Проектирование типовых операционных последовательностей нижнего уровня информационной системы учета продукции

М. В. Паринов, email: parmax@mail.ru [1]
К. В. Костин, email: phantasyftw@bk.ru [1]
Н. А. Чернышов, email: chernyshovkolya1994@mail.ru [1]
П. В. Шуваев, email: asutp3@molvest.ru[2]

[1] Воронежский государственный технический университет
[2] Молочный комбинат «Воронежский»

Аннотация. *В данной работе рассматриваются особенности проектирования нижнего уровня аппаратно-программного комплекса учета готовой продукции. Рассматривается процесс разработки алгоритмических последовательностей, описаны задействованные объекты и их взаимодействие. Результаты проектирования представлены в виде обобщенной диаграммы последовательностей UML, отображающей основные процессы системы.*

Ключевые слова: *Информационная система, микроконтроллер, проектирование, алгоритмические последовательности, диаграмма последовательностей, программный модуль.*

Введение

Для реализации любой информационной системы необходимо выполнить ее тщательное проектирование. В некоторых случаях (обычно для систем особо малого уровня сложности) этап проектирования интегрируют в разработку. Обычна данная методика приводит к негативным последствиям вследствие неоправданного значительного усложнения структуры системы и потери четкости связей между элементами [1,2].

Известны классические методики проектирования информационных систем. Они хорошо описывают проектные работы для достаточно сложных разработок. При этом процесс проектирования относительно простых информационных систем в реальности отличается. Отличия выражены невозможностью команды разработчиков тратить значительные временные ресурсы на классический проект, включающий типовой набор документации. Обычно она базируется на схемах, построенных с помощью языка UML.

Проектируемая нами подсистема в настоящее время может быть отнесена к невысокому уровню сложности с точки зрения объема разработки. Однако, согласно техническому заданию, система является гибко расширяемой и уже сейчас содержит элементы достаточно крупного продукта. Таким образом, на текущем этапе возникает задача проектирования достаточно компактной информационной системы с возможностью ее беспрепятственного расширения.

Для решения текущей задачи нами предложено использовать лишь отдельные схемы UML [3,4], которые описывают основные вопросы текущего состояния разработки и могут быть в дальнейшем использованы для ее расширения. Обычно проектирование информационных систем начинается с создания диаграммы прецедентов. Однако с учетом объема работ данной работой можно пренебречь, так как обобщенная картина взаимодействия системы и ее подсистем с внешним миром хорошо прослеживается из технического задания. Поэтому в качестве основных проектных документов мы представим диаграмму последовательностей и диаграмму классов.

Диаграмма последовательностей является более общим документом, тогда как диаграмма классов уже строго представляет структуру системы (подсистемы). Для построения диаграммы классов мы должны строго представлять объекты и связи проектируемой системы. Следовательно, диаграмма последовательностей будет начальном этапом проектирования системы.

Для создания диаграммы последовательностей необходимо определиться с основными функциональными объектами системы. Их можно получить из виртуальной (представляемой) диаграммы прецедентов.

1. Основные объекты подсистемы и диаграмма последовательности

На текущем этапе нами выделено 6 основных объектов подсистемы. Их количество может быть увеличено как путем расширения функциональности, так и методом уточнения, когда существующие объекты будут разделены на несколько подобъектов.

Рассмотрим основные объекты, которые показаны на диаграмме, представленной на рис.1.

TCP client – сетевой объект. Выполняет коммуникацию с верхним уровнем системы посредством локальной вычислительной сети. Используется транспортный протокол TCP для реализации аппаратного управления потоком и минимального контроля ошибок передачи. Объект отвечает за прием/передачу команд и данных.

