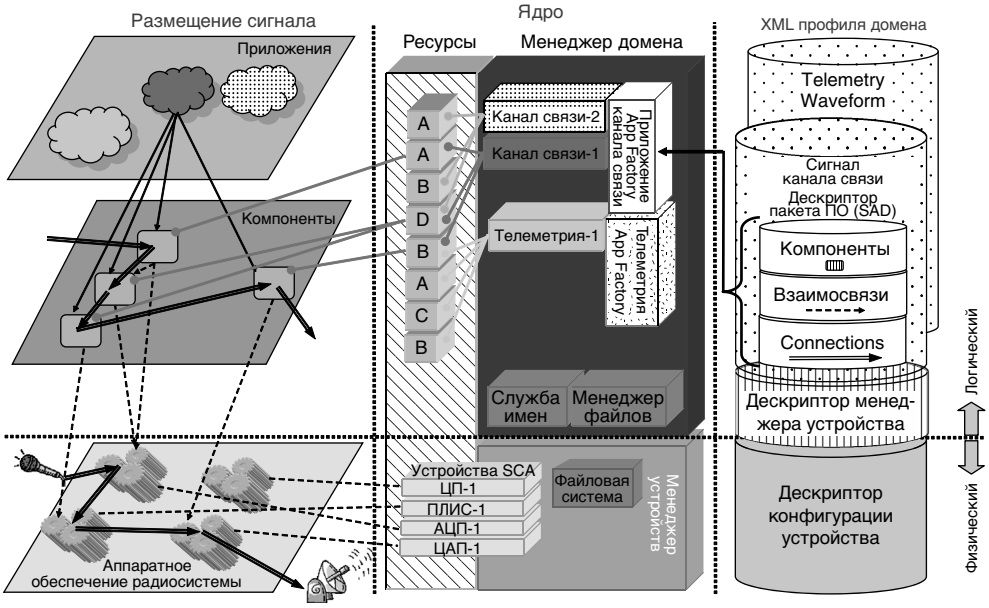

Рис. 1.5. Уровни абстракции в системе SCA

нал, взаимосвязи и зависимости программных компонентов и аппаратных ресурсов (рис. 1.5).

1.3.2. Ограничения интерфейса операционной среды

На рис. 1.6 показаны взаимосвязи между основными компонентами SCA-совместимой системы. На нижнем уровне находятся службы, обеспечиваемые ОС (которая формирует профиль среды приложения) как интерфейсы POSIX.

Профиль среды приложения (Application Environment Profile) предназначен для обеспечения ограниченного набора строго определенных системных вызовов, которые минимизируют влияние на код приложения. Это применимо лишь для ОС, работающей на ЦП, поскольку ЦСП и ПЛИС не имеют операционной системы. Тем не менее, некоторые процессоры ЦСП поддерживают ОС и CORBA. В этом случае для обеспечения доступа к функциям операционной системы необходимо использовать интерфейс POSIX. Все остальные компоненты (CORBA ORB, ядро SCA, компоненты без использования CORBA и драйверы устройств) могут иметь неограниченный доступ к операционной системе.

CORBA ORB обеспечивает единое связующее ПО для системы и имеет неограниченный доступ к базовой операционной системе. Ядро SCA — это реали-

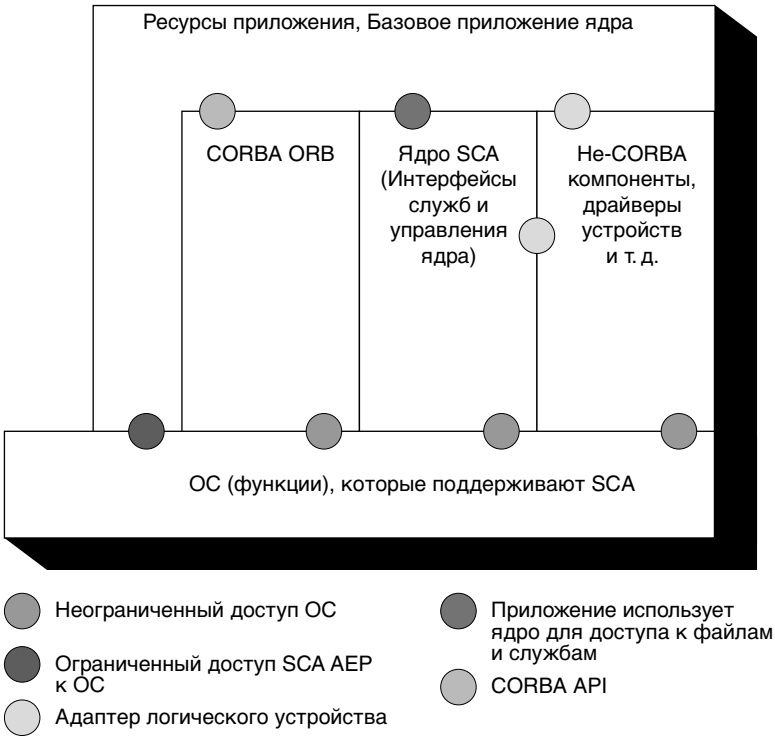

Рис. 1.6. Ограничения интерфейса SCA

зация спецификации SCA, определяющей необходимые компоненты и службы радиосистемы. Компоненты и драйверы устройств, не совместимые с CORBA, включают в себя низкоуровневые драйверы, подобные тем, которые поставляются производителем какого-либо устройства для различных ОС. Приложение состоит из набора компонентов и ресурсов, которые формируют рабочий сигнал. И наконец, ОС обеспечивает базовый набор зависящих от платформы служб.

Приложения CORBA ORB, ядро и драйверы устройств имеют неограниченный доступ к системным вызовам и службам ОС. Но, как показано на рис. 1.6, приложения имеют доступ только к набору системных вызовов POSIX. Это ограничение введено потому, что если приложение имеет ограниченный набор интерфейсов, то оно легче переносится (портируется) на другие платформы. В свою очередь, для обеспечения совместимости с SCA программная платформа радиосистемы должна поддерживать системные вызовы POSIX.

Несмотря на преимущества такого подхода для обеспечения совместимости при портировании сигнала приходится устранять многочисленные проблемы и ограничения POSIX. Это становится особенно ощутимо в том случае, если разработанное сигнальное приложение использует в качестве одного из функциональных узлов цепи формирования сигнала ЦСП или ПЛИС.

1.4. Структура спецификации SCA

Спецификация SCA состоит из трех основных частей:

- спецификация программируемых средств связи (JTRS-5000SCA),
- дополнения API (JTRS-5000API),
- дополнения по безопасности (JTRS-5000SEC).

Спецификация SCA является основой для разработки SCA-совместимых радиосистем. Она определяет требования к операционной среде и основным функциональным характеристикам. В данной книге основное внимание уделено содержанию спецификации SCA.

В дополнениях API (JTRS-5000API) содержатся инструкции и требования по разработке модульных и портативных компонентов приложений. Некоторые аспекты дополнений API рассмотрены в этой книге. Однако подробное рассмотрение дополнений API и вопросов создания портативных приложений потребует описания значительно большего числа подробностей, чем позволяет объем данной книги.

Дополнение по безопасности определяет специальные требования к безопасности и API, относящиеся к проектированию и построению защищенной радиосистемы первого типа. Некоторые разделы книги посвящены системе безопасности, поскольку это важно для понимания других разделов. Так же,

как и для дополнений API, подробности, связанные с вопросами защиты информации при создании SCA-совместимых радиосистем, остаются за рамками данной книги.

Разд. 3 спецификации SCA описывает требования к определению архитектуры базового программного обеспечения.

Организационная структура разд. 3 показана на рис. 1.7. В подразд. Operating Environment (Операционная среда) рассматриваются службы и ядро SCA. В разд. 3.2 содержатся описание интерфейсов приложений и функциональные требования. Разд. 3.3 определяет требования к интерфейсу устройства SCA, а в разд. 3.4 перечислены общие требования.

