УДК 004.4’2

УПРАВЛЯЕМАЯ МОДЕЛЯМИ РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ РАСПРЕДЕЛЕННЫХ ВСТРОЕННЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

А.В. Скворцов, к.т.н., ведущий научный сотрудник; В.И. Аржаев, к.т.н., зав. отделом
(НИИ «Центрпрограммсистем», пр. 50 лет Октября, 3а, г. Тверь, 170024, Россия, skvortsovAV@cps.tver.ru, arzhaevVI@cps.tver.ru)

Аннотация. Представлена сравнительная характеристика методологий разработки ПО для распределенных встроенных систем реального времени ROOM и COMET. Предложен путь совершенствования моделирования процесса взаимодействия при обмене потоковыми данными распределенных вычислительных платформ, представляющих собой встроенные системы на кристалле, функционирующие в режиме реального времени. Рассмотрены основные стереотипы ролей взаимодействующих объектов проектирования приложений реального времени. Исследована возможность реализации параллельных распределенных приложений реального времени в виде сетей Кана с использованием принципов метода COMET. Приведены разработанные элементы модели и их реализация на языке C++ для POSIX API.

Описан подход быстрой разработки ПО для встроенных систем реального времени на основе исполняемых моделей при комплексировании методов объектно-ориентированного и архитектурного проектирования и моделирования параллельных объектов с применением UML.

Сформулированы основные принципы разработки приложений реального времени для распределенных встроенных систем, предложен паттерн обмена потоковыми данными на основе сетей Кана.

Ключевые слова: *распределенные встроенные системы реального времени, проектирование и разработка программного обеспечения, унифицированный язык визуального моделирования, UML, ROOM, COMET, сети Кана.*

Постоянное совершенствование аппаратного обеспечения встроенных систем реального времени в настоящее время приводит к тому, что такие системы имеют вычислительную мощность и объемы памяти, сопоставимые с универсальными компьютерами. Одновременно происходит увеличение доли распределенных систем, объединяющих отдельные пространственно-разнесенные вычислительные элементы в интегрированную платформу.

Усложнение задач, решаемых встроенными распределенными системами реального времени, и повышение требований к их надежности, эффективности и скорости разработки ведут к необходимости адаптации существующих методологий проектирования ПО, управляемого моделями, к применению в данной сфере.

Методологии объектно-ориентированного проектирования приложений реального времени

К известным методологиям проектирования приложений для систем реального времени относятся

– ROOM (Realtime Object Oriented Methodology) – объектно-ориентированная методология разработки систем реального времени;

– COMET (Component and Model-based development Methodology) – метод архитектурного проектирования и моделирования параллельных объектов с применением UML (Unified Modeling Language).

Методология ROOM предложена еще до появления UML [1], но позже была адаптирована к использованию совместно с UML. Применение ROOM при проектировании программных систем реализуется посредством введения в UML нескольких стереотипов ролей объектов: капсула, соединитель, порт, протокол.

Капсулы представляют собой крупные объекты, образующие ядро проектируемой системы, которые инкапсулируют поведение, зависящее от состояния, и предоставляют интерфейс портов. Таким образом, порт – это интерфейс к капсуле, но в отличие от классических интерфейсов объектно-ориентированного проектирования порты позволяют использовать атрибуты и могут иметь сложную структуру. Они предоставляют возможность использовать синхронный или асинхронный режим обмена сообщениями между капсулами. В распределенных системах реального времени капсулы используются для представления активных параллельно выполняющихся объектов.

Порты являются реализацией классов, имеющих стереотип протокола. Протокол описывает допустимый набор сообщений, которыми капсулы, агрегирующие порт реализации протокола, могут обмениваться, а также определяет логику и последовательность обмена сообщениями через порты в виде модифицированных диаграмм состояний UML.

Соединители обеспечивают статическое и динамическое связывание портов, агрегируемых различными капсулами.

Методология ROOM является собственнической и реализована в пакете CASE-средств IBM Rational Rose Real time.

