А.С. Кнутов

Анализ формальных моделей для специфицирования распределенных автоматизированных систем обработки данных на этапе их проектирования

ФГУП «НПЦАП имени академика Н.А. Пилюгина»
Тел.: +7 (495) 535–33–96

Рассмотрены особенности автоматизированных систем обработки данных ответственного применения. Проводится сравнение применимости сетей Петри, модели потоков данных и формализмов языков группы UML в качестве формальных моделей, пригодных для использования на стадии проектирования.

Рассмотрен вопрос структурирования АСОД ОП с учетом длительного срока их эксплуатации, в условиях, когда этот срок значительно превышает гарантийный срок службы технических средств и срок действия сертификатов на используемое программное обеспечение.

Ключевые слова. *Система обработки данных, проектирование, моделирование, диаграммы, процессы, целевой функционал, обеспечивающая платформа.*

A.S. Knutov

Analysis of the formal models for specifying the distributed automated data processing system at the design stage

Federal State Unitary Enterprise 'Academician Pilyugin Scientific-Production Center of Automatic and Instrument Making'
Tel.: +7 (495) 535–33–96

The article describes the features of the automated data processing systems (ADPS) for special applications, or responsible use (RU). Author compares the applicability of Petri nets, data-flow models and formalisms of UML group of programming languages as formal models suitable for use at the design stage.

The paper considers structuring the ADPS for RU in linkage with the long term of their operation, in conditions where this period far exceeds the warranty period of service of technical part of the systems and the period of validity of certificates on the software used.

Keywords. *The data processing system, design, modeling, diagrams, processes, target functionality, the providing platform.*

Введение

В данной статье рассматриваются некоторые проектные вопросы при создании автоматизированных систем обработки данных (АСОД) ответственного применения (ОП). Под термином АСОД ОП здесь понимаются такие программно-аппаратные автоматизированные системы, отклонение в работе которых от заданных начальных условий информационного обеспечения и условий реализации вычислительного процесса может привести к катастрофическим последствиям.

Исходя из этого, рассматриваемые АСОД ОП – это системы с жестко детерминированным поведенческим обликом, характеризующиеся явной доминантой роли программного обеспечения над аппаратными средствами [1].

Разработчик таких систем обязан обеспечить выполнение ряда жестких и часто противоречивых требований. А именно:

- требования по надежности функционирования системы в целом и ее составляющих – программ и аппаратуры;
- требования по безопасности функционирования системы в целом;
- требования по защите данных от несанкционированного доступа;
- требования по операторской эргономике и удобству использования системы;
- требования по обязательному применению некоторых трудно интегрируемых между собой аппаратных и программных компонент.

Зачастую требования, определяемые в техническом задании на систему, либо излишне детализированы, вынуждая разработчика прибегать к не самым оптимальным решениям, либо, наоборот, имеют слишком общий, нечеткий и расплывчатый характер, что создает для разработчика условия блуждания и следования путем, отличным от оптимального.

1. Жизненный цикл программного обеспечения и особая роль в нем проектной стадии

В настоящее время разработаны многочисленные представления о том, какие этапы проходит программное обеспечение на пути своего создания от начальной постановки задачи и до получения реальной пользы от эксплуатации готовой системы.

Основная идея концепции жизненного цикла программного обеспечения в контексте его разработки состоит в том, чтобы разбить полный цикл разработки на такие стадии, которые позволили бы контролировать результат данной стадии по качеству исполнения запланированного объема работ. При этом полный жизненный цикл разработки призван обеспечить на выходе получение продукта с заданным набором свойств и с заданным уровнем качества.

Универсального цикла жизни для разработки программного обеспечения на сегодня не создано. Крупнейшие софтверные компании и компании с развитым софтверным кластером, включая IBM, Microsoft, Boeing, Nokia, Samsung и аналогичные им, придерживаются собственных уникальных методологий разработки, в основе которых лежит их собственное представление о необходимых стадиях разработки и о необходимых методах контроля результатов [3].

