Рыжов В. С.

ПОСТРОЕНИЕ РАСПРЕДЕЛЕННЫХ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ ИНТЕГРИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ПРЕДПРИЯТИЯ

ВВЕДЕНИЕ

В данной работе рассматривается модель многоуровневой информационной системы, интегрирующей разнородные классы объектов различной конфигурации и степени сложности. Предложено архитектурное решение, позволяющее динамически расширять действующую систему, подключая к ней объекты как уже имеющихся в ней классов, так и объекты новых классов. В работе также обобщается опыт практической реализации модели в виде распределенного приложения. Анализируется опыт эксплуатации разработанных систем, включающих в себя подсистемы контроля доступа на объект, энергоснабжения, пожарной безопасности, видеонаблюдения и других систем жизнеобеспечения. Работа затрагивает следующие аспекты построения систем обеспечения безопасности и жизнедеятельности предприятия.

1. Интеграция разнородных подсистем в единую систему.
2. Адаптация системы к объектам различной конфигурации и степени сложности.
3. Защита самой информационной системы на всех уровнях в целях обеспечения безопасности предприятия.

Необходимость интеграции разнородных подсистем обусловлена тем, что в систему безопасности предприятия входят подсистемы контроля доступа, видеонаблюдения, пожарной охраны и другие системы жизнеобеспечения, т. е. совершенно разнородные системы, включающие в себя объекты различных типов, но их существование в едином информационном пространстве требует тесной интеграции.

Требование легкости адаптации системы обусловлено существенными различиями объектов, жизнедеятельность которых должна обеспечить система. Необходимость множественного внедрения такой системы вытекает из желательной унификации поведения систем жизнеобеспечения пред-

приятия и немалой стоимости разработки таких систем. Как следствие, становится необходимой возможность внесения изменений в сценарии поведения системы во время ее функционирования.

Особенно важен аспект безопасности самой системы, призванной обеспечивать безопасность предприятия. Под безопасностью понимается как обеспечение защиты системы на разных уровнях, например, защиты от несанкционированного доступа и прослушивания сети, так и защиты от физического повреждения данных, а также способность системы к восстановлению в случае всех видов повреждений.

ПРЕДЛОЖЕНИЯ ПО АРХИТЕКТУРЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ И ЖИЗНЕДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

Для правильного выбора архитектуры и способов реализации системы необходимо в полной мере сформулировать требования к системе.

Стабильность работы

Каждая часть системы должна стабильно работать, максимально независимо от остальных частей системы.

В частности, предлагаемая архитектура должна предусматривать решение следующих проблем.

- Работоспособность аппаратных средств (контроль и восстановление после сбоев).
- Подтверждение доставки сообщений при сетевом взаимодействии или гарантированной доставки таких сообщений (буферизация потоков данных).
- Работоспособность сценариев обработки, создаваемых пользователем (контроля времени обработки и зависания).

Планирование межзадачного взаимодействия

Система управления потоками работ должна обеспечивать спецификацию и планирование межзадачных зависимостей. Кроме того, параллельным выполнением могут поддерживаться восстановление и управление; в этом случае возможна интеграция планировщика, поддерживающего со-

блюдение межзадачных зависимостей, с системой управления ослабленными транзакциями.

Система управления потоком работ состоит из планировщика и агентов задач. Агент задачи контролирует выполнение задачи обрабатывающим объектом; для каждой задачи существует один агент. Планировщик — это программа, которая обрабатывает потоки работ, запуская на выполнение различные задачи, обрабатывая различные события и оценивая условия, связанные с межзадачными зависимостями. Планировщик может представить задачу на выполнение (агенту задачи) или потребовать прекращения выполнения ранее запущенных задач. В случае транзакции, затрагивающей несколько баз данных, задачи являются подтранзакциями, а обрабатывающие объекты — это локальные СУБД. В соответствии со спецификацией потока работ планировщик обеспечивает планирование зависимостей и является ответственным за обеспечение того, что все задачи достигнут приемлемого состояния завершения.

Существует три архитектурных подхода к построению систем управления потоками работ. При использовании централизованного подхода имеется один планировщик, который планирует задачи всех параллельно выполняющихся потоков работ. Частично распределенный подход предполагает наличие одного (экземпляра) планировщика для каждого потока работ. В случае, когда вопросы параллельного выполнения могут быть отделены от функции планирования, последний подход является более естественным. В случае полностью распределенного подхода планировщик отсутствует; агенты задач координируют свое выполнение, взаимодействуя друг с другом для разрешения межзадачных зависимостей и прочих требований выполнения потоков работ.

