УДК 629.735.33

ПРИМЕНЕНИЕ UML ПРИ ПРОЕКТИРОВАНИИ ВСТРАИВАЕМЫХ СИСТЕМ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

А. Е. Леонтьев,
аспирант
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье представлена методика проектирования бортовых систем цифровой обработки сигналов с использованием нотации UML. Рассматривается пример построения многомодульной системы обработки радиолокационной информации, особое внимание уделяется описанию динамических характеристик. Предлагается метод для оценки загруженности вычислительных ресурсов и коммутационных соединений, расширяющий возможности анализа системы на стадии ее проектирования.

Данная методика является хорошим дополнением к таким методам, как математическое моделирование, и позволяет разработчику получить законченную математическо-программную модель системы ЦОС.

The article is about a method of designing digital signal processing (DSP) avionics system with using the UML notation. The primer of developing multiprocessor system of radiolocation data processing is considered, and especially is emphasized the description of system's dynamic characteristics. Also, in this article is suggested a method for calculation the workload of processing elements and communication buses, witch extend abilities of system's analysis at early stages of its developing.

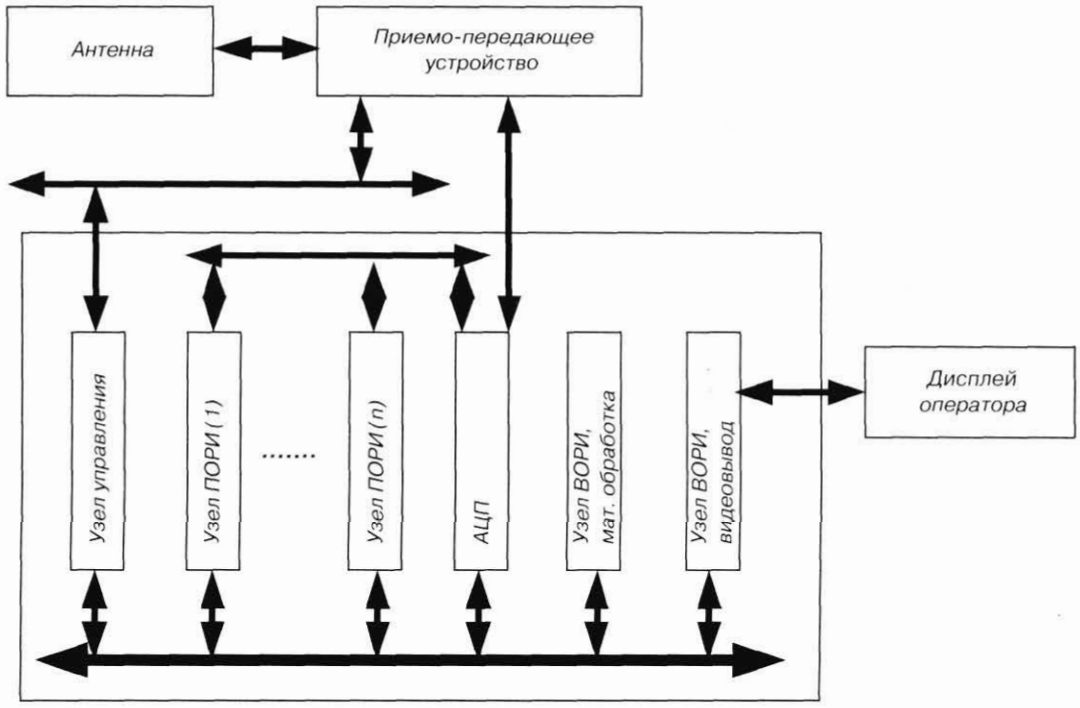
This method is a good addition for mathematic modeling methods, and it allow a developer to get completed mathematical-programming model of DSP system.

Введение

Вопрос применения унифицированных методов проектирования программного обеспечения (ПО) при разработке систем цифровой обработки сигналов (ЦОС) долгое время практически не рассматривался.