Однако все известные методологии разработки предусматривают в том или ином виде наличие следующих стадий:

- исследование предмета автоматизации и специфицирование задачи, в результате чего производится формулирование наиболее общих требований по структуре будущей системы и по ее функционалу;
- разработка функциональной структуры системы как совокупности упорядоченно взаимодействующих между собой частей общего функционала;
- разработка структуры системы как совокупности взаимодействующих между собой программных модулей, реализующих заданный функционал;
- разработка частных программных средств будущей системы;
- комплексирование частных программных средств и их совместная отработка, формирование полной системы, функционирующей в реальных условиях;
- проведение квалификационных испытаний, подтверждающих определенные в техническом задании параметры разработки;
- приведение программного обеспечения к виду и форме, позволяющим его хранение, распространение и установку на вычислительных средствах;
- ввод системы в эксплуатацию, наработка необходимых и целесообразных коррекций в постановке задачи, в функционале системы и в ее системной структуре;
- проведение итерации по учету наработанных коррекций.

Этап проектирования программной системы включает в себя разработку функциональной структуры и системной структуры.

Особая роль проектного этапа в рамках жизненного цикла характеризуется тем, что на нем производится формирование полного и детального представления о будущей системе:

- об основном предназначении системы и рамках ее применимости;
- о предполагаемых функциональных возможностях системы в различных условиях применения;
- о составе и особенностях интерфейсов взаимодействия системы с человеком и с внешним окружением;
- об особенностях поведения системы в тех или иных ситуациях;
- о прочих свойствах и особенностях системы.

Проектная стадия жизненного цикла на практике призвана обеспечить:

- безусловный учет функциональных и эксплуатационных требований в проекте;
- отработку методических, сценарных и интерфейсных решений методом имитационного моделирования [2];
- своевременное уточнение существа разработки путем всестороннего рассмотрения проекта широким кругом экспертов;
- снижение общей трудоемкости разработки за счет снижения последующей вариантности разработки и отсечения излишних ветвей на дереве функциональных возможностей.
- Важным инструментом проведения этих мероприятий является совокупность статических и динамических моделей будущей системы [3].

2. Проектные модели представления процессов обработки данных

Ниже рассмотрены три типа моделей представления систем применительно к АСОД ОП:

- модели дискретных динамических систем [4];
- модели потоков данных [5, 6];
- модели в нотации языков UML-группы [7, 8].

2.1. Модели в виде дискретной динамической системы

Модели дискретных динамических систем широко используются в кибернетике и информатике, поскольку они позволяют описывать и исследовать технические устройства (вычислительные машины, сложные алгоритмы, операционные системы, вычислительные сети и т. п.) на высоком уровне абстракции.

Среди многих существующих методов описания и анализа дискретных параллельных систем (конечные автоматы, сети конечных автоматов) выделился подход на основе сетей специального вида, предложенных Карлом Петри для моделирования асинхронных информационных потоков в системах преобразования данных.

Сети Петри позволяют представить полную систему в виде ее компонент и действий, производимых компонентами. Компоненты и их действия представляются абстрактными **событиями**.

Совокупность действий, возникающих как реализация событий при работе системы, образует **процесс**, порождаемый этой системой.

Соотнесение функционирования системы с аспектами реального времени производится через события (начало этапа обработки, окончание этапа обработки, наступление заданного момента времени и т. д.).

В сетях Петри дискретные системы представляются двумя непересекающимися множествами – множеством **событий** и множеством **условий**.

Существует ряд инструментальных реализаций таких моделей в виде так называемых Петри-машин, которые позволяют строить по заданным формулам иерархическую сеть с ожиданием и могут имитировать ее функционирование.

Однако подобные динамические модели не могут обеспечить всестороннее отражение структурных особенностей построения системы, способов взаимодействия системы с внешним миром и других аспектов, выходящих за рамки чистой логики. За рамками модели остаются также свойства и характеристики обрабатываемой информации. В силу этого эти модели относительно рассматриваемых АСОД ОП следует отнести к моделям низкого уровня репрезентативности.