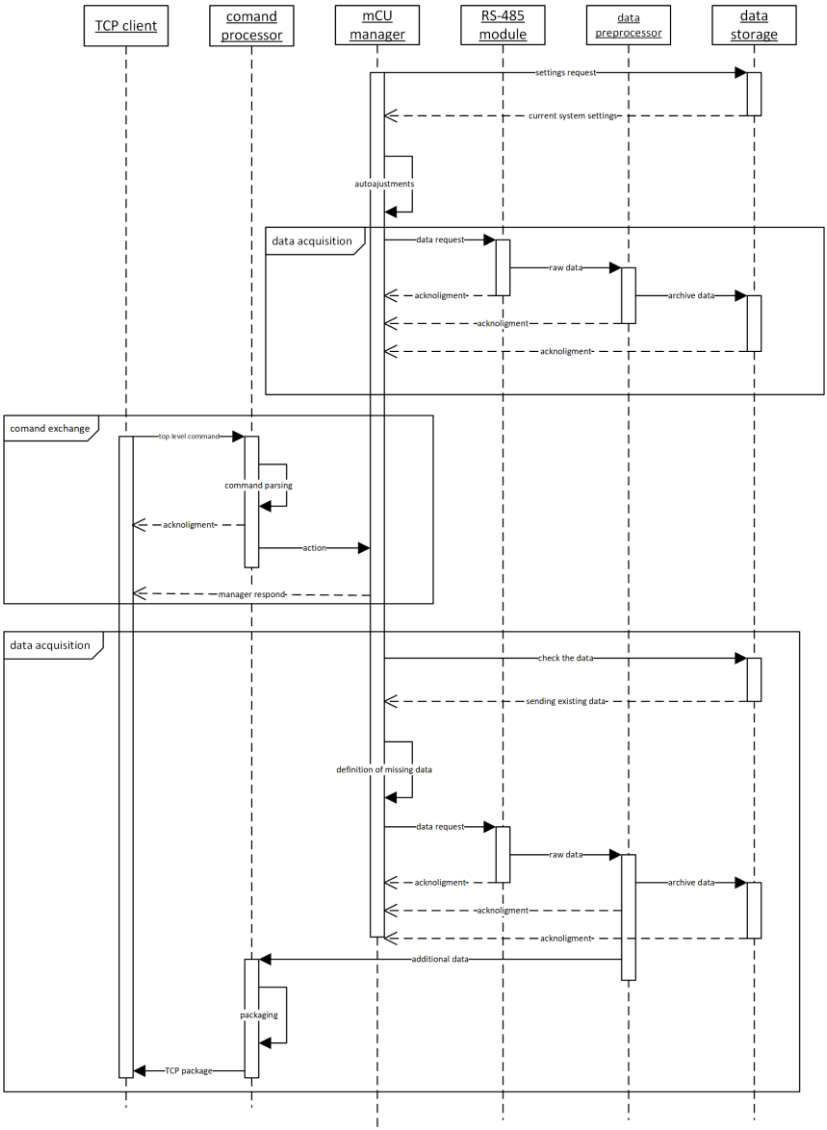

Рис. 1. Диаграмма последовательностей

К нему относится реализация стека сетевых протоколов и работы с соответствующим аппаратным обеспечением. Подготовка данных к

отправке и обработка полученных команд не относится к его функциональности.

Command processor выполняет предварительный анализ полученных от сетевого клиента данных. Далее (если данные не повреждены) он выполняет их разбор и запуск обрабатывающих функций основного менеджера. При этом данный объект подготавливает аргументы функций, а также контролирует их вызов и результаты выполнения. По результатам формируется ответное сообщение для сетевого объекта. Также рассматриваемый объект подготавливает сырые данные для отправки; выполняет их упаковку, формирует служебную информацию.

mCU manager является основным объектом нижнего уровня. По факту является центральным диспетчером. Реализует выполнение всех периодических задач. Обеспечивает отклик на сложные команды, требующие сложного взаимодействия различных объектов. mCU manager обеспечивает работу устройства в автоматическом режиме и является управляющим объектом. Однако его алгоритмы могут быть изменены посредством внешних команд.

RS-485 module реализует работу с интерфейсом RS-485 и оборудованием, работающем в системе посредством данного интерфейса. К нему относится программная поддержка аппаратных ресурсов, а также процедуры обмена с промышленным оборудованием. Основная функциональная нагрузка объекта приходится на данные процедуры, так как для выполнения достаточно простых команд и запросов данных требуется сравнительно сложная программная реализация обмена данными.