Операционная среда (разд. 3.1 спецификации SCA) определяет требования, которым должна удовлетворять SCA-совместимая система. Она содержит общие компоненты ядра — менеджер домена (Domain Manager), службу регистрации (Log Service), компоненты, необходимые CORBA, и др.

В разд. 3.2 определены требования, которым должны соответствовать приложения SCA. Эти приложения позволяют полностью реализовать сигнал, поэтому раздел будет более всего интересен разработчикам сигнальных приложений. Тем не менее, некоторые связанные с приложениями аспекты представлены как компоненты ядра.

Управление устройствами радиосистемы и их настройка осуществляются через интерфейс логического устройства. Для того, чтобы успешно интегрировать устройство в SCA-систему, производитель устройств и/или системный интегратор должен обеспечить реализацию этого интерфейса. Эти требования определены в разд. 3.3. Содержимое разд. 3.1—3.3 и основные области функциональных требований будут подробно описаны в последующих главах. Общие

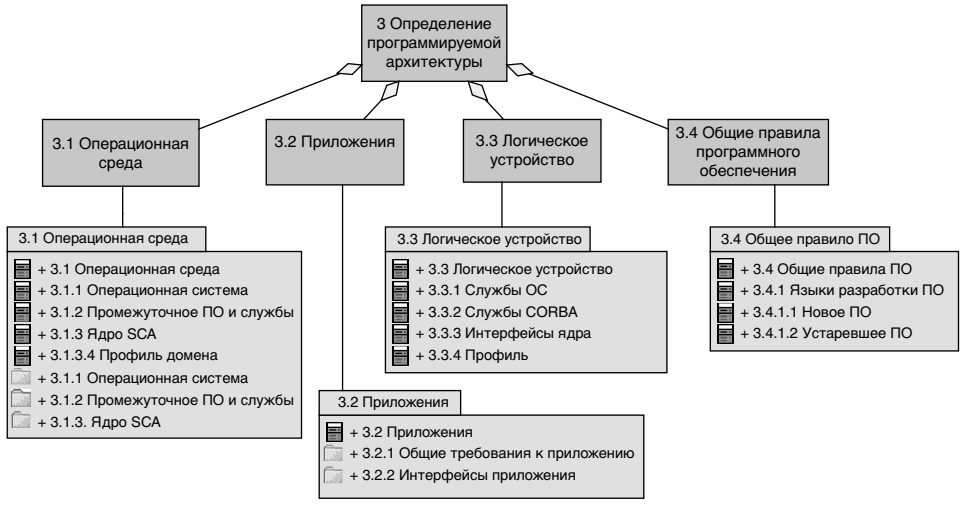

Рис. 1.7. Структура спецификации SCA

требования к ПО (разд. 3.4) и другие нефункциональные требования будут описаны в этом разделе.

На рис. 1.8. показано, как соотносятся спецификация SCA и ее реализация. Спецификация SCA обеспечивает интерфейс и характерные высокоуровневые параметры, реализуемые ядром. Рисунок также иллюстрирует разработанные группой OMG стандарты, которые рассмотрены в спецификации SCA.

Группа интерфейсов ядра и спецификации поведения (behavioral specifications) находятся в пакете ядра. Профиль домена задает XML-файлы, которые используются для описания основных компонентов аппаратного и программного обеспечения и взаимосвязей, необходимых для загрузки сигнала.

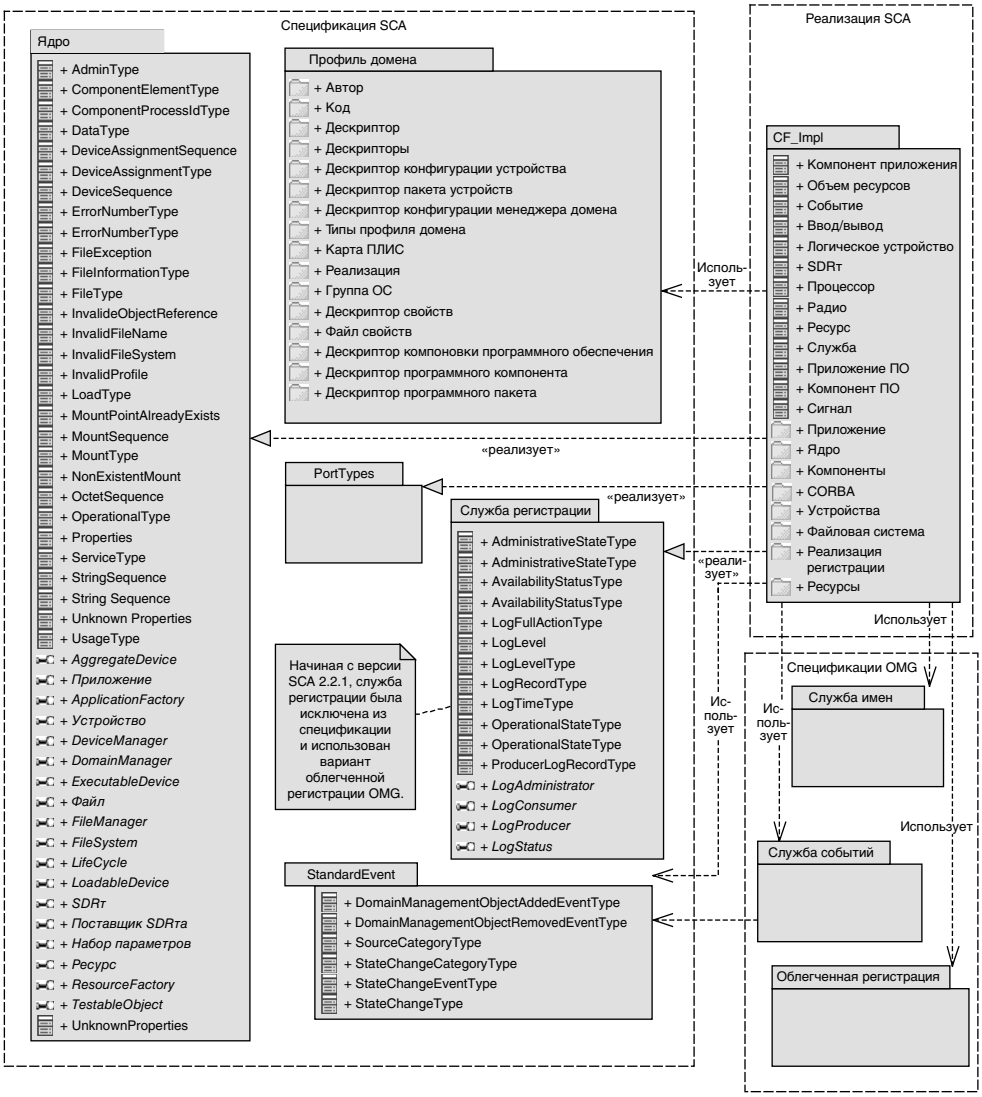

Рис. 1.8. Спецификация SCA в зависимости от реализации

Существует несколько толкований термина «профиль домена». Некоторые подразумевают под ним набор XML-файлов, по мнению других, это обработанные структуры данных из этих файлов, третьи же называют профилем домена внутренние структуры данных, которые содержат сведения о домене SCA-системы.

Авторы данной книги придерживаются точки зрения, что XML-файлы представляют собой удобочитаемую версию профиля домена, а внутренние структуры данных, созданные как часть обработки XML-профиля, формируют внутренний профиль домена, используемый ядром SCA в процессе создания сигнала. Иерархическая древовидная структура, сгенерированная синтаксическим анализатором (парсером) рассматривается как промежуточная структура данных, которая преобразует текстовые XML-файлы в каноническую форму, позволяющую быстрое извлечение из них необходимой для загрузки сигнала информации. Несомненно, можно пропускать обработку структуры XML-файла и непосредственно использовать его как представление внутреннего профиля домена. Но такая форма представления данных будет неэффективна при использовании строгих ограничений (constraint enforcement), которые используются компонентом ядра SCA ApplicationFactory при создании экземпляров приложения.

