УДК 621.395.345

Б.С. Гольдштейн

ТЕХНОЛОГИЧЕСКИЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЦИФРОВЫХ СИСТЕМ КОММУТАЦИИ

Опыт проектирования коммутационных узлов и станций с программным управлением показывает существенное отставание темпов развития программного обеспечения (ПО) от прогресса в области создания аппаратных средств. Прослеживается основная тенденция в области разработки систем ПО, заключающаяся в том, что создание эффективных и надежных систем все меньше зависит от искусства программирования и все больше от технологии ("инженерии") создания программного продукта. При этом все более значительную роль играют стандартизация, спецификация и верификация ПО.

Другой вывод из тех же предпосылок приведен в качестве одного из основных тезисов монографии Б. Шнайдермана [1]: "Программы нельзя производить на сборочном конвейере, как автомобили, нельзя и измерить производительность числом строк или команд, написанных за час. Понимание того, что для проектирования и разработки программного обеспечения требуется высокий интеллект и мастерство, позволит администраторам создать особые условия программистам с тем, чтобы обеспечить качество программ". Однако, если даже представители администраций связи обратят внимание на эту цитату и сделают из нее соответствующие выводы (на что автор не устанет надеяться последние 15 лет), одних этих мер недостаточно для преодоления сложившейся кризисной ситуации в проектировании ПО систем коммутации.

Как известно, создание систем ПО узлов коммутации относится к классу сложных слабоструктурированных проблем, и правильное конструирование ПО невозможно без предварительной структуризации технологической системы его разработки.

В ЛОНИИС использована методология проектирования ПО систем коммутации, в соответствии с которой согласовывается и до некоторой степени дисциплинируется творческая активность разработчиков, что обеспечивает последовательное продвижение процесса проектирования всегда в сторону уменьшения его неопределенности и увеличения согласованности отдельных его этапов при тщательном документировании результатов каждого из них. На этом пути много трудностей и нерешенных задач, встречающихся на каждом этапе проектирования.

Пока не существует общепринятого подхода к решению вопросов, на какие этапы нужно разделять процесс проектирования, какую документацию следует выдавать на каждом этапе и по каким критериям оценивать его завершенность. Согласно рассматриваемой методологии при разработке проектов систем коммутации автор исходил из схемы процесса разработки ПО, приведенной на рис. 1, в основе которой лежит фундаментальная идея нисходящего проектирования (метод сверху – вниз).