Data preprocessor – выполняет предварительную обработку данных. Фактически объект реализует цифровой фильтр. Он позволяет выявить и исправить однозначно ошибочные данные, а также определить факт недопустимого количества ошибок, получаемых от оборудования. В дальнейшем данный модуль планируется расширить дополнительными статистическими функциями.

Data storage – выполняет простейшие задачи по работе с внешней памятью. Они выделены в отдельный объект, потому что для микроконтроллера [5] работа с файловой системой, запись, чтение и редактирование файлов является достаточно сложной задачей, требующей достаточно продвинутых аппаратно-программных решений. К этому объекту относится программная реализация обмена хранимыми данными и программная поддержка аппаратного обеспечения, реализующего внешнюю память.

2. Взаимодействие между основными элементами подсистемы

Рассмотрим основные алгоритмические последовательности, описывающие взаимодействие между представленными объектами. В верхней части диаграммы на рисунке 1 показан этап инициализации микроконтроллера. mCU manager отправляет запрос о получении текущих настроек к модулю, обслуживающему хранилище данных. Ответными сообщениями центральный диспетчер (mCU manager) получает настройки и приступает к их обработке. В случае получения некорректных данных mCU manager может повторить запрос или частичный запрос.

После получения всех данных центральный диспетчер анализирует данные, сравнивая с текущими настройками устройства. Далее запускается процедура безопасной смены настроек. По ее завершении mCU manager переходит в режим непрерывного управления, выполняя все функции управления.

Даже первые рабочие версии системы имеют достаточное количество алгоритмов, которые невозможно отразить в данном материале в виде одной диаграммы. Таким образом, представленная на рисунке 1 диаграмма является значительно сокращённым вариантом. Согласно нее после инициализации запускается цикл опроса счетных устройств и сбора данных. Данный цикл является основным режимом работы подсистемы. Он выполняется автоматически с частотой опроса, определяемой настройками. Режим работы устройства позволяет не запускать цикл после старта системы и загрузки настроек. Однако данный режим не является типовым.

Цикл опроса счетных устройств начинается с формирования запроса от mCU manager. Запрос поступает объекту, ответственному за взаимодействие с аппаратными внешними ресурсами и интерфейсами. В соответствие с поступившим запросом высокого уровня данный объект формирует последовательность команд для передачи по интерфейсу RS-485 для счетных устройств. Структура команд определяется оборудованием, используемом на самом нижнем уровне для счета продукции. RS-485 модуль на основе полученного запроса формирует последовательность низкоуровневых команд и выполняет анализ полученных ответов. По результатам анализа в центральный диспетчер передается соответствующее сообщение.

При этом полученные от счетных устройств результаты не оцениваются. Предварительный статистический анализ и цифровую фильтрацию выполняет data preprocessor. Он позволяет определить наличие сомнительных и гарантированно ошибочных данных. Сведения о них поступают в центральный диспетчер, который принимает решение

о продолжении нормального режима работы, повторении запроса или других действиях.

Обработанные данные поступают в модуль работы с дисковым устройством. Они сохраняются в файл локального банка данных. Сохранение выполняется если предыдущий модуль определил полученные данные допустимыми для дальнейшей обработки. В противном случае data preprocessor не передает данные для сохранения, а выдает только отчет об ошибке центральному диспетчеру. По завершению записи data storage сообщает об успешном завершении операции mCU manager.

Следующая типовая операция рассматриваемой подсистемы – прием и передача команд посредством TCP соединения. Нижний уровень выполнен в виде клиента. Общая архитектура системы предполагает наличие единого сервера, объединяющего несколько клиентов. Таким образом, подсистема нижнего уровня должна поддерживать только одно соединение.

Классическим примером обмена командами является показанный на рисунке 1 блок. Здесь сервер передает клиенту некоторую команду. По ее получению модуль TCP клиента извлекает сообщение из пакета и передает его в модуль обработки команд. Непосредственно объект клиента не выполняет никаких функций анализа поступивших данных на верхних уровнях OSI модели. В его функциональность входит только поддержка всего стека протоколов и прием и отправка сообщений.

