Кросс-система программирования ЯУЗА-6 для специализированных ЭВМ реального времени
(в 70 – 80-е годы прошлого века)

В.В. Липаев
lip@ispras.ru

Аннотация. Рассмотрены особенности и проблемы эффективного создания в 70-е годы сложных комплексов программ оборонного применения, базирующихся на специализированных ЭВМ. Представлены методы, требования и реализация адаптируемых кросс-систем автоматизации программирования и тестирования для различных типов сложных комплексов программ объектных ЭВМ. Изложены результаты многолетнего применения в ряде предприятий кросс-системы автоматизации программирования и отладки ЯУЗА-6 с использованием БЭСМ-6.

Ключевые слова. Кросс-система программирования; комплексы программ; автоматизация программирования и отладки программ; специализированные ЭВМ; трансляторы и интерпретаторы программ; автокоды и макроязыки программирования.

1. Введение

В 70-е годы программная инженерия оказалась наиболее востребованной для решения крупных задач в оборонной технике и для государственных административных систем [1]. Массовое прикладное программирование в академических организациях, промышленных институтах и в вузах в это время оставалось на уровне индивидуального создания небольших программ и обычно не требовало применения методов программной инженерии. Опыт предприятий оборонной промышленности, накопленный в 60-е годы, позволил обобщить особенности и проблемы развития программной инженерии для эффективного создания наиболее сложных комплексов программ оборонного применения, базирующихся на специализированных ЭВМ.

Быстрый рост сложности функциональных задач и требований к ресурсам ЭВМ для их решения в ***60-е годы***, не могли быть удовлетворены при доступной в то время технической и элементной базе вычислительных машин. Эту проблему разработчики систем стремились решать путем адаптации архитектуры ЭВМ к особенностям и характеристикам сложных

функциональных задач. Поэтому в ряде предприятий оборонной промышленности еще в конце ***50-х годов*** начали разрабатываться многочисленные специализированные ЭВМ. При подготовке требований к таким объектным ЭВМ военного назначения для эффективного использования их ограниченных вычислительных ресурсов, необходим был детальный анализ алгоритмов и программ, предназначенных для функционирования в реальном времени.

Относительно узкая ориентировка каждого типа ЭВМ на совершенно определенные функциональные задачи, давала возможность значительной экономии оборудования и улучшения использования характеристик ЭВМ по памяти и производительности ***на имевшейся элементной базе***. С другой стороны, в то время от программистов требовалась максимальная эффективность использования доступных ресурсов и знаний тонкостей архитектуры специализированных ЭВМ при реализации алгоритмов, что, в частности, определило широкое применение машинно-ориентированных языков программирования – автокодов.

Ресурсы объектных ЭВМ позволяли размещать программы только функциональных задач оборонных систем, а для автоматизации их программирования и отладки приходилось использовать ***стационарные универсальные ЭВМ с большими ресурсами***. Принципиально новые возможности открылись в начале ***70-х годов*** в связи с появлением в НИИ-5 (МНИИПА) ЭВМ БЭСМ-6. Началось создание ***системы автоматизации разработки и отладки программного обеспечения (САРПО)*** ЯУЗА-6 (под руководством В.В. Липаева, Л.А. Серебровского) [1, 2]. Эти работы были активно поддержаны со стороны руководства института и министерства. Министерством радиопромышленности СССР было открыто достаточное финансирование работ и значительно увеличен коллектив специалистов.

Были сформулированы следующие ***основные концептуальные особенности САРПО ЯУЗА-6***:

- возможность автоматизированной адаптации системы на структуру и архитектуру команд различных специализированных ЭВМ;
- применение трех входных языков программирования для специализированных ЭВМ – автокода, макроязыка и алгоритмического языка;
- обеспечение высокого качества программ по использованию памяти и производительности ЭВМ, транслированных с алгоритмического языка (коэффициент расширения в пределах 1,1 – 1,2);
- автоматизированную стыковку программных компонентов по глобальным переменным и по передачам управления;
- автоматизированный контроль структурного построения и

использования памяти программными компонентами;

- автоматизированное тестирование и отладка программ с использованием стационарной универсальной ЭВМ на уровне входного языка программирования с интерпретацией команд специализированной ЭВМ;
- автоматизированный выпуск документации на весь комплекс программ и на отдельные компоненты в соответствии с ГОСТами и пригодной для ввода программ в специализированную ЭВМ.

Перечисленные особенности отличали САРПО ЯУЗА-6 от обычных трансляторов как ***комплексную систему программной инженерии***, предназначенную для автоматизации основных этапов разработки крупных комплексов программ, при условии высокой эффективности создаваемых программ по использованию ресурсов специализированной ЭВМ. Комплексная автоматизация должна была в несколько раз повысить производительность труда специалистов при разработке больших управляющих программ (размером порядка 100 тыс. команд). При этом должны были быть исследованы, и учитываться принципиальные особенности специализированных ЭВМ и комплексов программ оборонных систем того времени [1, 2].