К преимуществам данной методологии следует отнести то, что использование портов, соединителей и протоколов обеспечивает эффективное определение сложных интерфейсов, которые в отличие от обычных UML-интерфейсов являются двунаправленными, поскольку описывают набор сообщений и логику обмена как для клиента, так и для сервера.

Недостатком методологии является то, что применение ROOM фактически означает принудительное использование соответствующего шаблона проектирования, достаточно тяжеловесного и избыточного во многих возникающих на практике задачах приложений встроенных систем реального времени. Кроме этого, единственная доступная реализация методологии в составе CASE-среды не поддерживает прямую и обратную разработку программного кода в части описания поведенческих аспектов протоколов взаимодействия капсул. Создаваемый программный код на языках C, C++ и Java компонуется с собственнической библиотекой, что накладывает ограничения на политику лицензирования результатов проектирования.

Метод COMET также основан на UML и использует итеративную модель жизненного цикла разрабатываемого ПО, он совместим с унифицированным процессом разработки (USDP) и спиральной моделью жизненного цикла [2, 3].

Как и в других основанных на UML методологиях разработки, на этапе моделирования требований используются прецеденты. На этапе анализа разрабатываются статическая и динамическая модели системы в виде набора классов и объектов. Особенности метода COMET, отличающие его от других объектно-ориентированных методологий разработки программного обеспечения, проявляются на этапе проектирования, где вводится понятие подсистемы. Подсистема в COMET – это один из стереотипов компонентов. Подсистемы реализуются как набор параллельно выполняющихся заданий на одной вычислительной платформе.

Таким образом, при проектировании архитектуры программной системы основным критерием выделения подсистем является их пространственная распределенность. Подсистемы взаимодействуют, обмениваясь сообщениями посредством граничных (интерфейсных) объектов особого типа. В свою очередь параллельные задания, формирующие подсистему, представляются как активные объекты, которые могут взаимодействовать друг с другом в пределах одной подсистемы либо, преодолевая границы, в разных подсистемах различными способами: асинхронно, синхронно, при помощи групповых коммуникаций.

Основным преимуществом метода COMET является его ориентированность на проектирование систем реального времени, состоящих из распределенных параллельно выполняющихся компонентов, в то время как другие основанные на UML методологии применимы в основном для последовательных систем. Кроме того, по сравнению с ROOM COMET является более гибкой, но в то же время более сложной в использовании методологией.

Модель проектирования распределенных приложений реального времени в виде сетей процессов Кана с использованием метода COMET

Сети процессов Кана являются одним из распространенных методов моделирования распределенных вычислительных задач, в котором группа детерминированных процессов взаимодействует через неограниченные FIFO-каналы.

Сети процессов обладают детерминированным поведением, которое не зависит от вычислительных и коммуникационных задержек. Изначально разработанные для моделирования распределенных систем, сети процессов оказались эффективными и для моделирования систем обработки сигналов.

Для реализации параллельных распределенных приложений реального времени в виде сетей Кана возможно использование принципов метода COMET, для чего были разработаны элементы модели и их реализация на языке C++ для POSIX API.

Основные принципы предлагаемого подхода к разработке приложений на основе исполняемых моделей:

– процессы, составляющие вычислительную сеть, представляются активными объектами COMET;

– для передачи токенов данных между процессами используется асинхронный обмен сообщениями на основе очередей, блокирующих процесс «получатель» в отсутствие токенов для обработки;

– связь между распределенными подсистемами обеспечивается посредством интерфейсного компонента, который агрегирует активный объект «координатор», конфигурируемый протокол взаимодействия подсистем, реализующий поведение, управляемое состоянием, и граничный объект интерфейса передачи токенов данных.

Статическая модель FIFO-очередей для организации параллельных потоков в сеть Кана приведена на рисунке 1 в виде UML-диаграммы классов.