Восстановление потока работ

Целью восстановления после отказа при выполнении потоков работ является обеспечение атомарности потоков работ в связи со сбоями. Процедуры восстановления должны гарантировать, что в случае возникновения отказа в любом из компонентов, принимающих участие в обработке потока работ (включая планировщик), поток все равно достигнет приемлемого состояния — возможно, с использованием компенсации. Мы можем предположить, что обрабатывающие объекты, принимающие участие в выполнении потока работ, обладают своими собственными локальными системами восстановления и сами обрабатывают возникающие в них локальные отказы. Поэтому мы будем обсуждать только отказы менеджеров выполне-

ния потока работ (планировщиков и менеджеров параллельного выполнения).

Чтобы восстановить контекст среды выполнения, процедура восстановления после отказа должна восстановить информацию о состоянии к моменту отказа, включая информацию о состоянии выполнения каждой задачи и информацию о зависимостях планирования для менеджера параллельного выполнения. Следовательно, соответствующая статусная информация должна сохраняться на надежном запоминающем устройстве [2].

Отказ менеджера параллельного выполнения может быть обнаружен посредством механизмов временного контроля, а информация о состоянии может быть реконструирована путем просмотра журналов. В предположении возможности прямого восстановления [3] и идемпотентности обрабатывающих объектов, выполнение потока работ может быть продолжено простым перезапуском незавершенных задач.

Нам также требуется обращать внимание на содержимое очередей запросов. Если используется механизм, обеспечивающий функции стабильных программных каналов (например, в системе VMS компании DEC очереди могут быть реализованы с использованием почтовых ящиков — стабильных очередей сообщений, доступных как виртуальные устройства ввода/вывода), сообщения хранятся на постоянном запоминающем устройстве и не теряются в случае отказа. Если очереди хранятся на ненадежном запоминающем устройстве (например, в среде ОС UNIX очереди могут быть реализованы с использованием сокетов), вместо восстановления содержимого очередей можно пойти на то, чтобы планировщики повторно посылали свои запросы.

Во многих коммерческих продуктах для распределения задач по обрабатывающим станциям, ответственным за их выполнение, используются интерфейсы электронной почты. Это простое решение обладает преимуществами, вытекающими из относительной надежности существующих систем электронной почты и содержащихся в них возможностей по организации очередей и повторных посылок сообщений.

Планирование потока работ

Основными целями планировщика является обеспечение следующих требований.

- **Корректность планирования.** Процесс планирования не может нарушить ни одну из зависимостей, представленную в спецификации потока работ. Кроме того, планировщик связан ограничения-

ми, налагаемыми глобальным управлением параллельным выполнением, поскольку неконтролируемое взаимодействие задач, принадлежащих разным потокам работ, может привести к неверным результатам. Определение того, могут ли быть удовлетворены временные зависимости планирования, является особенно трудной задачей [1]. Планировщик должен учитывать, что при наличии временных зависимостей логические значения предикатов планирования могут изменяться динамически, без каких бы то ни было воздействий со стороны системы. В то же время, эти зависимости ограничивают возможные действия планировщика (например, указывая, что задача не может быть запущена ранее 10:00).

- **Безопасность.** Планировщик должен гарантировать, что поток работ будет завершен в одном из специфицированных приемлемых заключительных состояний. Перед попыткой выполнения потока работ планировщик должен обратиться к спецификации, чтобы проверить, не завершается ли этот поток в неприемлемом состоянии. Если планировщик не может гарантировать, что поток работ будет завершен в приемлемом состоянии, он должен отвергнуть такую спецификацию, не пытаясь выполнить данный поток работ. Даже если спецификация потока работ является безопасной, т. е. приемлемое состояние завершения может быть всегда достигнуто, все равно остаются проблемы с выполнением потока работ, возникающие из-за возможности возникновения разного рода тупиков. Чтобы гарантировать отсутствие тупиков в потоке работ, можно использовать формальные методы спецификации и анализа. Задачей планировщика потока работ является построение стратегии выполнения, гарантирующей, что поток работ не будет завершен в неприемлемом состоянии.

- **Оптимальная политика планирования.** Планировщику следует достигать приемлемых состояний завершения «оптимальным» образом. Тем не менее, понятие «оптимальный» может различаться для разных приложений. Одним из возможных определений может быть достижение цели за наиболее короткое время. Другим — связывание оценочной функции с выполнением каждой задачи. Тогда целью планировщика было бы выполнение всего потока работ с минимальной стоимостью. Если заранее известны вероятности успешного выполнения задач, планировщик может использовать их

для нахождения стратегии выполнения, приводящей с максимальной вероятностью к глобальному успешному завершению.