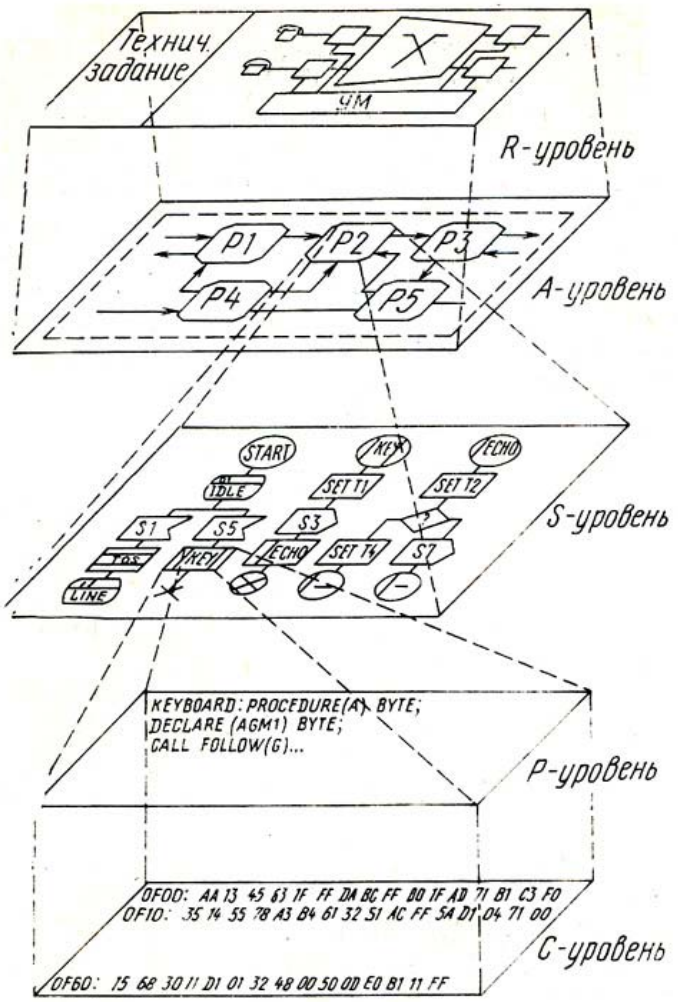


Рис. 1

В схеме предусмотрена иерархическая декомпозиция процесса разработки на последовательность шагов, уточняющих проект. Такими укрупненными шагами (уровнями проектирования) являются определение и формализация требований и интерфейсов коммутационного оборудования (R- уровень), архитектура (системная и функциональная) и модульная структура ПО (A- уровень), разработка SDL-спецификаций модулей (блоков, процессоров, процедур, макросов, структур данных) и межмодульных интерфейсов (S- уровень), написание и отладка программ (P- уровень), непосредственно реализация в машинных кодах для объектовых машин распределенной системы управления узла коммутации (C- уровень).

Уровни проектирования различаются как степенью конкретизации (возрастающей сверху вниз), так и языковыми средствами описания. Таким образом, представление системы ПО на вышестоящем уровне является в известном смысле "общим прародителем" семейств представлений нижестоящих уровней. Фактически на всех уровнях проектирования, а не только на S- уровне осуществляется последовательная спецификация задач, решаемых программным обеспечением. Под спецификацией здесь понимается описание высокого уровня в терминах, характерных для самой задачи, а не для ее реализации, служащее основой для дальнейшей детализации и разработки ПО [2]. Можно считать, что каждый уровень проектирования получает спецификации от

предшествующего ему уровня и вырабатывает данные, в свою очередь, необходимые для спецификации одного (или более) из последующих уровней.
Отличительные свойства спецификаций – однозначность, точность, формальность, понятность и читабельность. Как отмечено В. Н. Агафоновым [2], язык программирования более высокого уровня может считаться языком спецификаций по отношению к языку более низкого уровня (например, Паскаль по отношению к Фортрану или ПЛ/М по отношению к Ассемблеру). При этом спецификация программного модуля не обязана быть короче самого модуля, ибо от спецификации требуется не кратность, а точность и понятность.

Определение и спецификация требований к ПО узла коммутации являются основными задачами R – уровня проектирования. На этом уровне разрабатываются технические требования, структурная схема станции, интерфейсы с коммутационным оборудованием и т.д. Языком проектирования, как правило, служат естественный язык со всеми присущими ему недостатками, среди которых следует выделить неоднозначность, связанную с тем, что естественный язык недостаточно точен для описания систем ПО и разные разработчики могут по разному понять одну и ту же фразу ТЗ, и неполноту описания ПО, которая усугубляется тем, что при разработке большой и сложной системы ПО узла коммутации проходит много времени, прежде чем становится ясно, какой информации R- уровня недостает.

Другая трудность, возникающая на R- уровне проектирования, – невозможность удержания семантики описания требований на одном адекватном уровне детализации. В результате одни описания R – уровня описывается несколько туманными, другие 0 несколько детализованными, что по существу предоставляют собой элементы реализации, причем, возможно, не самой удачной, выбранной без рассмотрения остальных частей системы и не позволяющей разработчикам следующих уровней проектирования использовать эффективные структуры данных или приемы программирования.

В последнее время предпринят ряд попыток усовершенствовать R- уровень проектирования путем создания более формальных методологий, обеспечивающих точность, непротиворечивость и полноту проектирования. Примером такой формализованной методологии может служить система SREM (Software Requirement Engineering Methodology), разработанная в Центре новой техники противоракетной обороны США [3]. Документация требований к системе противоракетной обороны (ПРО) содержит 8248 параграфов, для которых SREM обеспечивает автоматизированную проверку самих параграфов и их взаимосвязей, а также верификацию и тестирование спецификаций. С учетом соизмеримости данного программного проекта с ПО современного узла коммутации уместно провести еще один факт из этого же источника: в течение последнего десятилетия затраты ВВС США и НАСА на программное обеспечение более чем вдвое превышали затраты на аппаратуру.

После завершения R- уровня проектирования, т.е. когда точна внешняя спецификация системы программного управления коммутационного узла заменяет ее неформальное описание, начинается разработка архитектуры ПО (A- уровень).

A- уровень проектирования можно условно разделить на два подуровня: разработка функциональной архитектуры (AF – подуровень) и разработка системной архитектуры (AS- подуровень). Принципы проектирования обоих этих подуровней перетерпели за

последние 15 лет принципиальные изменения. Вычурные системы решения, хаотические управляющие структуры и тысячестрочные подпрограммы сменились тщательно определенными и хорошо документированными функциональными модулями. Широко внедрилась в практику А- уровня проектирования концепции виртуальных машин. Произошло заметное смещение критериев проектирования: алгоритмам управления ресурсами управляющих процессов отводится значительно меньшая роль, чем проблемам структуризации системы и взаимодействия процессов.