Все вышеперечисленные абстракции базируются на единых службах, которые обеспечивают важнейшие функции для всех компонентов SCA:

- **служба имен (name service)** — позволяет компоненту радиосистемы найти требуемую службу или приложение и подключиться к нему;
- **служба событий (event service)** — обеспечивает асинхронный механизм для трансляции компонентами событий в канал событий и регистрацию компонентов для получения этих событий;
- **служба регистрации (log service)** — обеспечивает базовую функцию регистрации внутри радиосистемы SCA, которая позволяет компонентам создавать записи, описывающие какие-либо действия или состояния, снабженные временными метками, и сохранять их в файле с возможностью последующего извлечения;
- **файловая служба (file service)** — возможно, это одна из важнейших служб, которая обеспечивает единую абстракцию файловой системы для всех компонентов SCA-системы независимо от фактической реализации ОС и файловой системы.

Первая глава книги организована в соответствии с описанными выше высокоуровневыми абстракциями. Во второй главе рассматриваются общие службы фреймворка, обеспечивающие основу для компонентов и ссылок, используемых в описании остальных логических интерфейсов. После описания общих служб в последующих главах рассматриваются описанные выше логические абстракции, начиная с менеджера ресурсов и заканчивая менеджером домена.

1.5. Выводы

Спецификация SCA обеспечивает независимую от реализации инфраструктуру для реализации и развертывания приложений. Общие требования SCA определяют набор аппаратных и программных требований, которым должна соответствовать система. В следующей главе представлен концептуальный обзор работы SCA-совместимой системы, использующий модельный принцип прецедентов, или сценариев использования (Use case) унифицированного языка моделирования UML.

ГЛАВА 2

СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Функционирование системы удобно описывать с помощью прецедентов (сценариев использования), которые описывают общее поведение системы при взаимодействии с пользователем либо с внешней средой. В этом разделе описываются различные прецеденты SDR. Они предназначены для подробного анализа рабочих параметров абстрактной SCA-совместимой радиосистемы. Детальное описание какой-либо конкретной радиосистемы требует более широкого набора операционных сценариев и прецедентов.

На рис. 2.1 показаны субъекты, имеющие отношение к радиосистеме, и различные действия, которые они выполняют. *Конечный пользователь радиосистемы* — это обычный оператор радиостанции. Он по сути работает с радиосистемой на базовом уровне оборудования, например, включает и выключает питание или инициализирует систему. На более абстрактном уровне пользователь взаимодействует с радиосистемой, управляя сигнальными приложениями. На этом уровне начинает проявляться уникальность программно-определяемого радио. Пользователь может конфигурировать систему для загрузки и запуска сигнала в динамическом режиме, то есть радиостанция по требованию может быть (пере)настроена для выполнения компонентов, поддерживающих различные сигнальные приложения.

Настройку, интеграцию, тестирование радиосистемы и установку сигнальных приложений выполняет *радиоинженер*.

В следующих параграфах представлен качественный анализ необходимых действий, рассмотренных в каждом из показанном на рис. 2.1 прецеденте. Цель этого подхода — обеспечить единую модель поведения и взаимоотношений для ключевых функций SCA-совместимой радиосистемы и показать все необходимые шаги и режимы работы. Так как мы занимаемся исследованием конкретных поведенческих факторов компонентов ядра SCA, этот же подход будет применен и в следующих главах. Необходимо отметить, что диаграммы, представленные в следующих разделах, в общем виде иллюстрируют взаимодействия между процессами и не являются точным отображением UML-модели этих взаимодействий.

2.1. Запуск системы

Прецедент «Пуск» (Startup) выполняется при переводе радиосистемы из выключенного состояния во включенное, когда выполняются включение и ини-

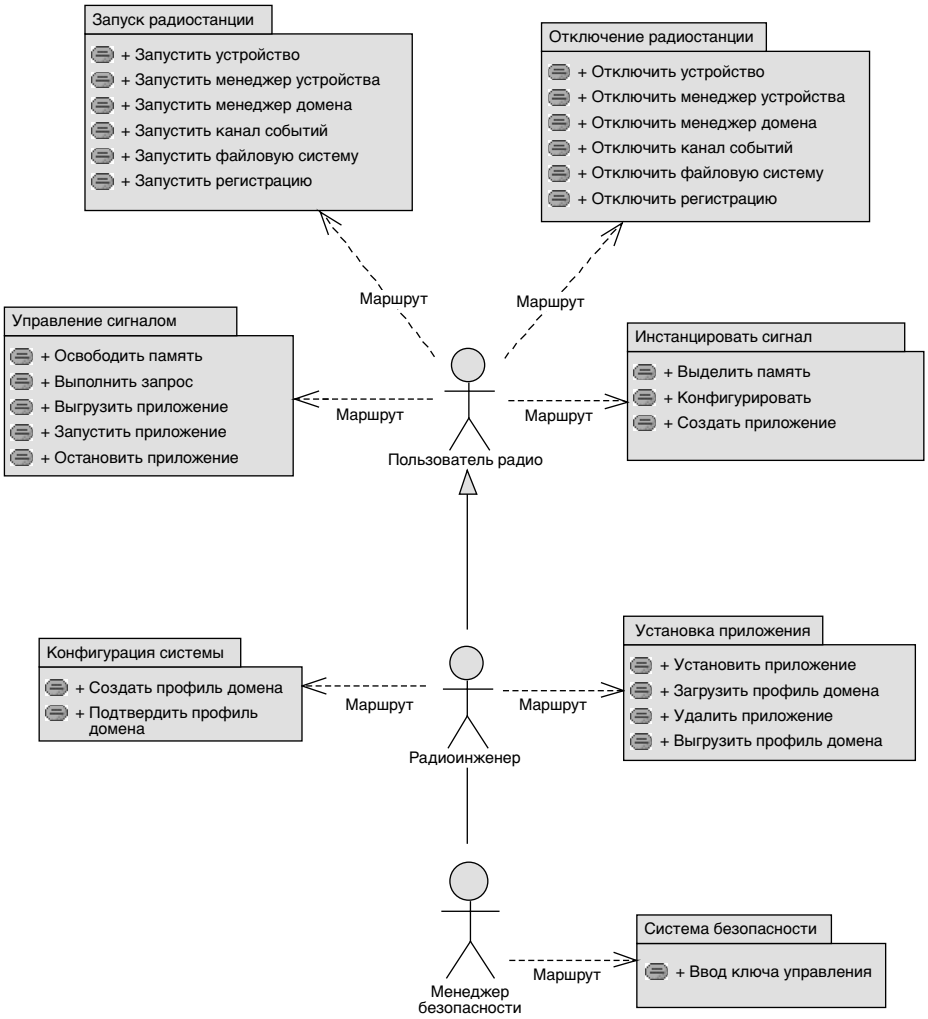

Рис. 2.1. Функции и действия пользователей радиосистемы

циализация всех аппаратных компонентов системы и загрузка и инициализация ядра SCA.

Рис. 2.2 показывает основные связи между прецедентами, образующими стартовую последовательность.

Прецедент «Start Device Manager» (запустить менеджер устройств) описывает запуск устройства, которое обычно называют «узел» (node). Здесь этот термин используется для обозначения аппаратного компонента (например, плата с интерфейсом VME или Compact PCI), который включает в себя одно или несколько физических устройств, выполняющих какие-либо необходимые для реализации SCA-совместимой радиосистемы вычисления. Как правило, в качестве исходного или загрузочного узла используется одноплатный компьютер (SBC) или иное устройство с центральным процессором и операционной системой.