Ранние системы ЦОС разрабатывались на аналоговой элементной базе, реализовывали несложные операции фильтрации или спектрального анализа, имели достаточно большую погрешность вычислений, а их модернизация была существенно затруднена. С переходом элементной базы на цифровую основу и значительным ростом индустрии компьютерной автоматизации широкое распространение в области ЦОС получили цифровые сигнальные процессоры (ЦСП), которые позволяли перенести вычисления с аналоговой формы на цифровую. Математические алгоритмы стали реализовываться не на аналоговых схемах, а с помощью программ на языках Си и Ассемблер, исполняемых на ЦСП. При создании таких систем главное внимание уделялось аппаратной конфигурации, а методы проектирования ПО не находили применения в связи с тем, что решаемые задачи либо были слишком просты, либо имели массу значительных ограничений (по времени выполнения, использованию ресурсов памяти и интерфейсов ввода-вывода) [9]. Дальнейшее развитие индустрии привело к появлению высокопроизводительных многопроцессорных комплексов с частотами ЦСП до 1 ГГц, программных сред быстрой разработки ПО для ЦСП с использованием шаблонов вычислений, операционных систем реального времени (ОСРВ) для ЦСП и полной поддержки языка Си++. В связи с этим значительный круг задач стал реализовываться не аппаратно, а программно, появилась возможность разрабатывать сложные алгоритмические модели с поддержкой многозадачности и мультипроцессорной обработки, и острее встал вопрос об использовании методов проектирования ПО для систем ЦОС. Использование данных методов, особенно объектно-ориентированных, позволило бы также обеспечить легкую модернизацию и реконфигурацию разрабатываемого комплекса, а также облегчить процесс тестирования системы.

Однако разработчики ЦОС-систем при рассмотрении объектно-ориентированных (ОО) ме-



■ ***Рис. 1.*** *Типовая схема РЛС*

тодов высказывают следующие опасения [8]:

ОО-методы не эффективны для систем реального времени;

программы, написанные на ОО-языках программирования, работают медленнее, чем на процедурных;

необходимо вносить серьезные изменения в процесс разработки.

Все эти опасения можно опровергнуть. Во-первых, применение объектно-ориентированных методов позволяет обеспечить лучшую декомпозицию системы, выделить более требовательные к ресурсам задачи и рассматривать их более тщательно. Во-вторых, все современные среды разработки для ЦСП обладают высокооптимизирующими компиляторами языков С и С++. На высокоуровневых языках пишется в основном каркас системы (организация ввода-вывода, взаимодействие с функциями ОСРВ), в который затем вставляются оптимизированные математические функции, поставляемые вместе со средствами разработки. Таким образом, получается быстрый и легко переносимый код. Кроме того, с появлением Unified Modeling Language (UML) и развитых средств проектирования ПО, нацеленных на разработку систем реального времени, процесс освоения новой технологии не будет очень долгим и сложным. UML содержит много концепций, уже знакомых разработчикам, и позволяет описывать многие аспекты проектируемых систем.

На сегодняшний день существуют различные методологии, позволяющие расширять возможности стандартного набора UML-диаграмм. В области реального времени одной из наиболее известных является UML-RT (UML for Real Time), разработанная Б. Селиком. Однако, согласно данным проведенного анализа[3], данный метод не является оптимальным, поскольку:

в результате добавления собственных элементов придется вводить расширение на расширение, что значительно затруднит читаемость системы;

программный код, генерируемый по UML-RT (при использовании инструментального средства Rational Rose RealTime) сложно переносить на компиляторы для ЦСП;

информации об успешном применении и существующих системах, построенных на UML-RT, крайне мало.

Предметная область: радиолокационная станция для мониторинга воздушного пространства

Рассмотрим на конкретном примере использование UML-методологии для построения сложной системы реального времени с использованием ЦОС. Предметом разработки будет радиолокационная станция (РЛС) для мониторинга воздушного пространства. Типовая схема РЛС представлена на рис. 1.

В состав станции входят: антенна, приемо-передающее устройство, механизм управления антенной и блок вычислителя. Блок вычислителя, как правило, представляет собой многопроцессорный комплекс, построенный на базе магистрально-модульного стандарта (VME, CompactPCI).