Трудности А- уровня проектирования связаны и с тем, что при создании каждого нового поколения коммутационных узлов необходимо пересматривать вопросы общей эффективности ПО и распределения функций между программной и аппаратной реализацией. Для решения вопроса об аппаратной реализации некоторой функции управления узлом целесообразно руководствоваться следующими критериями. Функция должна быть достаточно простой, при этом необходимость ее изменения в будущем должна быть маловероятной, а меньшая скорость выполнения этой функции в случае ее реализации программными средствами – неэффективной.

После определения программно-аппаратных интерфейсов на А- уровне проектирования разрабатывается структурная модель системы ПО, состоящая из иерархии содержательных функций, эффект выполнения которых влияет на функционирование коммутационного оборудования узла и обслуживание вызовов. Такая структурная модель в рекомендованном МККТТ языке спецификаций и описаний SDL называется диаграммой дерева блоков.

Блок представляет собой наиболее крупный объект в SDL, который, в свою очередь, может быть разбит на субблоки, каждый из которых содержит один или несколько процессов. Разбиение системы ПО на составные части осуществляется таким образом, чтобы каждая из частей была небольшой, удобной для восприятия, соответствовала естественному функциональному разбиению, а связи между частями, возникающие в результате разбиения, были бы как можно более слабыми.

На каждом этапе разбиения специфицируются также следующие компоненты: каналы, входные сигналы, выходные сигналы и данные.

Формально процесс разбиения комплекса ПО на n блоков, в совокупности выполняющих функции комплекса, может быть записан в виде рекурсивной процедуры

```
PROCEDURE AF (I,0)
```

`I` – входные сигналы, `0` – выходные сигналы,
`IF` – размерность {1} мала `THEN 0` определяются непосредственно.
`ELSE` блок делится на n субблоков с входными сигналами I (1), I (2), …, I (n) и выходными сигналами 0 (1), 0 (2), …, 0 (n) так, чтобы AF[I (1), 0 (1)], [I (2), 0 (2)], …, AF [I (n), 0 (n)] в совокупности составляли 0 = {0 (1), 0 (2), …, 0 (n)} и AF (I, 0).
`ENDIF`

```
ENDPROCEDURE
```

Завершается А- уровень проектирования разработкой программной документации типа А, главную роль в которой играют диаграмма взаимодействия процессов и прикладываемые к этой диаграмме список процессов, спецификации каналов и сигналов, словарь глобалов и структура данных станционного файла. На рис. 2 в качестве примера приведен фрагмент диаграммы взаимодействия процессов реального

ПО терминального модуля городской АТС. Программная документация А- уровня служит исходными данными для проектирования SDL- спецификаций программных процессов, процедур и макросов, что в отечественной литературе объединяется под названием алгоритмического обеспечения АТС. По поводу последнего термина автор считает необходимым отметить следующее.

Понятие алгоритм в его общем виде принадлежит к числу основных первоначальных понятий математики, не допускающих определения в терминах более простых понятий. Неформально алгоритм можно определить как совокупность правил, определяющих эффективную процедуру решения любой задачи из некоторого заданного класса задач. Сам термин произошел от арабского имени великого узбекского математика IX века Мухаммеда аль Хорезма и, следовательно, известен достаточно давно, но как математические объекты алгоритмы исследуются с 30-х годов нашего столетия. Уточнения понятия алгоритма основываются, в частности, на понятиях частично-рекурсивной функции или машины Тьюринга.

Нетрудно показать, что составление алгоритма управления коммутационным узлом является в математическом смысле алгоритмически неразрешенной проблемой. Действительно, область аргументов такого алгоритма обязательно должна включать состояния и текущие значения реального времени функционирования узла коммутации. С другой стороны, доказан известный тезис Черча, утверждающий, что любая вычислительная арифметическая функция является частично-рекурсивной. Следовательно, алгоритмическая неразрешимость проблем вытекает из простых мощностных соображений: всех арифметических (числовых) функций – континуум, а частично-рекурсивных – счетное число.