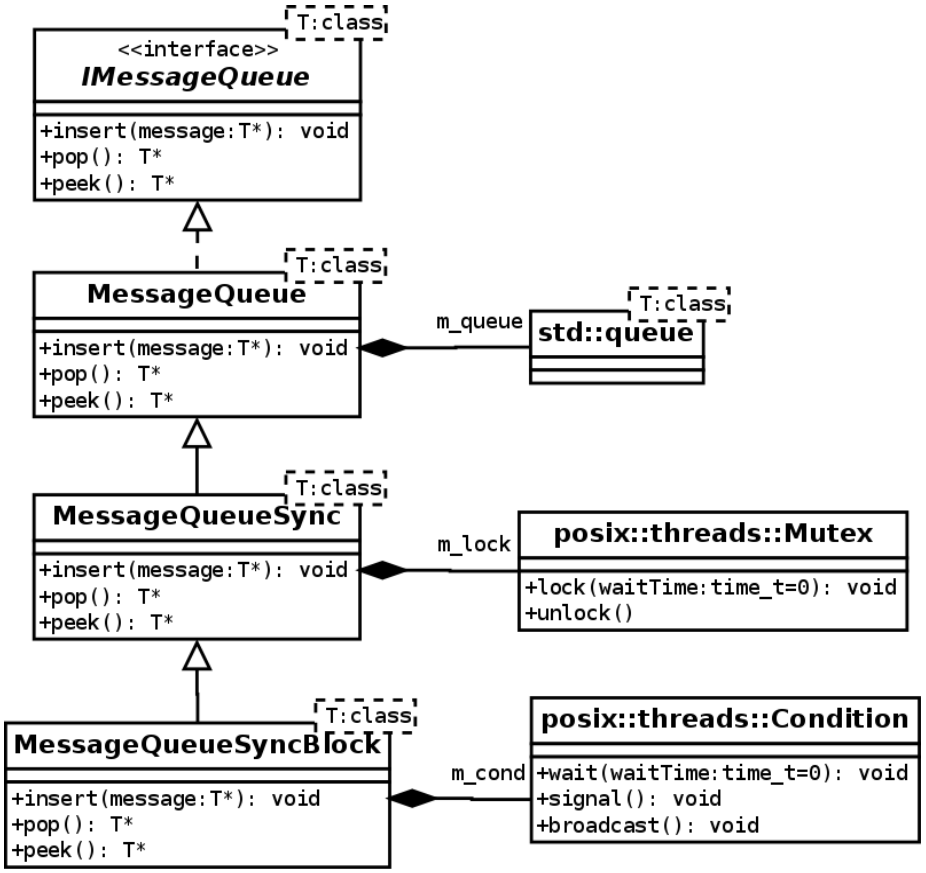

Рис. 1. Модель FIFO очередей для организации сети Кана

Основу модели составляет иерархия классов очередей различного назначения, реализующих простой интерфейс FIFO IMessageQueue. Интерфейс, как и конкретные реализации, является обобщенным относительно типа данных, хранимого в очереди. Класс MessageQueue предоставляет минимальную реализацию интерфейса, которая нуждается в сериализации методов для использования в многопоточной среде. Класс MessageQueueSync решает эту проблему, обеспечивая взаимоисключающий доступ к очереди пишущего и читающего потоков с использованием мьютекса. Класс MessageQueueSyncBlock расширяет функциональность FIFO-очереди, превращая ее в монитор Хоара с использованием условных переменных, который обеспечивает потокобезопасный доступ к хранимым токенам и блокирование читающего потока в случае отсутствия в очереди ожидающих обработки токенов.

На рисунке 2 приведена диаграмма объектов интерфейсного компонента взаимодействия распределенных подсистем.

Компонент предоставляет клиентам подсистемы (активным объектам процессов Кана) интерфейс FIFO-очереди токенов. Таким образом, разбиение спроектированной сети процессов Кана на пространственно распределенные подсистемы осуществляется простым разрывом канала передачи токенов между передающим и принимающим процессами с использованием пары интерфейсных компонентов.