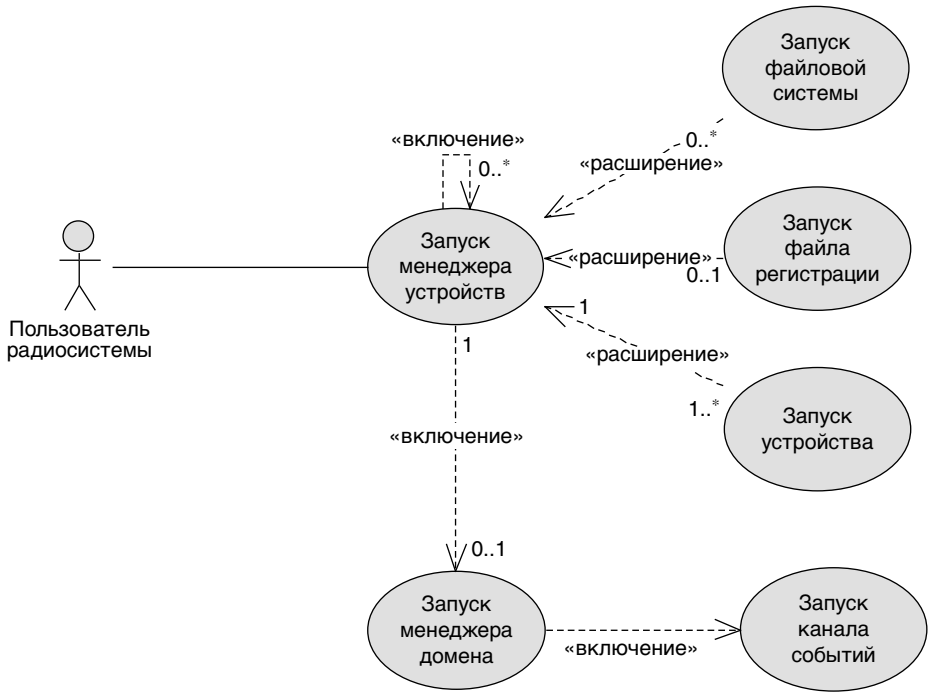

Рис. 2.2. Запуск радиосистемы

Основным условием запуска системы SCA является наличие загрузочного устройства и его менеджера (Device manager), который управляет запуском и инициализацией ядра SCA и связанных с ней компонетов. Начав работу, менеджер устройств запускает ряд служб и приложений (рис. 2.3), начиная с инициализации файловой системы (File system). Файловая система выделяет для менеджера устройств и связанных с ним компонентов некоторый объем памяти. Затем менеджер устройств запускает службу регистрации (Log Service), которая дает возможность записывать системные сообщения, предупреждения и неисправности всех аппаратных и программных компонентов системы SCA. Далее, менеджер устройств запускает менеджер домена (Domain manager), который служит основным хранилищем (репозиторием) и пунктом управления системой. После этого менеджер устройств запускает все связанные с ним компоненты.

Рассмотренный выше принцип стал использоваться лишь в текущей версии SCA. При текущем подходе невозможно гарантировать то, что компонент, в случае его инициализации клиентом, будет полностью инициализирован и готов к приему поступающих запросов CORBA. Например, менеджер устройств на загрузочном устройстве может запускать менеджер домена, но при попытке регистрации последний может не загрузиться полностью и не инициализироваться. Таким образом, только при наличии в компонентах ядра SCA встроенного механизма автоматического перезапуска они, скорее всего, будут запущены без сбоев.

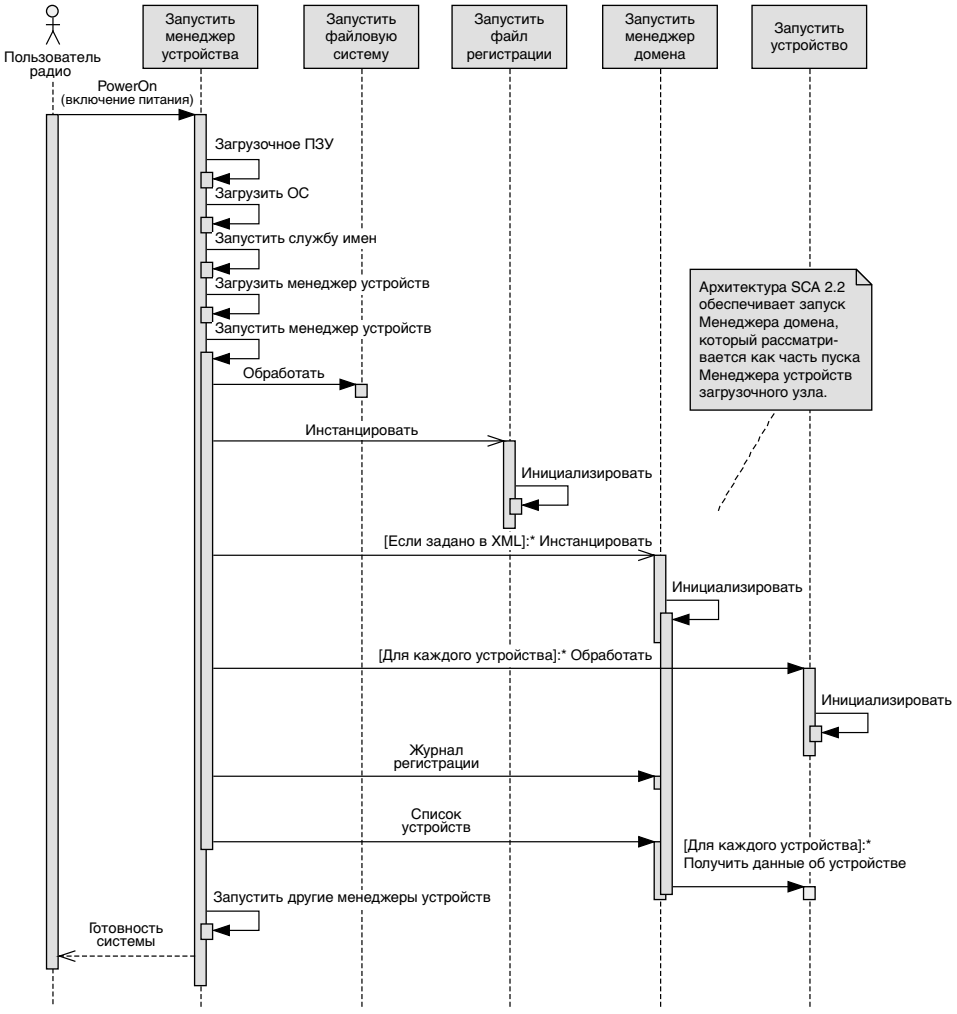

Рис. 2.3. Последовательность операций при запуске радиосистемы

После включения питания загрузочный узел задействует загрузочное ПЗУ (Boot ROM), которое запускает необходимые начальные процедуры и инициализирует оборудование. После этого запускается операционная система. Она обеспечивает основные службы и функции для загрузки и управления базовыми программными компонентами SCA.

После того как загрузилась ОС, система готова для загрузки компонентов ядра SCA. Первый элемент, который необходимо загрузить, это служба имен CORBA, которая позволяет приложениям определять местоположение доступных служб и других приложений. Этой службой регистрируются компоненты ядра SCA, таким образом, она должна быть доступна до инициализации этих компонентов.

На данном этапе выполняется загрузка программной реализации менеджера устройств. После инициализации он в первую очередь должен считать дескриптор

конфигурации устройства (DCD), связанного с менеджером устройств. DCD — это XML-файл, содержащий информацию о конфигурации и параметры для запуска устройства. Синтаксическая структура дескриптора DCD и XML-файлов рассматривается во второй части книги, некоторые примеры приведены в третьей.