- **Обработка отказов.** Планировщик должен быть способен достигать приемлемого состояния завершения даже в случае возникновения отказа. Например, планировщик мог бы продолжить обработку после отказа и соответствующего восстановления, как будто «ничего не произошло», обеспечивая «возможность прямого восстановления» (forward recoverability). И наоборот, планировщик мог бы прекратить выполнение всего потока работ (т. е. достигнуть одного из глобальных состояний аварийного завершения). Оба подхода требуют того, чтобы информация о состоянии сохранялась в случае отказа, поскольку даже во втором случае может потребоваться фиксация или возобновление выполнения некоторых подтранзакций (например, компенсирующих подтранзакций). Следовательно, планировщик должен сохранять на надежном запоминающем устройстве всю информацию о своем состоянии, которая может потребоваться для восстановления и продолжения обработки.

Быстродействие системы

Генерация статистических отчетов должна занимать приемлемое время. В системе должны быть предусмотрены механизмы подготовки отчетов по большому количеству информации, периодически (например, раз в день) и в фоновом режиме.

Время реакции системы на событие должно быть сопоставимо со временем реакции аппаратной части.

Средства конфигурирования системы

Система должна предлагать удобные средства конфигурирования, предназначенные для инсталлятора и наладчика системы. Следует предусмотреть возможность простого создания конфигурации для большого количества однотипных устройств, однотипных сценариев бизнес-логики.

Масштабируемость

Система должна предусматривать возможность реализации как минимальной конфигурации на одном компьютере, так и распределенной системы компьютеров, объединенных в сеть.

Развиваемость

Система должна предусматривать возможность расширения как путем добавления новых устройств, так и путем добавления новых компонентов бизнес-логики и вариантов пользовательского интерфейса.

Физически система представляет собой набор внешних устройств (электромеханических устройств охранной системы), подключенных к одному или нескольким компьютерам, объединенным в сеть Ethernet.

С программной точки зрения система представляет набор компонентов, взаимодействующих посредством обмена сообщениями. Компоненты системы функционируют в среде виртуальной операционной системы (виртуальной машины), которая не зависит ни от конкретной ОС, установленной на компьютере, ни от аппаратной реализации компьютера.

Предлагаемая архитектура позволяет строить системы различного масштаба — от минимальных, реализованных на одном компьютере с подключенными непосредственно к нему внешними устройствами, до распределенной по большому числу различных компьютеров, объединенных в единую сеть. Система строится по блочному принципу из компонентов, спектр которых может расширяться в процессе развития системы.

Ядро виртуальной ОС

Ядро виртуальной ОС реализует функции обработки и маршрутизации сообщений, обеспечивая взаимодействие между компонентами системы.

Каждый компонент системы использует API ядра системы для отправки сообщений. В случае локального адресата (в рамках одного узла виртуальной ОС) сообщение попадает в очередь на обработку соответствующего компонента. В противном случае сообщение попадает в очередь на отправку сообщения через сеть. Предполагается, что механизм отправки не предусматривает блокирования основного потока выполнения, т.е. сообщение ставится в очередь, а выполнение основного процесса продолжается. Алгоритмы доставки должны гарантировать доставку сообщений адресату. Сообщение удаляется из очереди только после подтверждения его передачи следующему звену в маршруте (при сетевой передаче) или при постановке

в очередь на обработку локальному адресату. При отсутствии подтверждения система периодически повторяет попытку доставки сообщения, пока не будет достигнуто предельное время передачи сообщения.

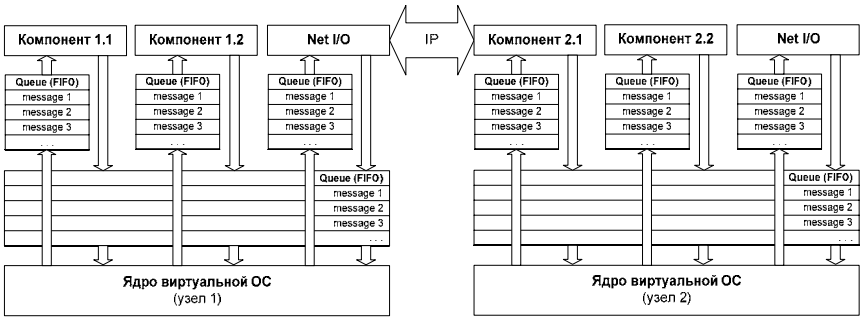

Рис. 1. Ядро виртуальной операционной системы – обработка сообщений

Механизм обработки очереди сообщений может быть различным. Базовым является механизм FIFO без распараллеливания процессов обработки: пока не окончена обработка текущего сообщения, остальные сообщения ждут в очереди. Возможно, что для некоторых компонентов необходимо будет реализовывать другие механизмы выборки из очереди (например, при обработке исключительных транзакций) и/или параллельной обработки сообщений.