На цифровой вычислитель ложатся следующие задачи:

первичная обработка радиолокационной информации – ПОРИ (выполнение алгоритмов фильтрации и спектрального анализа, характе-

■ ***Рис. 2.*** *Диаграмма прецедентов узла ПОРИ*

ризующихся интенсивными вычислениями и работающих на быстродействующих вычислителях – микросхемах жесткой логики или целочисленных ЦСП; программы на таких устройствах реализуются с использованием логики конечных автоматов);

вторичная обработка радиолокационной информации – ВОРИ (выполнение более сложных алгоритмов для оценки информации, полученной после первичной обработки, таких как формирование графической информации для вывода на дисплей оператора, обнаружение и оценка параметров движущихся объектов; программы представляют собой сложные алгоритмические конструкции с большим количеством условных переходов и выполняются на ЦСП с плавающей точкой);

программы управления и контроля (предназначены для управления режимами работы РЛС и контроля работоспособности вычислительной системы; выполняются на вычислительном узле с ЦСП или обычным процессором, имеющим доступ ко всем магистралям передачи данных системы).

При первом анализе списка задач видно, что система представляет собой комплекс программ различной сложности, работающих на ЦСП разной архитектуры и взаимодействующих между собой через различные интерфейсы. Необходимо построить спецификацию системы так, чтобы система легко модернизировалась и была наращиваемой, а спецификация содержала в себе все требования, предъявляемые к системам реального времени.

Статическая структура проектируемой системы

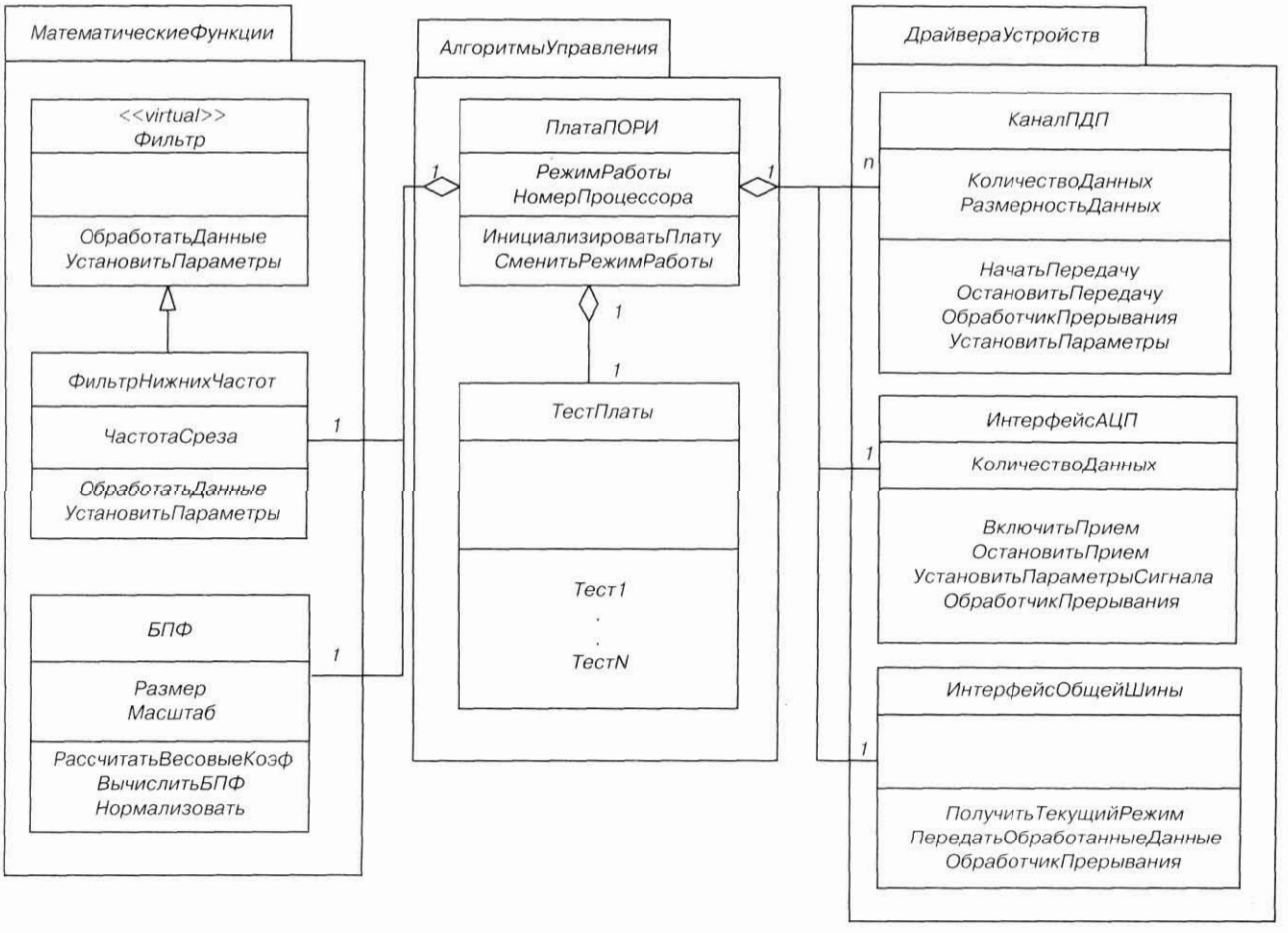
Рассмотрим спецификацию ПО для узла ПОРИ. Данный узел является в составе станции наиболее требовательным к ресурсам, и его ПО не предусматривает использование операционной системы. В задачи узла входят: получение оцифрованных сигналов с узла аналого-цифрового преобразователя, получение текущего режима обработки сигналов, выполнение математической обработки и передача обработанных данных на другие узлы для дальнейшего анализа. Задачи описываются при построении спецификации диаграммы прецедентов (рис. 2). Как правило, данный узел представляет собой многопроцессорную плату, где полученные данные симметрично разбиваются между процессами и параллельно обрабатываются.