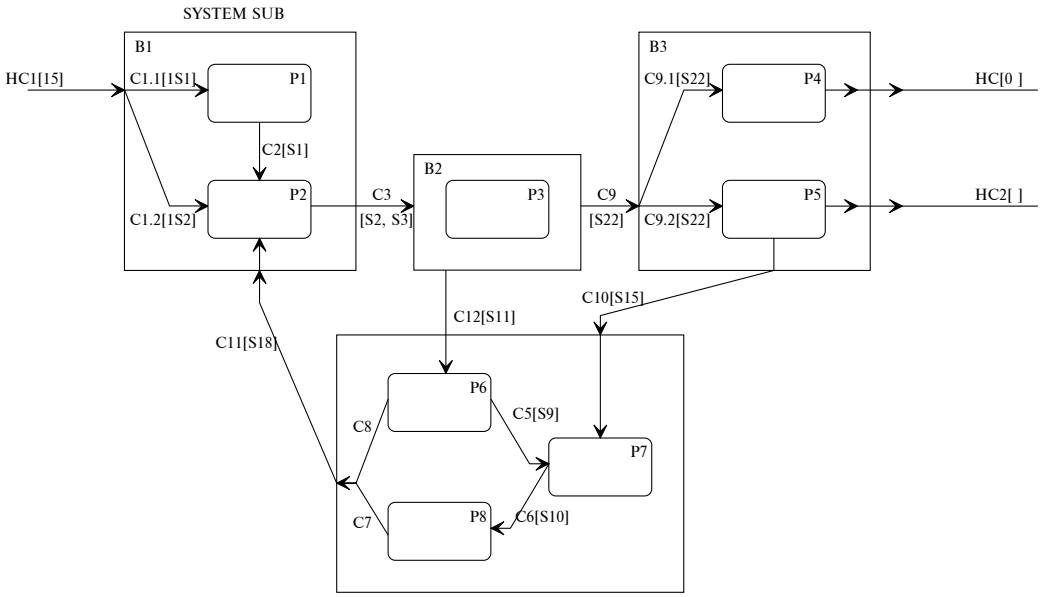


Рис. 2

Тем не менее, термин "алгоритмическое обеспечение" прочно укоренился в лексиконе специалистов по программному обеспечению систем коммутации. Интуитивно под этим термином понимается спецификации ПО узла коммутации. Именно этот последний термин используется в настоящей статье.

Детальное проектирование S-уровня включает уточнение спецификаций интерфейсов программных модулей, структур данных и проектирование SDL-

диаграмм модулей. При уточнении интерфейсов окончательно определяются порядок и структура параметров, глобалов и сообщений, составляющих интерфейс.
Проектирование SDL- диаграмм на S- уровне также выполняется сверху вниз методом пошаговых уточнений. При этом на каждом уровне декомпозиции используются следующие компоненты: каналы, сигналы, процессы, процедуры, макроопределения. Точно так же, как конструкторские чертежи в технике, SDL- диаграммы – это не просто картинки, а законченный и богатый язык.

Механизмы SDL обманчиво просты, обладают большой выразительной силой, что делает SDL естественным и удобным для применения. Требуется совсем небольшая практика, что бы научиться читать на SDL и точно воспринимать информационное содержание, передаваемое графическими обозначениями и словами языка. Однако, будучи языком с точно определенной семантикой, SDL налагает жестокие ограничения на правила истолкования смысла спецификаций, а также и на правила написания SDL-диаграмм (синтаксис языка SDL) [4].

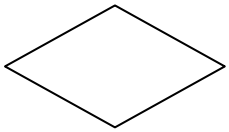
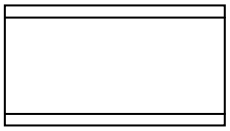
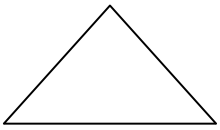
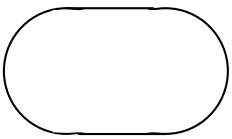
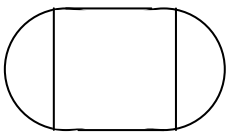
Необходимо отметить, что разработка языка SDL проводится с 1968 г. Первая версия языка SDL была опубликована в 1977 г., вторая – в 1982 г., а третья, расширенная и модернизированная, – в 1985 г. Согласно четырехлетним планам работ МККТТ этим версиям присвоены наименования соответственно SDL-76, SDL-80, SDL-84. Первые две версии представляли собой средства полуформального описания систем коммутации с помощью преимущественно графического псевдокода. В SDL-84 возможности формального структурированного описания систем развиты существенно глубже, вплоть до создания полностью формализованных и таким образом выполнимых спецификаций ПО систем коммутации. В настоящей статье рассматривается только язык SDL-84. Чрезвычайно важным преимуществом языка является наличие в нем трех эквивалентных синтаксисов: графического – SDL/GR, программоподобного – SDL/PR, и абстрактного. В таблице представлены основные символы SDL/GR и эквивалентные им операторы SDL/PR [4], а на рис. 3 – фрагмент SDL- диаграммы процесса управления логикой сигнализации на SDL/GR.