Таким образом, ядро виртуальной системы является связующим элементом, который объединяет компоненты системы в виртуальную сеть, обеспечивающую обмен сообщениями.

Компоненты системы

При инициализации компонента система регистрирует имя (адрес) этого компонента, который затем используется при обмене сообщениями с другими компонентами системы. Для каждого компонента системы определен список сообщений (команд), которые он способен выполнить, а также список событий, которые могут быть сгенерированы данным компонентом. Кроме того, компонент системы может посылать сообщения другим компонентам.

Предполагается, что компоненты организованы в иерархию по принципу зависимости (например, vista → multiplexer → com-port), понятную для инсталлятора и наладчика системы. Эта иерархия должна быть отражена и на уровне GUI (Graphic User Interface).

Каждый компонент системы определяется алгоритмом обработки поступающих сообщений, параметрами конфигурации и внутренним состоянием. В общем виде взаимодействие с компонентом системы можно разделить на три категории: конфигурирование, контроль и управление. Предполагается, что помимо специфических команд, обрабатываемых определенным компонентом, любой компонент способен выполнять некоторые базовые команды (например, запрос о списке команд управления или параметров конфигурации).

Конфигурирование компонента

Для каждого типа компонентов предполагается наличие компонента, реализующего GUI для конфигурации, доступного инсталлятору системы. Такой компонент можно рассматривать как приложение, организующее пользовательский интерфейс и посылающее компоненту определенные сообщения с конфигурационными параметрами.

Список параметров конфигурации и соответствующих команд (сообщений) определен для каждого типа компонентов.

Контроль компонента

Для компонента определен список событий, которые могут быть сгенерированы им. Любой компонент системы может послать специальную команду данному компоненту и зарегистрировать себя как получателя оповещений о возникновении события. При возникновении указанного события все заинтересованные компоненты получат сообщение о возникновении события. Данная схема предназначена для организации терминалов и приложений, обрабатывающих и визуализирующих состояние компонентов. Для каждого типа компонентов предполагается наличие простейшего варианта GUI для контроля, доступного инсталлятору и наладчику системы.

Управление компонентом

Компонент может быть предназначен для управления внешним устройством или просто иметь некоторое внутреннее состояние, которым можно управлять. Для компонента определен список управляющих команд с необходимыми параметрами. Такие команды посылаются в результате обработки событий, реализованных в компонентах системы. Для каждого типа компонентов предполагается наличие простейшего варианта GUI для посылки команд управления, доступного инсталлятору и наладчику системы.

Менеджеры системы

Менеджеры системы — это абстрактный компонент системы, объединяющий компоненты по функциональным моделям. Менеджеры системы отвечают за инициализацию и конфигурирование компонентов при начальном запуске системы (включении питания), а также за сохранение и восстановление конфигурации системы.

Конфигурация менеджера содержит список и конфигурацию компонентов системы, а также список некоторых параметров, единых для компонентов менеджера. В некотором смысле, ядро узла операционной системы тоже является менеджером (а также и компонентом), но находится на самой верхушке иерархии компонентов.

Менеджер внешних устройств

Этот менеджер осуществляет загрузку (подключение) драйверов внешних устройств. Драйвер обрабатывает полученные от системы сообщения и генерируют соответствующие управляющие команды для устройства. В ответ на изменение состояния устройства (назовем такое изменение событием) драйвер генерирует сообщения и посылает их всем заинтересованным (зарегистрированным) получателям.

Драйверы устройств осуществляют функции контроля над работоспособностью устройств, а также инициализацию и восстановление после аппаратных сбоев.

Менеджер приложений

Приложения имеют API, аналогичный драйверам. Обычно приложения регистрируются у драйвера определенного устройства для обработки определенных событий, т.е. назначение приложений — осуществлять бизнес-логику обработки событий.

Одним из возможных приложений является компонент, позволяющий интерпретировать скрипты, написанные на некотором языке пользователем системы.

Другой вариант приложения — это компонент, организующий GUI для взаимодействия с другими компонентами системы.