На начальном этапе необходимо определить структурную модель системы; для этого используется диаграмма классов, соответствующая стандарту UML [2, 6]. Диаграмма классов позволяет показать статическую структуру разрабатываемой системы, ее составляющие в виде объектов или классов, связи между ними и их внутреннюю структуру. Для узла ПОРИ строятся три основных множества классов: 1) драйверы устройств; 2) математические функции; 3) алгоритмы управления. Главным классом является класс ПлатаПОРИ, включающий в себя остальные посредством агрегативных связей. Диаграмма классов представлена на рис. 3. Классы драйверов устройств и математических функций могут быть унаследованы от базовых виртуальных классов. Математические классы представляют собой своеобразные функции-«обертки» для вызова оптимизированного под конкретный процессор вычислительного кода, созданного разработчиком или поставляемого в виде библиотек вместе с системой программирования. Таким образом, при переносе программы на другую вычислительную платформу общая структура программы сохраняется, необходимо только переписать отдельные участки кода. Для удобства множества классов могут быть объединены в пакеты, которые используются в дальнейшем как шаблоны для построения модернизированных конфигураций систем.

Декомпозиция системы на классы предоставляет разработчику, в том числе, возможность удобного тестирования отдельных компонентов системы, гибкого изменения и наращивания структуры ПО. К примеру, для добавления в пакет математических функций фильтра высоких частот достаточно создать новый класс путем наследования виртуального класса Фильтр и заполнить виртуальные функции алгоритмами высокочастотной фильтрации.

Динамические характеристики

На следующем этапе необходимо определить динамическое поведение системы, которое является одним из наиболее важных аспектов при описании систем реального времени [2, 6]. Для описания динамики системы в UML используется три типа диаграмм [7]: диаграммы состояний, взаимодействия и последовательности. Диаграмма состояний, построенная на базе конечных автоматов, позволяет описать изменения состояний объекта при подаче на него управляющих воздействий. Состояния мо-



■ ***Рис. 3.*** *Диаграмма классов для узла ПОРИ*

гут быть вложенными и разделенными на несколько параллельных подсостояний, что позволяет моделировать вопросы межзадачной синхронизации и параллельной обработки.