Значительное расширение функций автоматизации разработки программ на мощных технологических ЭВМ (БЭСМ-6), а также быстрый рост числа различных структур специализированных ЭВМ, для которых была необходима автоматизация программирования, приводит к сокращению изменяемой части инструментальных систем разработки программ. В результате ***экономически целесообразно*** выделять в инструментальных системах программирования ***машинно-ориентированную на специализированные ЭВМ часть программ***, и отдельно автоматизировать их разработку. Примером может служить система ЯУЗА-6, в которой из 400 тысяч команд, для адаптации на специализированные ЭВМ в ней оказалось необходимо изменение около ***3 – 5% команд***. Для автоматизации такой настройки целесообразно было разработать дополнительную подсистему размером около 20 тысяч команд и соответствующие инструкции по её применению.

Основное достоинство разработанного и исследованного оригинального метода адаптируемых кросс-систем состояло в том, что эти системы могли быть ориентированы практически на любые типы специализированных ЭВМ, поддерживая при этом эффективность результатов программирования на достаточном уровне. Процесс адаптации кросс-систем был формализован и автоматизирован. При разработке адаптируемой кросс-системы в её машино-зависимые технологические программы были введены проблемно-ориентированные средства, предназначенные для обеспечения автоматизации адаптации. Адаптируемость кросс-системы достигалась совокупным

применением принципов параметризации, конфигурирования и функционального расширения. Методы подготовки адаптируемой кросс-системы и условия её применения, к которым она должна была адаптироваться, определялись классом специализированных ЭВМ, базовым автокодом и характеристиками программного продукта оборонной системы.

Для формирования подсистемы адаптации было ***исследовано множество характеристик и параметров различных специализированных ЭВМ***. При этом был использован не только статистический материал, накопленный при рассмотрении специализированных ЭВМ, но и априорные ***обобщенные характеристики широкого класса объектных вычислительных машин***. В пределах этого класса определялись типы данных, с которыми работают специализированные ЭВМ, специфика систем команд, их форматы и времена исполнения; состав памяти данных и команд; системы адресации и т. д. Данные характеризовались типом, формой представления, способом кодирования и точностью. В форматах команд учитывались длина, топология и значение полей форматов. Память данных и команд характеризовалась составом, емкостью и способом расположения данных и команд. В классе специализированных ЭВМ фиксировались также способы вычисления контрольных разрядов команды и виды кодограмм, которые определяют формат упаковки готовых программ в канале телекоммуникации, соединяющем технологическую ЭВМ со стендовым оборудованием и специализированной ЭВМ.

2. Требования к системе автоматизации программирования и отладки

К программным комплексам специализированных ЭВМ реального времени ***в 70-е годы*** предъявлялись особенно жесткие требования к использованию их вычислительных ресурсов, что вызывало необходимость построения и применения специфических методов программирования и соответствующих языков для систем реального времени. Для сложных комплексов программ было характерно разнообразие алгоритмов решения функциональных и вспомогательных задач. При этих условиях трудно было удовлетворить требования высокой эффективности получаемых программ и одновременно высокой производительности труда специалистов, если проводить программирование на одном языке. В то же время для обеспечения преемственности алгоритмов, решающих одинаковые задачи в различных системах, было целесообразно унифицировать их описания и структуру модулей программ и данных. Проведенные ***исследования показали, что перспективным является путь автоматизации программирования на базе трех взаимно связанных входных языков***, различающихся глубиной машинной и проблемной ориентации, применяемых в зависимости от типов задач в комплексе программ. По своей ориентации языки программирования ***полезно было разделить*** на:

- ***автокоды,*** предназначенные для записи программ с полным учетом структуры и системы команд конкретной специализированной ЭВМ;
- ***макроязыки,*** предназначенные для записи программ с использованием унифицированных операторов, состав и содержание которых не полностью учитывают структуру каждой специализированной ЭВМ и особенности ее системы команд;
- ***алгоритмические языки,*** запись программ, на которых почти полностью унифицирована и производится с учетом лишь структуры памяти специализированной ЭВМ или класса ЭВМ, либо вообще без учета особенностей конкретной ЭВМ, предназначенной для работы по этой программе.

Для специализированных ЭВМ указанные языки должны были образовывать взаимосвязанную ***систему языков программирования,*** в которой обеспечивался преемственный ***вертикальный переход*** от нижних уровней на более высокий уровень путем ввода новых конструкций в язык и, наоборот, исключением конструкций, не соответствующих задачам уровня. Разработчик функциональной программы или модуля специализированной ЭВМ мог выбирать уровень языка в зависимости от требований к эффективности использования ресурсов памяти и производительности ЭВМ. Удовлетворение этих требований зависело от характеристик трансляторов с того или иного языка. Было исследовано расширение текста транслированных программ при программировании на языках высокого уровня. Для специализированных ЭВМ по отношению к программе, записанной на автокоде, трансляторы ***с макроязыка давали расширения не более 10%, а с алгоритмического языка (модификация и русификация Алгола-60) около 20 – 30%***.