Считанный и обработанный менеджером устройств DCD определяет дополнительные программные компоненты и устройства, которые должны быть запущены в процессе общей инициализации менеджера устройств. Как правило, одна из первых задач, выполняемая менеджером устройств, это инициализация файловой системы (ФС). ФС SCA — это абстрактное, основанное на CORBA представление некой конкретной ФС, реализованной на базовом аппаратном обеспечении и операционной системе. Обычно менеджер устройств инициализирует лишь одну ФС, но, в зависимости от типа конфигурации аппаратного обеспечения и операционной системы, а также характеристик базового оборудования, ассоциированного с менеджером устройств, могут быть инициализированы несколько ФС (или же инициализация ФС не производится вообще).

Следующий инициализируемый компонент — это канал событий (Event Channel), обеспечивающий механизм асинхронных уведомлений, благодаря которому компоненты могут публиковать сообщения об изменении своего статуса и других событиях. Компоненты, которым необходимо считывать определенные сообщения, должны лишь подписаться на соответствующий канал. Функциональные характеристики канала событий в SCA перечислены в документе OMG Document formal/01-03-01: Event Service, v.1.1.

После создания канала событий менеджер устройств запускает службу регистрации. До появления версии 2.2 SCA служба регистрации была определена как часть спецификации SCA. Начиная с версии SCA 2.2.1 в архитектуре SCA указывается спецификация упрощенной регистрации OMG. Обе службы аналогичны по функциональности и используют одинаковый интерфейс IDL. Служба регистрации обеспечивает кратковременный, находящийся в ОЗУ файл данных, который может быть доступен любому компоненту SCA. Файл данных постоянно использует буферную память.

После запуска службы регистрации, если это указано в XML-файле DCD, менеджер устройств запускает менеджер домена. Последний обеспечивает хранилище верхнего уровня для всех устройств и программных компонентов, которые входят в домен радиосистемы, и управляет ими.

На этом этапе менеджер устройств запускает и инициализирует все устройства, указанные в XML-файле DCD. После этого менеджер устройств регистрирует через менеджер домена каждое запущенное устройство. Для всех зарегистрированных устройств менеджер домена запрашивает параметры устройства и информацию о его конфигурации.

Наконец, если это необходимо для старта системы, менеджер устройств загрузочного узла может запустить дополнительные менеджеры устройств. Каждый последующий запуск менеджера устройств осуществляется по той же схеме, что и запуск менеджера устройств загрузочного узла. Однако далее уже не нужно запускать менеджер домена или канал событий, поскольку для работы системы требуется лишь однократный запуск этих компонентов. Необходимость же инициализации ФС и службы регистрации зависит от ресурсов и требований к радиосистеме.

2.2. Завершение работы

Весьма интересен тот факт, что процесс отключения радиосистемы не описан подробно в спецификации SCA. Рассуждая логически, можно предположить (как показано на рис. 2.4), что менеджер домена осуществляет управление всей радиосистемой и именно он выдает команду отключения радиостанции. Это итеративная операция, во время которой зарегистрированные менеджером домена менеджеры устройств получают от него команду на отключение (рис. 2.5). В свою очередь, каждый менеджер устройств отправляет всем связанным с ним устройствам команду завершения их работы. После этого менеджер устройств останавливает связанную с ним файловую систему и службу регистрации, если таковая запущена.

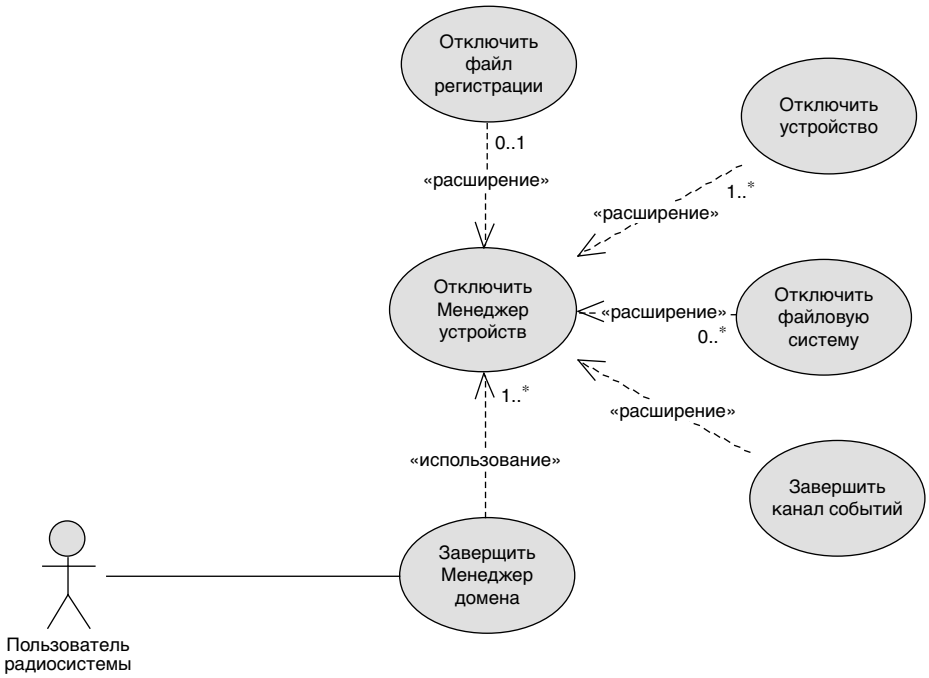

Рис. 2.4. Отключение радиосистемы

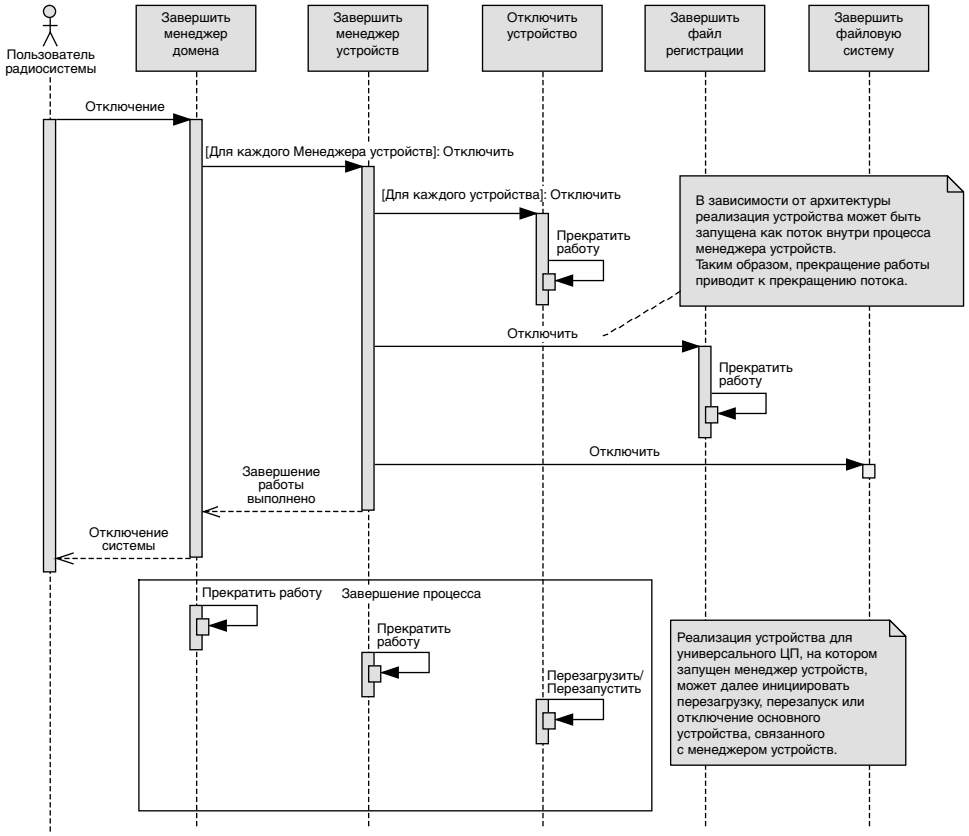

Рис. 2.5. Последовательность отключения компонентов радиосистемы

Завершив все указанные действия, менеджер устройств прекращает свою работу. Следует иметь в виду, что после остановки менеджера устройств и связанных с ним компонентов получить ответный сигнал от устройств невозможно, так как они перестают существовать в системе.