ПО узла ПОРИ должно соответствовать требованиям жесткого реального времени и обеспечивать своевременный отклик на все внешние события. Для этого можно применить следующие технические решения:

использовать внутрикристалльную процессорную память для проведения интенсивных вычислений;

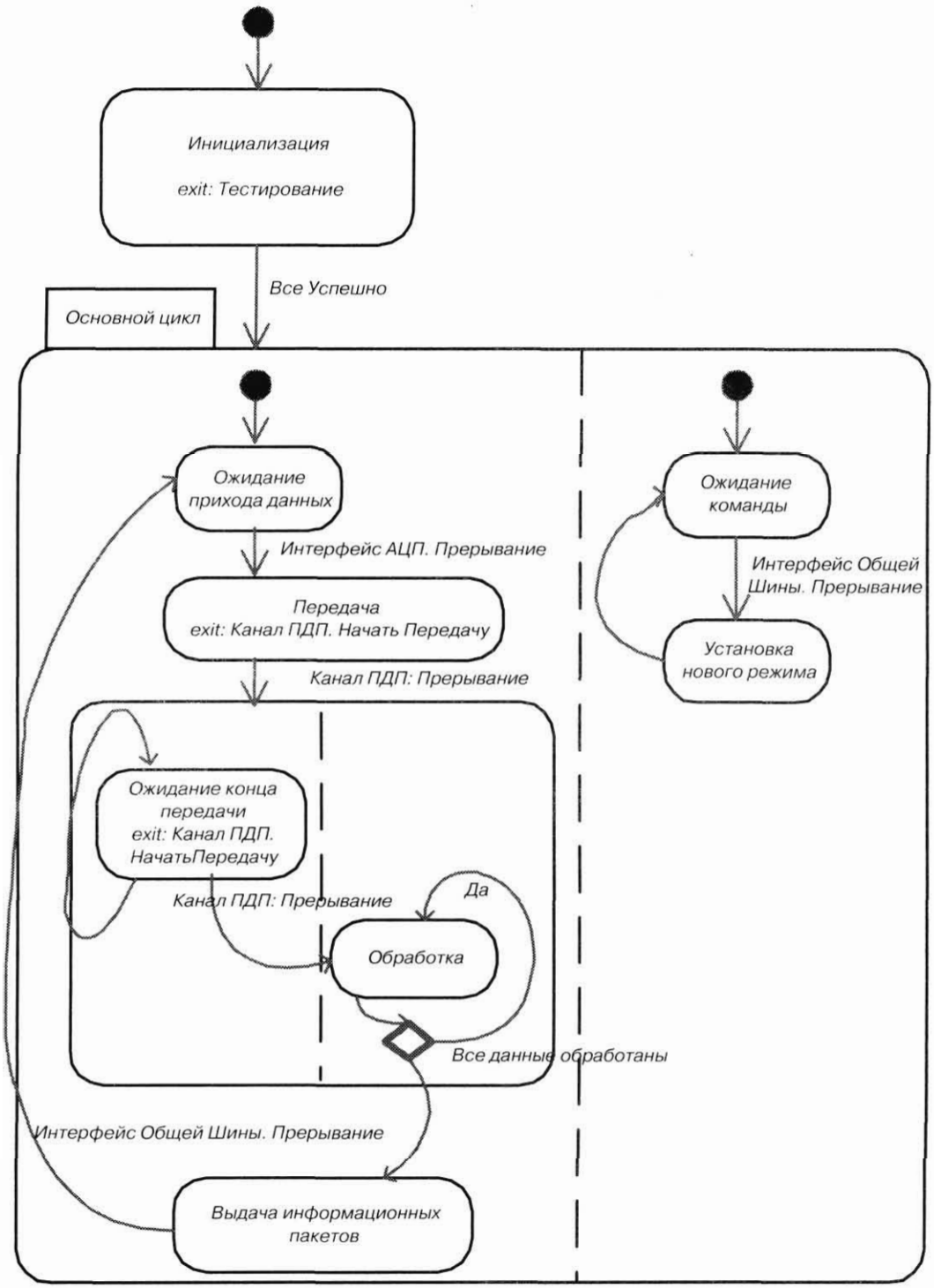
использовать каналы прямого доступа памяти (ПДП) для пересылки данных между различными интерфейсами;

отказаться от использования ОСРВ и построить структуру программы в форме конечного автомата.