Таблица

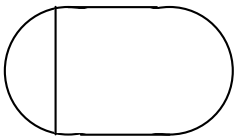
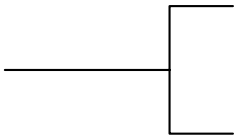
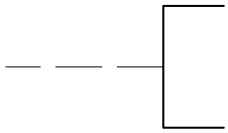
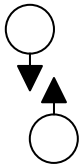
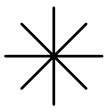
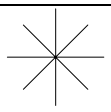
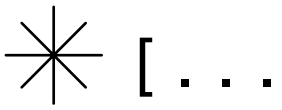
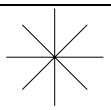
SDL/GR	SDL/PR	Значение символов
	STATE NEXTSTATE	Состояние, следующее состояние
	TASK	задача

Продолжение таблицы

SDL/GR	SDL/PR	Значение символов

	INPUT	ввод
	OUTPUT	вывод
	SAVE	сохранение
	DECISION	решение
	CALL	вызов процедуры
	MACRO	вызов макро
	CREATE	запрос создания процесса
	ALTERNATIVE	опция
	STOP	останов
	RETURN	возврат из процедуры
	ENDMACRO	выход из макро
	START	старт процесса
	PROCEDURE	начало процедуры

Продолжение таблицы

SDL/GR	SDL/PR	Значение символов
	MACRO EXPANSION	вход в макро
		расширение текста
	COMMENT	комментарии
	X: JOIN X	входной соединитель выходной соединитель
		все
	 [. . .]	все, кроме
	PROVIDED	непрерывный сигнал

Этап написания и отладка программ (**Р- уровень проектирования**), по различным опытным данным, составляет 30-40 % от общего объема работ по созданию ПО узла коммутации. До недавнего времени программирование микропроцессорных управляющих устройств систем коммутации велось на языке Ассемблер. Значительное улучшение характеристики микро ЭВМ привело к возможности эффективного использования языков высокого уровня, в состав которых входят рекомендуемый МККТТ язык CHILL, язык Паскаль, ПЛ/М, Си, Форт и др.

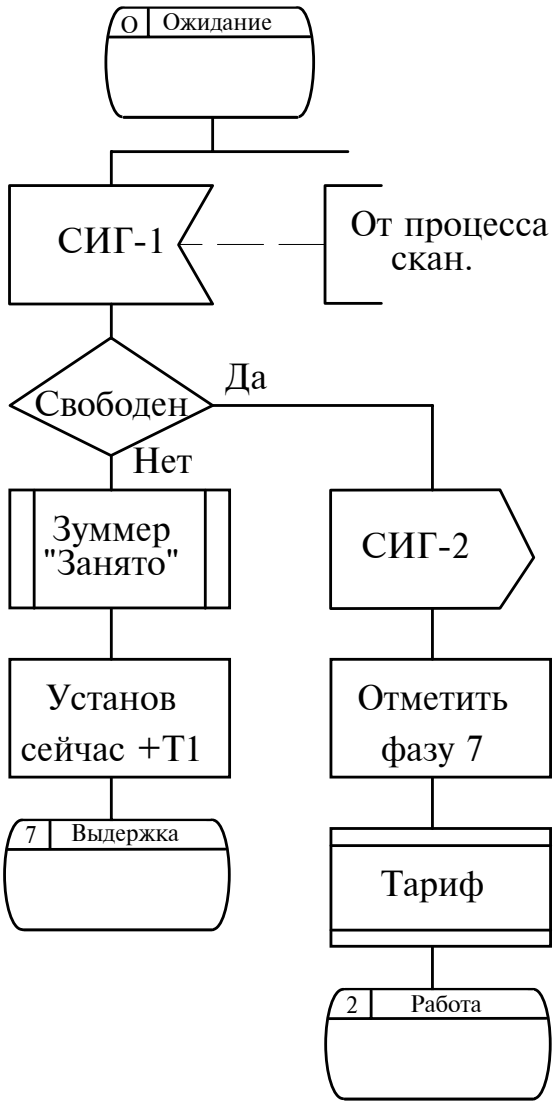


Рис. 3

Заметим, что включение в состав вышеупомянутых элитных языков стековой интерпретирующей системы Форт обусловлено ее полной структурированностью, расширяемостью, транспортабельностью (за исключением небольшого машинозависимого ядра) и простой синтаксиса. Кроме того, Форт дает возможность разработать словарь специально для конкретной телефонно-справочной области и представляет разработчику доступ ко всем разделам памяти и портам ввода-вывода.