2.2. Модели в виде потоков данных

Модель в виде потоков данных наилучшим образом сформулирована в методологии Гейна-Сарсона, ставшей стандартом «де-факто» в данном вопросе. В ней информационная система определяется как иерархия **диаграмм потоков данных** (ДПД, или DFD в англоязычной литературе). С помощью ДПД описываются асинхронные процессы преобразования информации на различных стадиях ее обработки, от начального ввода в систему и до конечной стадии выдачи результата потребителю. Этапы обработки информации выделяются структурно в виде систем и подсистем.

В иерархии диаграмм потоков данных особое место занимают диаграммы верхнего уровня – контекстные диаграммы, поскольку они определяют взаимосвязи основных подсистем с внешними входами и выходами. Внешние по отношению к рассматриваемой системе объекты обозначаются в методологии ДПД как **внешние сущности**.

Декомпозиция диаграмм сверху вниз позволяет создать многоуровневую иерархию до такого уровня, пока не будет достигнут уровень декомпозиции, на котором детализировать процессы далее не целесообразно или не требуется.

В соответствии с методологией использования ДПД, источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации.

С практической точки зрения модели на основе ДПД позволяют определить и зафиксировать пути движения информации и зафиксировать узлы ее обработки – будущие задачи АСОД ОП.

Состав понятий, которыми оперирует разработчик при использовании ДПД, включает в себя:

- внешние сущности (внешние источники и потребители информации);
- процессы (системы/подсистемы);
- накопители данных;
- потоки данных.

Модели подобного вида позволяют адекватно представить АСОД ОП с точки зрения состава и структуры информации, этапов ее обработки, определить места накопления и хранения информации. Одновременно с этим разработчик получает структурное решение будущей системы, правда без алгоритмических деталей обработки информации и без деталей распараллеливания и синхронизации задач.

2.3. Модели в нотации UML

Обобщение опыта по созданию математических методов, изобразительных средств и инструментов моделирования реальных систем привело к появлению унифицированного языка моделирования UML (Unified Modeling Language), интенсивное развитие которого быстро привело к появлению расширенного языка UML-2 и языка SYSML, ориентированного на описание работы аппаратной части систем.

Базовый язык UML включает в себя девять основных видов диаграмм:

1) диаграммы прецедентов;
2) диаграммы классов;
3) диаграммы объектов;
4) диаграммы кооперации;
5) диаграммы последовательности;
6) диаграммы состояний;

7) диаграммы деятельности;
8) диаграммы компонентов;
9) диаграммы развертывания.

Расширения базового UML имеют в своем арсенале дополнительные специализированные виды диаграмм. В различных реализациях UML-2 насчитывается более 15 типов диаграмм.

Важным обстоятельством появления языков UML-группы является их изначальная ориентированность на описание распределенных приложений, т. е. приложений, состоящих из нескольких географически разнесенных узлов. Каждый узел распределенной системы представляет собой самостоятельную вычислительную систему, информационно связанную с другими вычислительными системами. Соответственно, в диаграммах предусмотрены специальные элементы, ответственные за организацию параллелизма и синхронизации потоков и процессов.

Методы проектирования на основе языков UML-группы в настоящее время имеют реализацию в инструментах, позволяющих производить разработку как статических, так и динамических моделей.

По широте изобразительных возможностей язык UML и его расширения являются существенно более мощным средством, нежели модели на основе динамических сетей и моделей в виде потоков данных.

Стандартизация изобразительных средств и языковой семантики UML в сочетании с объектно-ориентированным подходом способствовали развитию средств моделирования и их интеграции с традиционными средствами разработки программ на языке программирования. Сегодня многочисленные инструментальные системы разработки Borland, Microsoft и других производителей включают в себя компоненту моделирования на UML как неотъемлемую часть.