Диаграмма состояний для узла ПОРИ представлена на рис. 4. После начальной инициализации системы и определения номера процессорного элемента система входит в состояние ожидания приема данных.

После поступления прерывания по окончании приема данных начинается пересылка части полученных данных во внутрикристальную память процессора, их обработка и параллельная пересылка следующей части данных. По окончанию обработки всех данных формируются информационные пакеты и рассылаются по другим узлам системы.

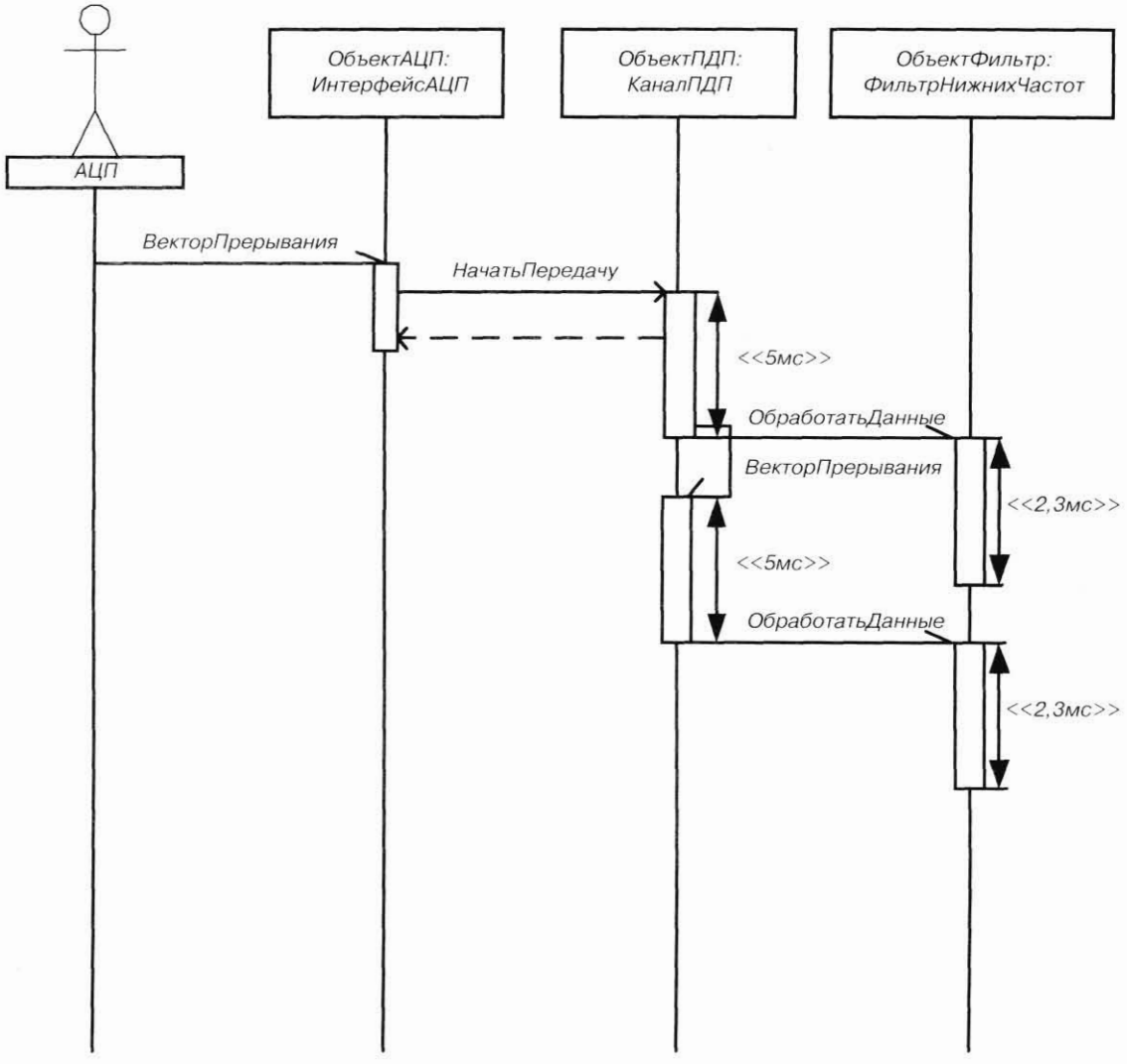
Диаграммы последовательности и взаимодействия представляют собой два различных взгляда на специфику отношений между объектами системы [7]. Диаграммы последовательности показывают временной аспект отношений, в то время как диаграммы взаимодействия акцентируются на порядке обмена сообщениями. Диаграммы последовательности позволяют наглядно показать временные рамки выполнения задач. В случае разработки ПО ЦОС этапу проектирования программной системы всегда предшествует этап математического моделирования задачи, позволяющий получить точное время выполнения каждого алгоритма. По аппаратной конфигурации системы (разрядность и скорость передачи интерфейсных шин, быстродействие каналов ПДП) рассчитывается время, затрачиваемое на прием и передачу данных между вычислительными элементами. Диаграмма последовательности для процесса обработки данных узла ПОРИ представлена на рис. 5.