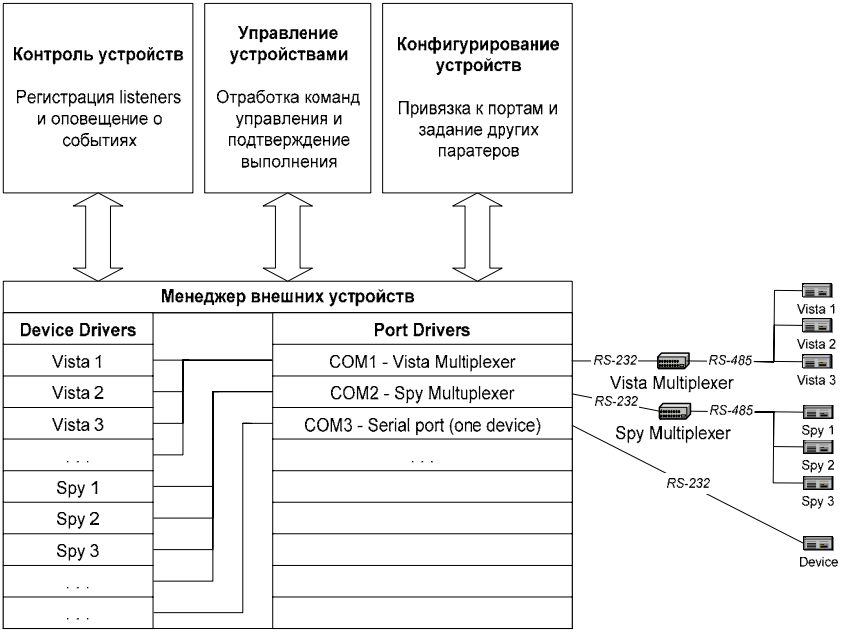

Рис. 2. Менеджер внешних устройств

Менеджер сетевого взаимодействия (Net I/O)

Этот менеджер осуществляет передачу сообщений другим узлам виртуальной ОС посредством шифрования протокола IP с обычной IP-маршрутизацией с помощью технологии SSL или IPSec, а также прием сообщений из сети для данной виртуальной машины с постановкой их в очередь на обработку. Кроме того, менеджер осуществляет защиту от несанкционированного доступа посредством механизма, подобного механизму firewall.

Менеджер данных

Этот менеджер взаимодействует с СУБД или другой системой хранения информации. Запросы к данным осуществляются через компоненты, аналогичные драйверам устройств, предоставляющие соответствующий функционал для различных типов данных. Таким образом, на уровне менеджера данных осуществляется отделение логики работы с данными от способа ее

хранения (структуры БД). На базе этого менеджера можно также создавать отдельные компоненты, предоставляющие функционал для генерации отчетов.

Менеджер синхронизации и управления транзакциями

Менеджер синхронизации и управления транзакциями позволяет создавать и использовать специальные объекты для синхронизации работы приложений и остальных компонентов. Соответствующие «драйвера-фабрики» предназначены для определенных типов объектов: семафоры (semaphores), флаги (mutex), затворы (locks) и т.д.

Менеджер предоставляет механизм создания и управления транзакциями. Компоненты системы используют данный менеджер для совместного исполнения последовательности действий в рамках одной транзакции.

РЕЗУЛЬТАТЫ ВНЕДРЕНИЯ

Первая версия описанной системы реализована в виде программного обеспечения аппаратно-программного комплекса «Восток». Опыт эксплуатации показал, что использование распределенной архитектуры позволяет значительно повысить отказоустойчивость системы, что является одним из главнейших требований, предъявляемых к системам безопасности.

Комплекс внедрен на ряде предприятий Сибири. АПК «Восток» дважды награждался Большой золотой медалью выставки «Сибирская ярмарка». В настоящее время ведутся работы над второй версией системы.

СПИСОК ЛИТЕРАТУРЫ

1. **Georgakopoulos D., Rusinkiewicz M., Litwin W.** Chronological Scheduling of Transactions with Temporal Dependencies. — Houston, 1991. — (Tech. Rep. / Dept. of Computer Science, University of Houston; UH-CS-91-03).
2. **Win W., Ness L., Rusinkiewicz M., Sheth A.** Concurrency Control and Recovery Multidatabase Work Flows in Telecommunication Applications // Proc. of the SIGMOD Conf., May 1993. — VLDB Journal. — 1994. — Vol. 3, N 1. — P. 1–28.
3. **Reuter A.** ConTracts: A Means for Extending Control Beyond Transaction Boundaries // Proc. of the 3rd Intl. Workshop on High Performance Transaction Systems, Asilomar, California, 1989.