Даже самая поверхностная сравнительная оценка упомянутых языков программирования выходит за рамки данной статьи, поэтому ниже отмечаются только два аспекта этой проблемы.

Существенное значение в программах управления узлами коммутации имеют конструкции условных операторов. В настоящее время можно считать доказанным преимущество вложенных операторов типа IF-THEN-ELSE (используемых в языках Паскаль, ПЛ/1) перед формой IF-GOTO (используемой в языках Фортан и Бейсик). Еще большее значение имеет конструкция CASE. Ее достоинство при программировании задач обслуживания вызовов состоит в том, что разветвление на много путей осуществляется единственным хорошо организованным оператором, которому соответствует явный оператор END. Удобная форма задания предшествующих логических условий и комментарии улучшают читабельность такого синтаксиса.

Второй, более общий аспект – постепенный отход от сложившейся в 60-х годах программистской практики, рассматривавшей разработку ПО как результат обособленного труда программистов. Создание ПО узла коммутации является сугубо коллективной деятельностью. Точно так же язык программирования следует рассматривать не только как средство общения программиста с ЭВМ, но и как язык общения программистов между собой с помощью ЭВМ, но и как язык общения программистов между собой с помощью ЭВМ. Следовательно, наиболее значительными качествами программ становятся их структурированность и читабельность, обеспечиваемая языками высокого уровня.

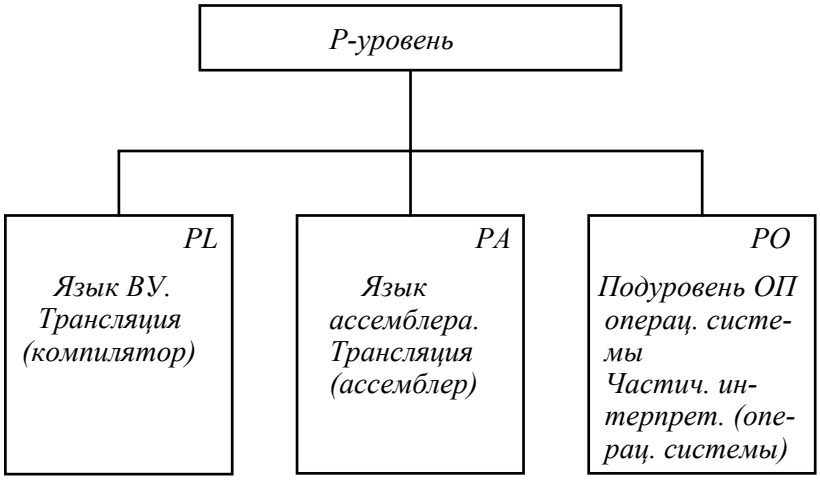


Рис. 4

Р-уровень проектирования можно условно представить в виде совокупности трех подуровней (рис. 4). Такой подход позволяет ослабить взаимосвязи технологической цепочке программирования, в частности не обусловливать заранее областей постоянной, оперативной и внешней памяти, доступные каждой группе программ, не вводить ограничения на использование идентификаторов, применять различные языки программирования и т.п.

При этом следует отметить [5], что уровень операционной системы может только частично "экранизировать" возможности аппаратуры. По аналогии с этим работа с языками программирования высокого уровня не исключает в ряде случаев непосредственного обращения к аппаратным интерфейсам.

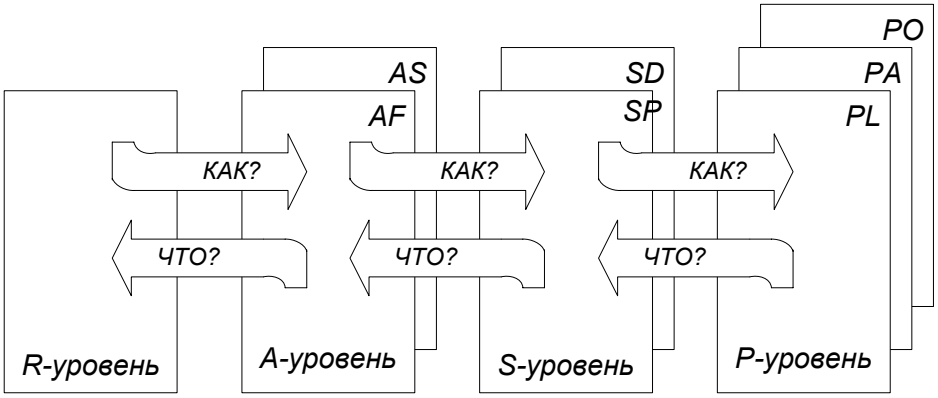


Рис. 5

Изложенные выше методологические принципы структурированного, иерархического упорядоченного проектирования ПО соответствуют шкале ЧТО-КАК (рис. 5).