Как только все менеджеры устройств получает команду об отключении, менеджер домена завершает работу. Но так как в процессе отключения устройств они могут посылать уведомления об ошибках или иных событиях, менеджер домена должен ждать некоторое время для того, чтобы проконтролировать отключение всех устройств и только после этого закончить свою работу.

Поскольку один из менеджеров устройств работает как загрузочный узел и обычно он же запускает менеджер домена, он может использовать интерфейсные запросы к ОС на аппаратное отключение или перезагрузку. Независимо от того, реализовано это или нет, менеджер устройств для процессора, на котором запущен менеджер домена, должен ждать до тех пор, пока менеджер домена прекратит работу. После того как это произойдет, менеджер устройств выполняет собственную процедуру отключения. Таким образом, когда менеджер

устройств получает запрос от менеджера домена на отключение, он откладывает процесс фактического отключения до тех пор, пока менеджер домена и прочие процессы SCA не завершат процедуры своего отключения. Только тогда менеджер устройств завершает свою работу и/или обращается к ОС с запросом на перегрузку или перезапуск.

2.3. Установка/удаление приложения

При установка приложения в SCA-совместимой радиосистеме считываются XML-файлы, описывающие необходимые для размещения сигнала в системе аппаратные и программные компоненты,. Ключевой момент здесь то, что сигнальное ПО не загружается в оборудование. На этом этапе требуется лишь список устройств, на которые загружаются программные компоненты, и связей, объединяющих их в функциональное приложение.

Результат установки приложения — загрузка файла дескриптора компоновки программного обеспечения (Software Assembly Descriptor — SAD). Файл дескриптора SAD — это высокоуровневый XML-файл, который определяет компоненты, схемы соединений, а также ограничения для заданного сигнального приложения.

Термин «профиль домена» иногда используется для обозначения и XML-файлов, и внутренних структур данных, представляющих содержание XML-файлов в удобной для использования компонентами ядра форме. Хотя в спецификации SCA под профилем домена подразумеваются именно XML-файлы, обрабатывать их при каждой загрузке сигнала затратно с точки зрения вычислительных ресурсов.

Большинство реализаций ядра SCA преобразуют информацию в легкий для обработки «машинный» формат. Это может быть XML-дерево или объектная модель документов (DOM), созданная парсером XML. Однако при обработке парсером XML-дерева теряются семантические связи между компонентами профиля. Это приводит к необходимости обхода вершин дерева при создании экземпляра (инстанциации) приложения.

В результате некоторые реализации ядра SCA создают внутренний набор структур данных, которые обеспечивают более эффективное представление для восстановления и обновления при создании и уничтожении экземпляра приложения.

При удалении приложения соответствующая информация профиля домена также удаляется. (В спецификации SCA изначально было указано, что XML-файлы с информацией профиля домена также должны быть удалены из файловой системы. Однако в версии SCA 2.2.1 эти XML-файлы сохраняются.)

Такое требование создает проблему для тех ядер SCA, которые используют внутренние структуры данных для представления проанализированного и установленного сигнального приложения, так как удаление приложения подразуме-

вает простое физическое удаление внутренних структур данных, представляющих профиль сигнала.

Кроме этого, удаление файлов вызывает проблемы с операционной системой в случае, если конечный пользователь обеспечивает возможность деинсталляции приложения. Внутренние структуры данных не удаляются, так как в противном случае обработка радиосигнала была бы невозможна до повторной загрузки XML-файлов (т. е. до повторной инсталляции приложения формы сигнала). Другими словами, это привело бы к удалению исходных XML-файлов, описывающих профиль.

Некоторые реализации SCA позволяют указывать, удалять ли физически XML-файлы, связанные с сигналом, или это относится лишь к внутренним структурам данных. Это дает возможность деинсталлировать приложение, освободив тем самым внутренние ресурсы и память для других приложений, в то же время давая возможность при необходимости переустановить сигнал.

При установке сигнального приложения выполняются считывание и обработка (парсинг) набора XML-файлов, задающих структуру сигнала. Прецедент «Установка приложения» (рис. 2.6) создает базовые внутренние структуры данных, определяющих необходимые для загрузки сигнального приложения компоненты и аппаратные средства. «Загрузка профиля домена» обеспечивает нужную

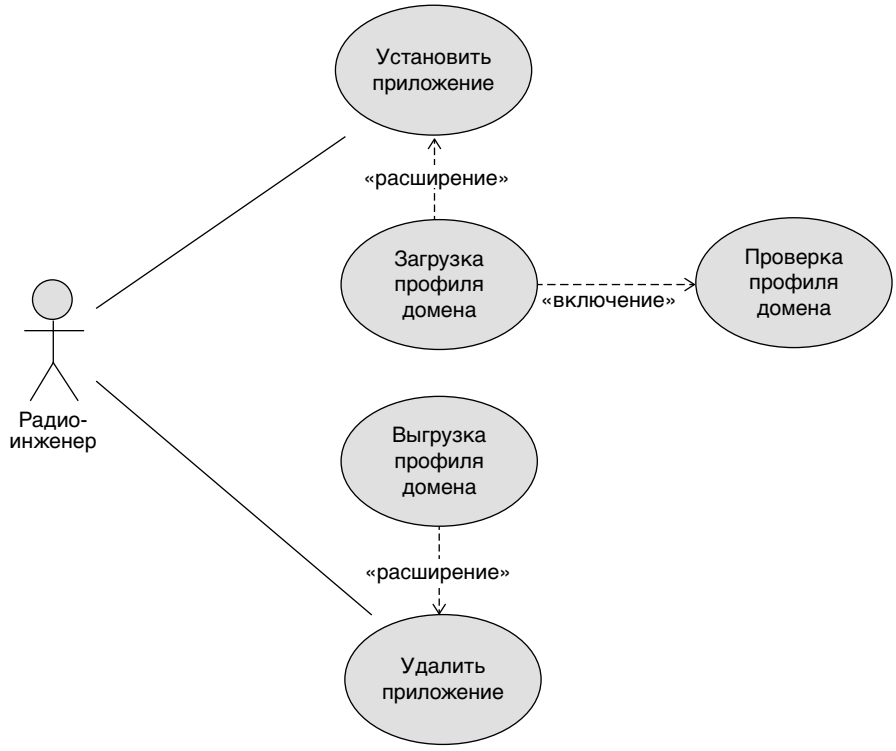

Рис. 2.6. Установка/удаление приложения

для создания внутренней модели профиля сигнала информацию, извлеченную из XML-файлов.

Прецедент «Проверка профиля домена» выполняет анализ обработанных XML-файлов в зависимости от определения типа документа (Document Type Definition, DTD), указанного в спецификации SCA. Последовательность процедур по установке/удалению приложения показана на рис. 2.7.

Деинсталляция приложения влечет за собой удаление из системы всех внутренних структур данных с информацией о профиле сигнала и, возможно, файлов XML, которые образуют профиль. Нужно отметить, что хотя удаление файлов XML заявлено как обязательная функция, смысла в этом нет, так как для переустановки приложения требуется повторная загрузка XML-файлов в систему.