Современные инструменты разработки программного обеспечения на основе UML дают разработчику ранее недоступные ему возможности:

- описать взаимодействие проектируемой системы с внешним миром с помощью специализированных диаграмм прецедентов;
- описать функционирование системы в иерархии диаграмм процессов, объектов, классов и прочих диаграмм с различных точек зрения;
- сформировать и исследовать статическую модель и тут же получить (продуцировать) ее реализацию на языке программирования;
- исследовать продуцированную модель системы в динамике и выработать необходимые коррекции в структуре и функционале проектируемой системы.

В начале 2000-х годов разработчикам стали доступны UML-инструменты, позволяющие создавать и исследовать сразу динамические модели систем, без явной стадии перевода диаграмм на язык программирования [8, 9].

3. Преимущества использования проектных моделей при разработке АСОД ОП

Модели системы, в особенности динамические модели, дают разработчику на стадии проектирования более реалистичное представление о трудоемкости предстоящей разработки по мере детализации модели системы.

При наличии хорошего инструмента у разработчика появляется возможность моделировать будущую систему на произвольном уровне ее декомпозиции и по структуре внутреннего устройства, и по динамике поведения.

При этом на стадии реализации системы, появляется возможность контролировать реальное положение дел в разработке на основе сопоставления разработанного объема реальной системы с имеющейся моделью.

Важным аспектом с практической точки зрения создания АСОД ОП является возможность сохранять проектный задел на уровне моделей для последующих разработок, особенно если полная мо-

дель АСОД ОП представлена в виде четкой модульной структуры. В этом случае разработчик получает своеобразный депозитарий типовых модельных блоков, выраженных с помощью UML-диаграмм.

Практическая польза блочных моделей АСОД ОП иллюстрирована ниже на примере этапного развития некой гипотетической АСОД ОП.

4. Разделение проектных моделей на уровни «функционал» и «платформа»

Длительные сроки эксплуатации АСОД ОП, быстро сменяемые технические средства и ограниченные сроки их гарантийной работы ставят разработчика перед необходимостью проектировать АСОД ОП не как статично исполненный технический объект, а как объект, развиваемый во времени [1, 2].

При этом понятно, что основная часть будущей системы, связанная с ее целевым назначением, не будет претерпевать существенных изменений в течение всего срока предполагаемой эксплуатации, который может составлять 15 лет и более. В свою очередь, обеспечивающая часть системы, которую принято обозначать термином «платформа», может устаревать через каждые пять-семь лет.

В качестве примера. Фирма Intel выпускает свои вновь разработанные процессоры в течение двух-трех лет. Процессоры специального применения выпускаются не более чем в течение пяти лет [9]. Аналогично поступают и другие производители. По истечении указанного срока разработчик остается один на один со своими проблемами, поскольку заказать точно такую же аппаратную часть у него нет возможности.

Как следствие устаревания аппаратной части наступает ситуация, когда следом устаревает ОС. Так, например, применяемая в системах ответственного применения ОС МСВС 3.0 на сегодня не поддерживается значительным количеством современного компьютерного «железа». Аналогичная проблема возникает в случае истечения срока действия сертификатов на используемое программное обеспечение, если продлить их действие не представляется возможным в силу объективных причин.

Таким образом, разработчик для обеспечения 15-тилетнего срока эксплуатации системы должен будет ее разработать трижды.

Чтобы смягчить означенную проблему, целесообразно было бы заранее заложить в систему свойство ее изменяемости в отдельных частях, т. е. возможность переноса главной части системы, ее целевой функции, с одной реализующей платформы на другую. Это легче будет осуществить в последующем, если изначально на проектном уровне систему структурно разбить на целевой функционал и обеспечивающую платформу.

В качестве иллюстрации такого подхода можно привести пример развития некоторой гипотетической АСОД ОП во времени.

Этап-1. Проектируется и реализуется некоторая АСОД ОП как совокупность отдельных рабочих станций, каждая из которых производит некоторый этап обработки данных. Перенос данных между ними производится на промежуточных носителях информации (ПНИ).