Поднимаясь вверх по иерархии, можно уяснить, ЧТО делает данный программный компонент, опускаясь вниз, можно понять, КАК этот компонент реализуется.

Другая, более важная цель пошагового продвижения вверх по иерархической лестнице проектирования от C- уровня к R- уровню заключается в анализе корректности преобразований, т.е. в анализе правильности соответствия спецификации ее реализации на данном уровне проектирования. При этом ни один из методов анализа правильности результатов разработки каждого уровня не является исчерпывающим, подтверждающим абсолютную корректность преобразований.

Так, тестирование может выявить только факт наличия, а не отсутствия ошибок. Например, Маккейб в [6] обращает внимание на то, что модуль в 50 строк, содержащий 25 конструкций IF-THEN, может содержать 33.5 мил разных путей выполнения.

С другой стороны, формальные методы верификации позволяют доказать правильности простых программных сегментов. Между тем не существует адекватных, практически приемлемых методов верификации сложных программных комплексов узлов коммутации. Поэтому необходимо использовать целый ряд дополнительных друг друга методов обнаружения, локализации и исправления ошибок ПО [7], многократно продвигаясь по уровням проектирования в соответствии с рис. 5.

Кроме указанных уровней проектирования и соответствующих этим уровням видов деятельности, создание сложного ПО систем коммутации неразрывно связано с такими вопросами, как инструментальные системы поддержки проектирования, организованная структура коллектива разработчиков ПО и документирование программного проекта.

Инструментальная система поддержки проектирования представляет собой совокупность языковых, программных и аппаратных средств, предназначенных для разработки и отладки ПО на всех уровнях проектирования и ведения базы данных проекта (рис. 6). В состав изображенной на рисунке инструментальной системы включены графические средства представления SDL- спецификаций, стандартизированные таблицы сигналов, состояний, глобалов и констант станционного файла, форматеры текстовой документации и т.п. К языковым средствам относятся языки программирования, языки описания данных, языковые транслятор SDL/PR в DSL/GR и языки управления самой инструментальной системой.

Для выполнения проектов ПО систем коммутации требуется создание специальной тщательно продуманной программы и структурированной организации, что, по мнению экспертов, является более важной и ответственной задачей, чем непосредственно разработка систем коммутации. При этом согласно закону Брукса [8] простое увеличение числа разработчиков или накопление средств вычислительной техники не обеспечивают желаемого эффекта. Важное значение при создании "прозрачной" сверху организационной структуры имеют подбор и расстановка кадров, психологический климат, квалифицированное управление проектированием и обоснованность плановых сроков.

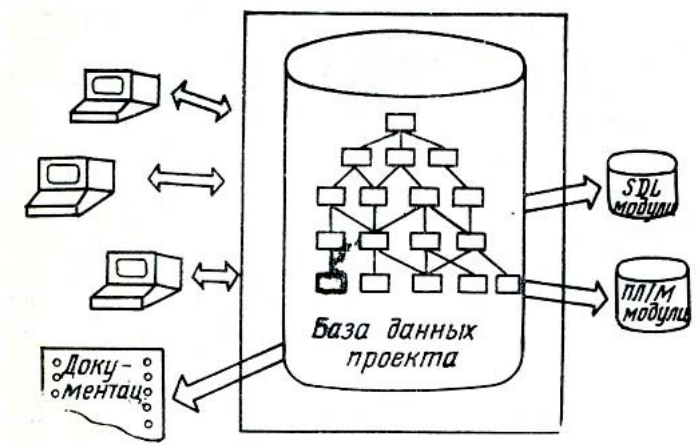


Рис. 6

Последними из перечисленных вопросов, неразрывно связанных с технологией проектирования ПО, является составление **программной документации** (ПД). С учетом спецификации производства и внедрения коммутационных узлов и станций выделим эксплуатационную (внешнюю) и внутреннюю техническую ПД.