Кроме этого (как показано на рис. 2.6), деинсталляция сигнального приложения не влияет на создание экземпляра сигнала. Хотя на первый взгляд концепция создания экземпляра деинсталлированного сигнального приложения выглядит противоречивой, она не лишена здравого смысла. После создания экземпляра сигнала приложению больше не требуется информация о профиле — так как информация о профиле использовалась в процессе создания. В результате все необходимые компоненты уже загружены, а ресурсы распределены. Поэтому созданный экземпляр сигнала можно запускать, останавливать, конфигурировать и выполнять с ним любые другие действия, допустимые для созданного экземпляра приложения и не требующие доступа к информации профиля.

Это позволяет создавать экземпляры сигнала, а затем их удалять, освобождая оперативную память и другие ресурсы, что может быть полезным, например, при

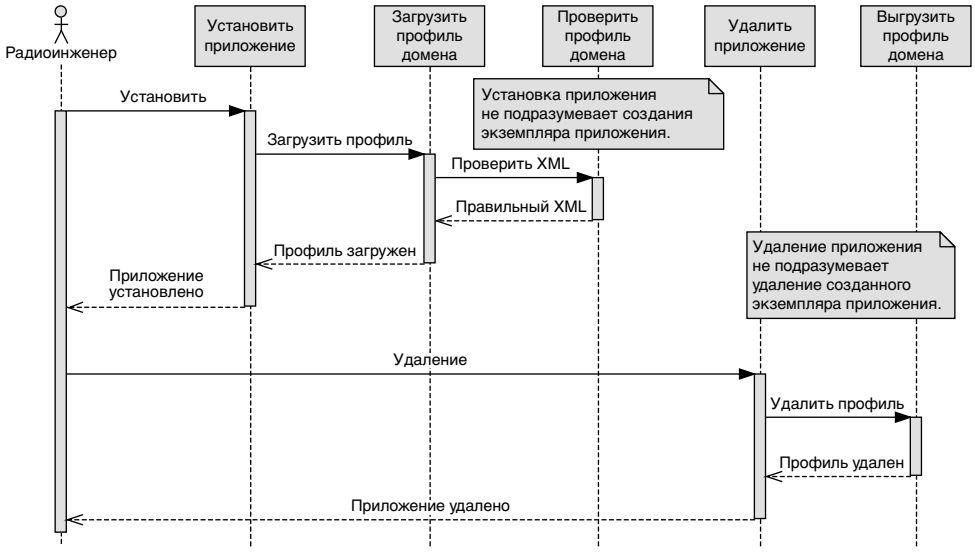

Рис. 2.7. Последовательность выполнения процедур установки/удаления приложения

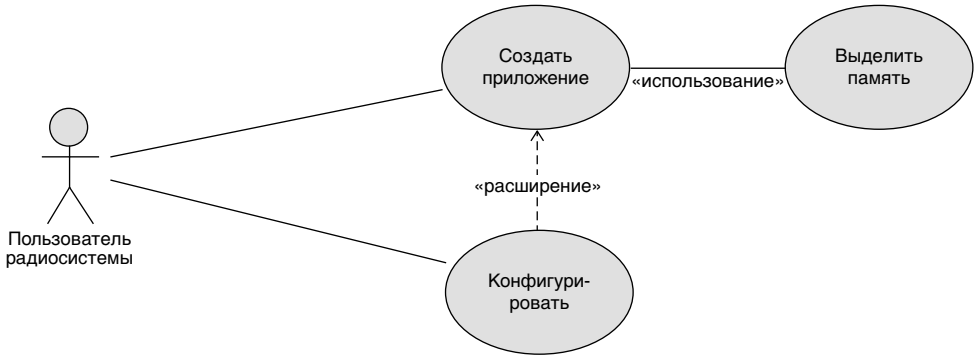

Рис. 2.8. Создание экземпляра сигнального приложения

установке другого сигнального приложения. Но если сигнал деинсталлирован (даже если созданные экземпляры приложения продолжают работать, не имея доступа к информации профиля), то выполнить создание новых экземпляров сигнала невозможно до тех пор, пока сигнал не будет заново установлен, обеспечив необходимую информацию профиля.

Аналогичным образом, если радиостанция отключилась, при включении системы менеджер домена переустанавливает все сигнальные приложения, которые были установлены на момент отключения. Если же сигнал был удален (даже если перед отключением его реализация была запущена), то менеджер домена не сможет его переустановить.

2.4. Создание экземпляра приложения (инстанциация)

После установки приложения выполняются необходимые для его инстанциации действия, а именно: загрузка программных компонентов и установка взаимосвязей между ними. Процесс применения информации профиля домена для инстанциации сигнала выполняется как часть прецедента «Создание экземпляра приложения».

Процесс создания экземпляра приложения имеет два важных аспекта. Его основная функция — загрузка программных компонентов. Но предварительно для этого необходимо распределить память устройств, на которые устанавливаются компоненты ПО (см. рис. 2.9).

Прецедент «Выделение памяти» выполняет проверка наличия в устройстве объема памяти, достаточного для загрузки программного компонента, и в случае использования ЦП — для работы компонента. Компонент может использовать определенный объем памяти или минимальное количество вычислительных циклов. При использовании ПЛИС может потребоваться введение

дополнительных логических цепей. Проблемы, связанные с выделением памяти, более подробно изложены в шестой главе, где рассматривается инстанциация приложения.

Часто в процессе создания экземпляра приложения связанным с компонентами и устройствами параметрам присваиваются определенные или случайные начальные значения. Эти значения определены в XML-файлах профиля домена и при использовании отображаются в виде внутреннего профиля домена. После инстанциации значения этих параметров можно изменить посредством операции конфигурирования. Это позволяет пользователю создавать экземпляр приложения и перед его запуском и использованием настраивать его параметры в зависимости от текущих требований. Например, экземпляр сигнала может быть создан с установленной по умолчанию частотой перескока, но в зависимости от конкретной ситуации использования данного сигнала может потребоваться изменение этого параметра. В этой ситуации может быть использована возможность изменения параметров сигнала уже после его создания.