Активный объект координатор находится в заблокированном состоянии в ожидании получения токенов либо из очереди на передающей стороне, либо от граничного объекта. Пассивный режим ожидания потоков, управляемый программными прерываниями сигналов поступления токенов в очередь или аппаратными прерываниями устройств ввода-вывода, позволяет использовать в конкретных реализациях невытесняющие алгоритмы планирования потоков, снижающие количество переключений контекста, за счет чего повышается пропускная способность системы.

Объект алгоритма «Протокол» определяет допустимый набор входящих и выходящих токенов, а также может реализовывать машину состояний протокола взаимодействия подсистем подобно тому, как это предусмотрено в методе ROOM.

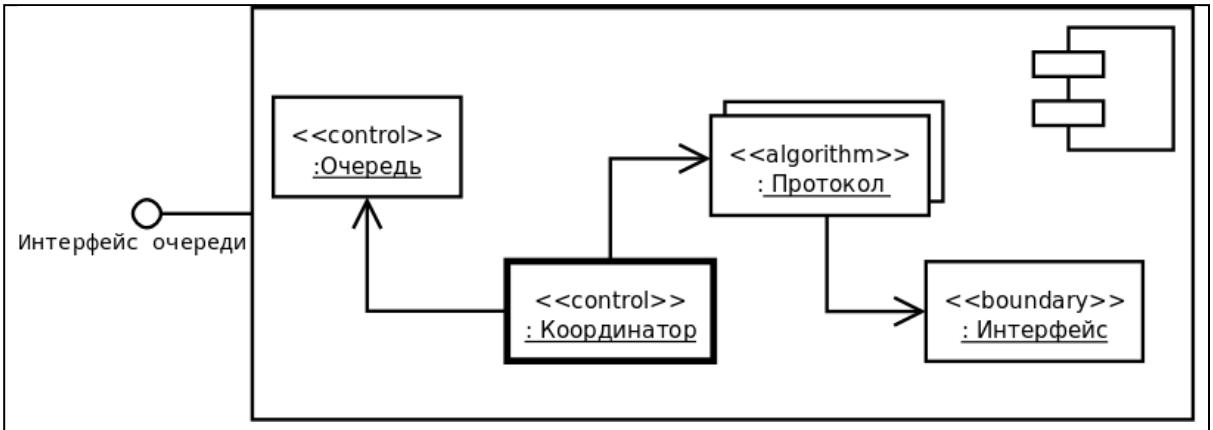

Рис. 2. Интерфейсный компонент взаимодействия распределенных подсистем

Таким образом, этап архитектурного проектирования в рамках процесса разработки встроенных систем реального времени ROPES [3] с использованием предлагаемой методики реализуется последовательностью описанных далее формальных операций.

В разработанной на этапе анализа модели вычислительной задачи или задачи обработки данных в виде сети процессов Кана выделяются подсистемы. При этом в соответствии с методологией COMET основным критерием является пространственная распределенность элементов модели.

В соответствии с шаблоном ROOM активные объекты процессов в подсистемах и интерфейсных компонентах реализуются в виде капсул, связанных асинхронными соединителями. Если протокол взаимодействия вычислительных процессов в рамках одной подсистемы или протокол взаимодействия подсистем обладает сложным поведением, он также представляется капсулой, реализующей конечный автомат протокола.

Преобразованная таким образом модель может быть реализована в среде IBM Rational Rose Real Time, что позволяет получать преимущества в скорости и качестве разработки за счет автоматической генерации части декларативного и процедурного кода, а также использования мощных средств отладки и тестирования исполняемой модели.

Следовательно, внесение в технологию ROPES элементов COMET проектирования распределенных приложений реального времени для встроенных систем с использованием основных идей сетей процессов Кана позволяет эффективно сочетать преимущества методологий ROOM и COMET.

Литература

1. Douglas B.P.. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison Wesley, 2002.

2. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений: М.: ДМК Пресс, 2011.

3. Gomaa H. Designing Real-Time applications with the COMET/UML method. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.6953&rep=rep1&type=pdf.