Командный парсер выполняет анализ команды на корректность, по результатам он отправляет модулю клиента сообщение с подтверждением получения корректной команды или об ошибке парсера. Далее объект выполняет обработку кода команды и формирует внутренние командные структуры для центрального диспетчера. Эти структуры формируют определенные действия, которые реализует mCU manager. В ответ на них он отправляет (в случае, предусмотренном алгоритмами управления) серверу подтверждения или сообщения об ошибках. Они также передаются TCP клиент объектом.

На представленной на рисунке 1 диаграмме представлена ситуация, когда полученной от сервера командой является запрос на получение результатов измерений за определенный период. Процесс получения и анализа команды сервера описаны выше. Далее mCU manager в ответ на внутреннюю команду выполняет попытку получить данные из внешней памяти. Для этого в объект data storage отправляется соответствующий запрос. Ответом служит имеющийся в дисковой памяти набор данных.

В неаварийном режиме работы устройства при запросе данных из предыдущих периодов все данные должны быть получены на этом

этапе. Однако вывод об этом может сделать только центральный диспетчер, который анализирует полученную информацию. В случае обнаружения фатальных ошибок данных в дисковой памяти будет выполнена попытка повторного опроса счетных устройств, если данный период времени еще может хранится в памяти счетчиков.

Мы считаем, что вероятность выполнения данного сценария мала. Однако велика вероятность поступления запроса данных, которые еще не были получены контроллером от счетных устройств и сохранены в память. В этом случае также потребуется запрос получения новых данных. Его процесс практически идентичен описанному ранее блоку сбора данных. После выдачи команды центрального диспетчера модуль аппаратного интерфейса формирует стандартный набор команд получения данных от оборудования. Полученные данные аналогично обрабатываются препроцессором. Однако результат не только записывается во внешнюю память, но и параллельно отправляется для отправки сетевым клиентом.

Перед отправкой полученные данные упаковываются объектом command parser. Процесс упаковки полностью реализуется средствами данного объекта без внешнего контроля. По завершению упаковки данные отправляются через модуль TCP клиента.

Заключение

Размер диаграммы, показанной на рис. 1, не позволяет представить все типовые сценарии взаимодействия в рамках рассматриваемого элемента подсистемы. Однако они выполняются по схожим методикам и соизмеримо задействуют функционал рассмотренных объектов. Структура представленной диаграммы позволяет ее расширение вниз путем добавления всех необходимых функций.

Недостатком текущей схемы является укрупненный объект mCU manager, который фактически координирует все информационные потоки внутри подсистемы. В настоящее время на начальном этапе разработки системы решено представить ее структуру в данном виде. Однако при расширении ее функциональности центральный диспетчер будет разбит на несколько более узкоспециализированных объектов. Эти изменения потребует создания новой диаграммы последовательностей.

Созданная диаграмма последовательностей является важным документом на этапе проектирования информационной системы и будет использована в разработке. Однако данный документ не является конечным перед реализацией. Также необходимо создать диаграмму классов и ряд развернутых графических представлений алгоритмов для начала реализации первого этапа разработки информационной системы.

Разработанная диаграмма последовательности ляжет в основу данных проектных документов.

В дальнейшем в ходе модернизации разрабатываемого продукта представленные диаграммы будут изменены. Однако представленные в работе материалы представляют базовую структуру системы, вероятность изменения которой низка. Уровень представленных в работе проектных изысканий достаточен для реализации коммерческого программного продукта.

Список литературы

1. Иванов, К. К. Проектирование информационных систем / К. К. Иванов. // Молодой ученый. — 2017. — № 19 (153). — С. 22-24.
2. Манако А.Ф. Подход к построению формализованного описания информационных систем для образования и обучения // Образовательные технологии и общество. 2013. Т. 16. № 1. С. 536-546.
3. Ларман, К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / К. Ларман. - М.: Вильямс, 2013. - 736 с.
4. Буч, Г. Язык UML. Руководство пользователя / Г. Буч , Д. Рамбо , А. Джекобсон. - М.: ДМК, 2015. - 432 с.
5. STM32 32-bit Arm Cortex MCUs [Электронный ресурс] : техническое описание. – Режим доступа : https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html