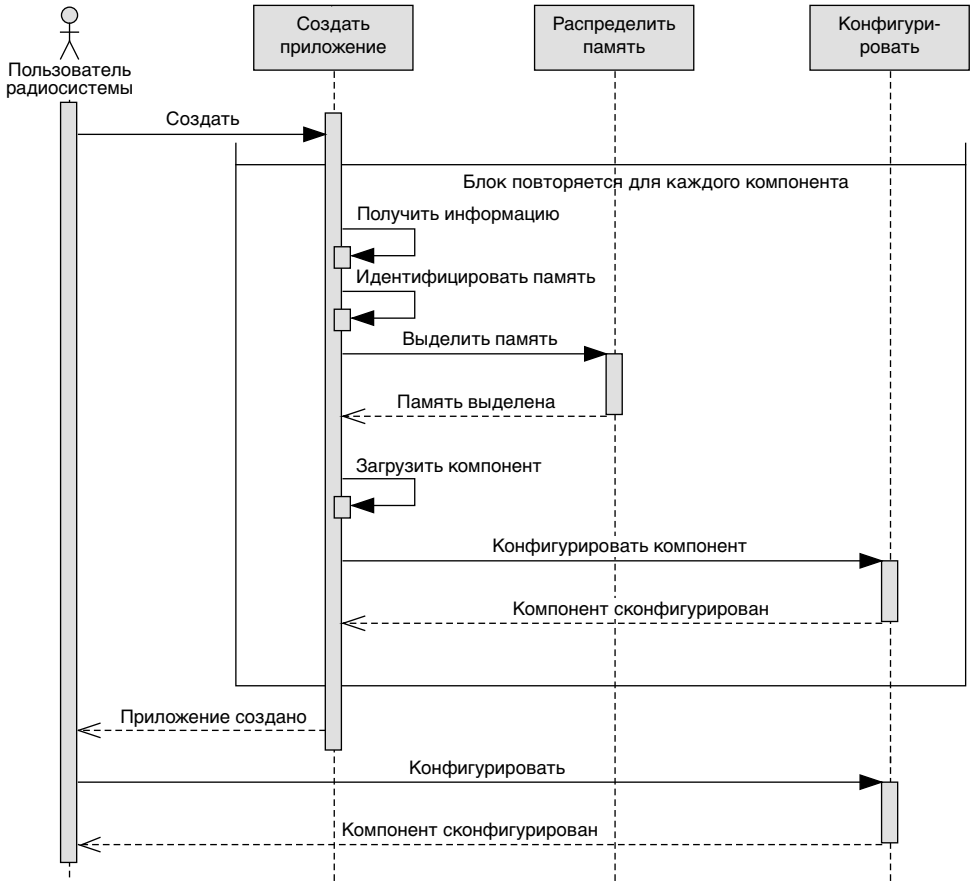

Рис. 2.9. Последовательность создания экземпляра сигнала

2.5. Управление приложением

После инстанциации приложение готово к использованию и может управляться пользователем. Он может запускать приложение, останавливать его, запрашивать и изменять параметры приложения и любых его компонентов, а также удалять компонент из системной памяти — «освобождать» его (Release application), см. рис. 2.10 и 2.11. Следует иметь в виду, что удаление сигнала с помощью операции «освобождения» — это удаление созданного экземпляра сигнала, а не деинсталляция его профиля (процесс, описанный в предыдущей части).

Запуск приложения — это процесс, во время которого созданное (то есть загруженное в память системы и с установленными взаимосвязями между компонентами) приложение получает команду начать обработку сигнала. По сути, запуск приложения инициирует движение потока данных для определенного сигнала в системе. Но это относится лишь к конкретному экземпляру сигнала, к которому применяется команда запуска. Все остальные экземпляры данного сигнала остаются в том же состоянии, в котором они были до подачи команды.

Если рассматривать загрузку сигнала в устройство, выполняющие последовательность команд (например, ЦП), этот процесс может показаться несколько

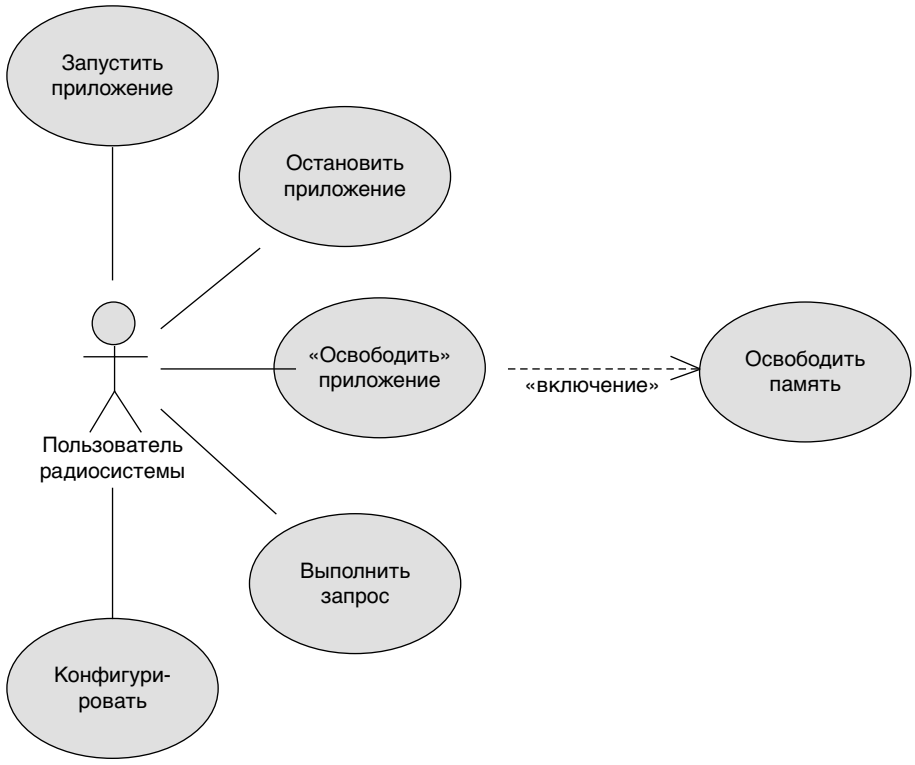

Рис. 2.10. Управление приложением

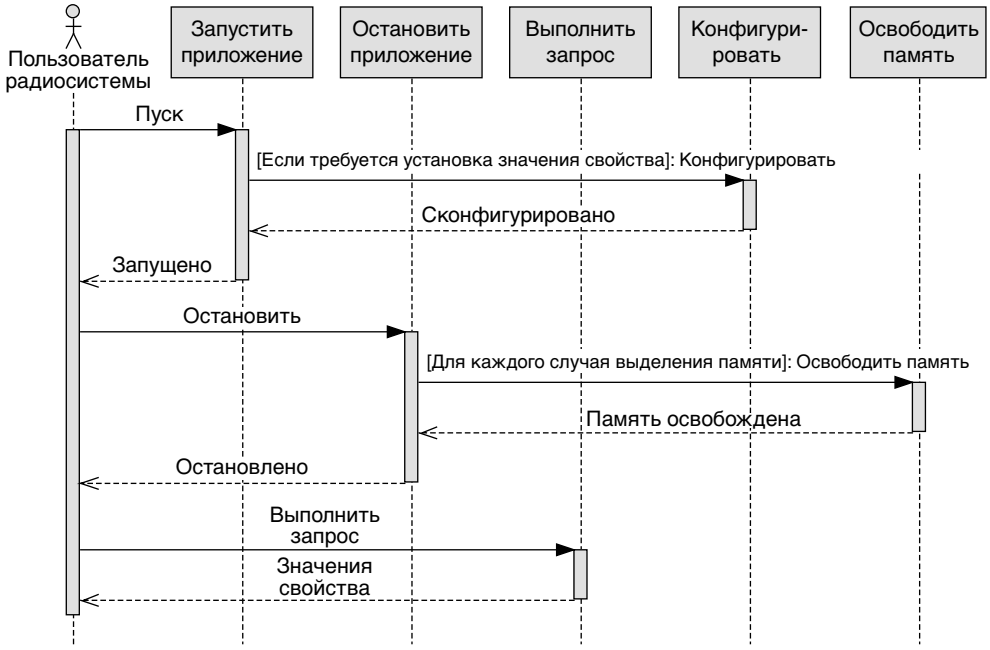

Рис. 2.11. Управление приложением радиосистемы

непонятным, так как устройство уже работает, то есть выполняет некие команды ОС. Но дело в том, что сигнальное приложение при этом находится в состоянии покоя и не обрабатывает поток данных.

В распоряжении пользователя есть также команда остановки. При выполнении этой команды сигнальное приложение переходит в состояние покоя, при котором все компоненты ПО остаются загруженными и подключенными, но поток данных сигнала не обрабатывается.

После того как сигнал установлен, к сигнальному приложению и его компонентам могут быть применены операции запроса (Query) и настройки (Configure). Они могут быть выполнены во время запуска или остановки сигнального приложени.

Разработчик сигнального приложения решает сам, будут ли выполняться команды запроса и настройки во время работы приложения. Например, некоторые значения параметров могут быть доступны только в том случае, когда сигнальное приложение остановлено, тогда как остальные свойства могут быть доступны в любое время.

И, наконец, установленный сигнал может быть «освобожден». Данная команда выполняется после остановки сигнального приложения и указывает сигналу удалить себя из памяти. Во время этого процесса на устройствах, выполняющих функции ведущих узлов, выполняется освобождение памяти. Эта функция крайне важна для работы системы, так как освобожденная память становится доступной