Целевое программное обеспечение в каждой рабочей станции структурируется в три блока: блок приема данных, блок обработки данных и блок выдачи данных. В качестве блока приема данных выступает простейший модуль чтения данных с ПНИ, а в качестве блока выдачи данных выступает простейший модуль записи данных на ПНИ. Блок обработки данных увязан с блоками приема и выдачи данных специально спроектированным протоколом взаимодействия.

Этап-2. По прошествии времени возникла необходимость переноса системы на сетевую платформу. При этом предполагается полностью сохранить функционал. В этом случае будет логично саму платформу разработать как самостоятельную систему. В рамках новой платформы разрабатываются специфичные задачи обмена и управления сетью как самостоятельным техническим объектом. Блоки обработки данных при этом не претерпевают существенных изменений в силу того, что сама по себе целевая задача системы не изменилась.

Взаимодействие блоков обработки данных с блоками приема и выдачи данных в этом случае реализуется в соответствии с уже заложенным в них протоколом взаимодействия, хотя модули приема и выдачи данных реализуются уже в виде новых модулей обмена по сети.

Этап-3. Сетевая стационарная платформа перестает удовлетворять эксплуатирующего субъекта и встает вопрос о переносе целевого функционала на мобильную платформу, использующую в своей основе беспроводные технологии передачи данных. Состав необходимых изменений в системе легко предугадать:

- нужно создать распределенные рабочие места, взаимодействующие между собой по беспроводным технологиям;
- нужно разместить на новых рабочих местах неизменившиеся модули обработки данных;
- нужно модифицировать модули приема и выдачи данных, увязав новую платформу с неизменившимися целевыми модулями.

Успех или неуспех подобных структурных решений во многом зависит от полноты представления о системе на ранних этапах ее создания. А полнота представления целиком и полностью стоит в зависимости от наличия и качества моделей.

Заключение

Особенности рассматриваемых систем АСОД ОП ставят перед разработчиком задачу выработки специальных приемов и методов проектирования таких систем.

Эти приемы и методы проектирования должны обеспечивать:

- исследование будущей системы с помощью статических и динамических моделей на как можно более ранних стадиях разработки;
- максимальную формализацию описания будущих свойств системы и требований к ней;
- разработку самых первых официальных документов (Техническое задание и Эскизный проект) с включением в них по возможности формализованных требований и формализованного описания предполагаемой реализации;
- продуцирование программных блоков системы непосредственно из ее формальной модели;
- формирование и детализирование структуры системы с прицелом на дальнейшее развитие ее функционала и возможной смены реализующей платформы;
- разбиение этапов разработки на стадии, результаты выполнения и качество выполнения которых можно было бы контролировать по объективным показателям.

Литература

1. **Микрин Е.А., Кнутов А.С.** Перспективы применения диагностической экспертной системы для наземного комплекса отработки бортового программного обеспечения КА // Материалы Международной научно-технической конференции ИМС 2005. Том 1.
2. **Микрин Е.А., Кнутов А.С.** Использование диагностической системы для наземного комплекса отработки бортового программного обеспечения Международной космической станции // Проблемы управления. 2005. № 1.
3. **Бобровский С.** Программная инженерия. Технологии пентагона на службе российских программистов. СПб.: Питер, 2003.
4. **Котов В.Е.** Сети Петри. М.: Наука. Главная редакция физико-математической литературы, 1984.
5. Диаграммы потоков данных // CIT-Forum, http://citforum.ru/programming/oop_rsis/glava2_6_1.shtml
6. Моделирование потоков данных // INTERFACE.RU, http://www.interface.ru/home.asp?artId=22366
7. **Хасан Гома.** UML Проектирование систем реального времени, распределенных и параллельных приложений. М.: ДМК Пресс, 2002.
8. **Лоненков А.** UML 2. СПб.: БХВ-Петербург, 2007.
9. Материалы семинаров «Встраиваемые компьютерные технологии» за 2010, 2011 и 2012 годы. Компания «РТ-софт», Москва.