Внешняя ПД определяется различными регламентирующими требованиями ЕСПД и Рекомендацией МККТТ серии Z, но, к сожалению, считается второстепенной работой, которая практически не связана с реализацией самого программного проекта. При таком подходе из-за постоянной нехватки времени на разработку ПО выпуск документации переносится на конец разработки, и сбор информации осуществляется вручную, причем не самым квалифицированным специалистом. Значительная часть промежуточных проектных решений к началу документирования теряется, обоснование выбора тех или иных решений не поддается восстановлению, что приводит к значительному увеличению трудоемкости составления программной документации и ее низкому качеству.

Еще большие сложности возникают при составлении внутренней (технической) документации. В известных автору проектах предпринимались попытки заменить документы устным общением между участниками проекта или рабочими материалами одноразового использования. Но руководство проекта лишалось реальной возможности следить за ходом разработки и управления проектом, а ПО оказывалось не готовым в планируемые сроки и не выполняло всех функций, обусловленных R-уровнем. Существенным недостатком такого прохода является зависимость одних разработчиков от других, более осведомленных специалистов, которые могут отсутствовать в тот момент, когда возникает потребность в той или иной информации о системе.

В конечном итоге такая практика приводит к растягиванию на многие месяцы (а иногда и годы) полного героики и самопожертвования волнующего этапа "комплексной отладки с оборудованием узла". Излагаемые в данный статье аспекты L-технологии разработки ПО систем коммутации основываются на машинном проектировании текстовых и графических документов, которое позволяет совместить процесс документирования с процессом разработки и обеспечения поддержку актуального состояния программной документации, сократить и снизить затраты н производство ПД. При этом ПД должна храниться на машинных носителях и широко использовать

шаблоны (развернутые планы соответствующих документов, содержащих указания по написанию пунктов этих документов).

Для хранения всей основной информации о программном проекте используется единая база данных, и результаты каждого уровня проектирования (см. рис. 1) сразу после их получения предоставляются в форме, пригодной для их обработки на ЭВМ. При необходимости получения твердой копии программного документа она также выделяется с помощью ЭВМ.

Заключение. Рассмотренный подход к проектированию ПО систем коммутации включает достаточно широкий спектр методов и средств разработки спецификаций, кодирования, диагностирования, отладки и документирования. Эффективность применения этих методов и средств в каждом случае может быть установлена только с учетом реальной проектной программатики, в связи с чем был произведен эксперимент, направленный на повышение производительности труда программистов и получение хорошо документированного программного продукта, удобного в сопровождении.

Эксперимент проводится при разработки программного комплекса ЭВМ обслуживания вызовов унифицированной цифровой ступени коммутации справочно-информационных служб "09". Для реализации комплекса был выделен коллектив разработчиков, выполняющих ранее программные проекты на ЭУМ "Нева-1", не имевших большого опыта разработки программных систем на микроЭВМ и в основном лишь о теоретически знакомых с языками программирования ПЛ/М-80 и Ассемблер 8080, принятых как основные для проекта.

Объем проекта составил 93500 операторов. Объем документации – около 800 листов. Пакет детализированных SDL- диаграмм содержал 483 листа 11-го формата. В результате эксперимента, проведенного с привлечением описанных методов, производительность методов, производительность труда при разработке пакета ПО в 1,8 раза превысила нормативную.

ЛИТЕРАТУРА

1. Шнайдерман Б. Психология программирования. Человеческие факторы в вычислительных и информационных системах. Пер. с анг. М.: Радио и связь, 1984.
2. Требования и спецификации в разработке программ. Пер. с анг. Под ред. В.Н. Агафонова. – М.: Мир, 1984.
3. Bell T.F., Bixeler D. C.; Dyer M.E. An Extendable Approach to Computer. – Aided Software Requirements Engineering. – IEEE Transactions on Software Engineering, 1977, v. SE-3, №1.
4. CCITT red book. Recomendations Z100-Z104. – Geneva. 1985.
5. Гольдштейн Б.С. Телефонная операционная система электронного узла коммутации. – Электросвязь, 1980, №8.
6. Mc Cabe T. A complexity measutr.- IEEE Trznsactions on Software Engineering, 1976, v. SE-2, №6.
7. Апостолова Н.А., Гольдштейн Б.С. Прогнозирование числа ошибок при отладке информационно связных программных комплексов реального времени. – В сб.: Проблемы совершенствования синтеза, тестирования, верификаций и отладки программ. Т. 1. – Рига: ЛГУ им. П. Стукачи, 1986.
8. Брукс Ф.П. Как проектируются и создаются программные комплексы? – М.: Наука, 1979.

Статья поступила 23 июля 1987 г.