■ ***Рис. 4.*** *Упрощенная диаграмма состояний для узла ПОРИ*

Для получения временны́х оценок на основе диаграмм последовательностей в литературе предлагаются такие методологии, как теория планирования в реальном времени, моделирование систем с помощью сетей Петри [5].

Завершающим этапом проектирования программной системы является краткое описание физической конфигурации системы и распределение программных компонентов по вычислительным узлам. Описание производится с помощью диаграмм размещения. Узел ПОРИ представляет собой четырехпроцессорный модуль с интерфейсами общей шины и АЦП, диаграмма размещения для него представлена на рис. 6. Оцифрованные данные с АЦП разбиваются на четыре части и параллельно обрабатываются на процессорах. Каждый процессор при начальной



■ ***Рис. 5.*** *Диаграмма последовательности для процесса обработки данных узла ПОРИ*

инициализации в зависимости от своего номера на плате выбирает свою часть информации для обработки.

Методика расширения UML-диаграмм для получения оценок ресурсоемкости

Следует заметить, что диаграммы размещения (см. рис. 6) отображают лишь примитивную структуру сложной системы и не позволяют доподлинно показать и специфицировать особенности функционирования вычислительных устройств, интерфейсов и коммутационных шин. Одним из основных вопросов, возникающих при проектировании систем ЦОС, помимо временных оценок является проблема определения загруженности вычислительных устройств и сред передачи данных. Эту проблему можно решить следующим предложенным методом, который предусматривает:

расширение и дополнение стандартных UML-диаграмм до учета временных и ресурсных характеристик системы (выделение на диаграмме размещения дополнительных функциональных элементов, введение дополнительной диаграммы времени выполнения задач);

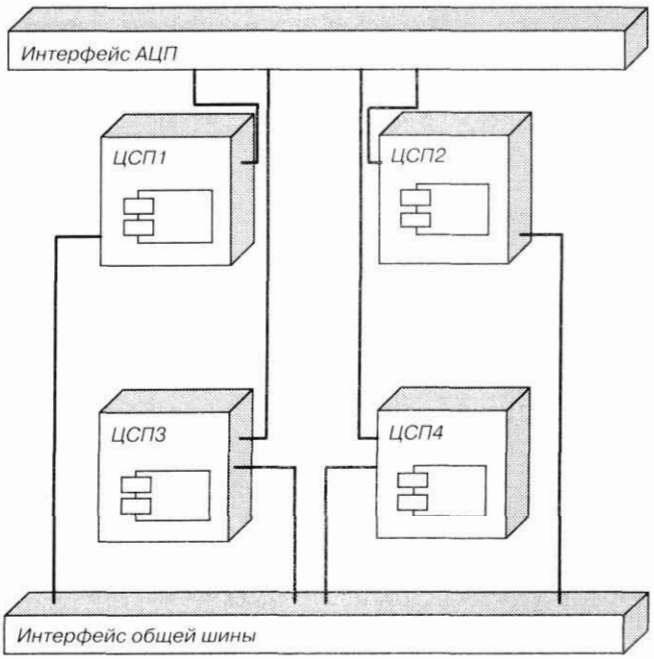
построение аналитических моделей аппаратных ресурсов системы;

трансляцию полученных UML-диаграмм в концептуальную модель [1];

совместный параметрический анализ концептуальной и аналитической модели и получение диаграмм степени загрузки системы.

В результате использования метода уже на ранних стадиях разработки систем можно получить карты загрузки вычислительных ресурсов и обнаружить «узкие места» системы. Это позволит избежать дополнительных рисков и сократить время разработки сложных систем с большим количеством вычислительных ресурсов и коммутационных соединений.

Полученные UML-диаграммы, разработанные с помощью программного средства проектирования ПО, можно использовать для генерации исходного кода на языках программирования C и C++ и документов, содержащих программные



■ ***Рис. 6.*** *Диаграмма распределения узла ПОРИ*

спецификации [4]. Использование специализированных средств проектирования ПО реального времени позволяет моделировать динамику проектируемой системы и получать полностью функциональный программный код, готовый к компиляции и ориентированный на конкретную программную и аппаратную платформу [6].

В рассматриваемом примере для генерации кода на языках высокого уровня используется инструментальное средство Rhapsody от компании i-Logix. Это средство генерирует программный код не только по диаграммам классов, но и по диаграммам состояний, позволяя получать полностью работоспособное приложение. При желании, например при вызове оптимизированной функции или системной функции операционной системы, можно добавлять операторы языков высокого уровня прямо в спецификации элементов UML-диаграмм, обеспечивая тем самым необходимую функциональность.

Заключение

Методология UML позволила бы описывать практически все аспекты, необходимые для построения современных систем ЦОС. С применением подобной технологии проектируемая система будет легко модернизироваться и наращиваться, а также отвечать заданным требованиям по производительности и надежности. Использование нотации UML в программах ЦОС является хорошим дополнением к существующим методам, таким как математическое моделирование, и позволяет разработчику получить законченные математические и программные модели, готовые к реализации.

Литература

1. **Бржезовский А. В., Жаков В. И., Путилов В. А., Фильчаков В. В.** Синтез моделей вычислительного эксперимента. – СПб.: Наука, 1992. – 231с.
2. **Гома Х.** UML. Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. – М.: ДМК Пресс, 2002. – 704 с.
3. **Arbelet L., Davy P.** L'embarqué tire le langage UML vers le temps réel // L' informatique no. 3 (Mars 2003). – P. 18–19.
4. **Boggs W., Boggs M.** Mastering UML with rational rose 2002. AL.: Sybex, 2002. – 714 p.
5. **Cheng M. A.** Real-time systems. Scheduling, Analysis and Verification. – New Jersey: Jonn Wiley and Sons, 2002. – 524 p.
6. **Douglass B. P.** Real-time design patterns: Robust scalable architecture for real-time systems. – Reading, Mass.: Addison-Westley, 2002. – 528 p.
7. **Jacobson I., Rumbaugh J., Booch G.** The unifed modeling language reference manual. Reading, Mass.: Addison-Westley, 1999. – 568 p.
8. **McNutt S.** DSP systems design goes object-oriented // Embedded systems programming no. 10 (October 2002). – P. 25–32.
9. **Wieringa R. J.** Design methods for reactive systems: Yourdon, statemate, and the UML. Morgan Kaufmann publishers, 2003. – 457 p.