Для обобщенной записи системы команд специализированных ЭВМ, определенных в классе этих машин был разработан ***базовый автокод***. От традиционных автокодов он отличался унифицированным построением синтаксиса и лексики на базе русского языка. Универсальность обобщенного формата в рамках класса специализированных ЭВМ обеспечивала инвариантность синтаксиса базового автокода к изменению этих машин. Лексика базового автокода по составу терминов разделялась на две части: постоянную и переменную. Переменная часть лексики формировалась в процессе идентификации кодов операций, устройств, регистров и других программно-доступных ресурсов специализированной ЭВМ.

Описание базового автокода представляло собой унифицированный документ, который использовался при программировании любой специализированной ЭВМ. В описание базового автокода введен специальный раздел, содержащий правила формирования представления автокода конкретной специализированной ЭВМ, которые регламентировал действия по обозначению машинно-зависимых элементов автокода. Все ***функции транслятора с автокода***, кроме генерации машинных команд, следовало

адаптировать к явным значениям параметров специализированной ЭВМ, автокода и характеристикам проекта программ оборонной системы. Программы, реализующие перечисленные функции, следовало проектировать по методу параметризации.

В большинстве проектов ***макрогенератор*** не зависел от параметров условий применения. Однако в специальных проектах в него могли быть встроены машинно-зависимые функции распределения памяти и масштабирования. В этом случае подготовка макрогенератора осуществлялась аналогично подготовке компилятора алгоритмического языка.

Компилятор алгоритмического языка необходимо было настраивать на структуру памяти и форматы специализированной ЭВМ, а также на правила программирования операторов алгоритмического языка, учитывающие специфику системы команд специализированной ЭВМ. Сложность семантики не позволяла применять для проектирования компилятора принцип параметризации. Поэтому при подготовке компилятора алгоритмического языка, допускалось частичное перепрограммирование семантик. Кроме формирования последовательности команд, при подготовке компилятора алгоритмического языка решались дополнительные задачи, связанные с распределением регистров, масштабированием, контролем точности, выполняемыми машинно-зависимыми процедурами.

Характеристики проекта комплекса программ служили для описания функционального распределения ресурсов специализированной ЭВМ под различные компоненты комплекса программ. В этом описании должны были закрепляться соглашения о связях, т. е. указываться номера регистров, которые отводятся для хранения адресов передачи управления от вызывающей программы к вызываемой и адресов возврата; адресов начала зон переменных; для распределения областей памяти под глобальные и локальные переменные.

Адаптируемость интерпретатора для отладки программ специализированной ЭВМ ***с исполнением на универсальной ЭВМ***, при выбранных показателях качества по производительности, обеспечивалась совместным применением трех принципов проектирования адаптируемых программ. Для параметризованной компоненты интерпретатора в базу данных для настройки включались информационные модули, содержащие всю информацию о структуре специализированной ЭВМ, а именно: таблицы регистров и триггеров; аппаратных прерываний; описания системы команд, т.е. кодов операций, типов операндов, времени исполнения команд, слова состояния процессора. Конфигурируемую компоненту интерпретатора, которую составляют процедуры, моделирующие арифметические, логические и управляющие команды, можно было записывать на макроязыке. В результате проведенных исследований были сформулированы основные ***требования к системе автоматизации программирования и отладки***,

которые практически полностью были реализованы в середине ***70-х годов*** в системе ЯУЗА-6, построенной на технологической ЭВМ БЭСМ-6.

Требования к САРПО ЯУЗА-6 для специализированных ЭВМ зависели от особенностей этих машин, а также от функциональных задач ряда оборонных систем и ***состояли в следующем*** [1, 2].

Единство системы автоматизации программирования и отладки комплексов программ должно было обеспечивать разработку высококачественных программных продуктов при минимальных затратах совокупного труда специалистов на их создание. Система должна была иметь средства общения человека со средствами автоматизации на языках задания и программирования, построенных по единым правилам. Вследствие ***ограниченных ресурсов специализированных ЭВМ***, технологические задачи автоматизации программирования и отладки программ следовало реализовать на стационарной, универсальной ЭВМ, сохраняя на специализированных машинах только минимум обслуживающих программных средств, необходимых для комплексной отладки в реальном времени.

Высокая эффективность получаемых программ по использованию памяти и производительности мобильных ЭВМ определялась стабильностью решаемых задач, перечень и объем которых обычно устанавливался на ранних стадиях разработки оборонной системы управления. Экономичность программ должна была обеспечиваться не только по количеству машинных команд, но и по объему оперативной памяти, используемой для хранения переменных и констант, что приводило к необходимости работы с величинами, занимающими часть слова, с масштабированными переменными и к плотной упаковке величин в одной ячейке памяти.

Многоязыковый вход в систему автоматизации представлял перспективный путь эффективного использования ресурсов мобильных ЭВМ при программировании для управляющих систем на базе нескольких взаимосвязанных входных языков, различающихся глубиной машинной и проблемной ориентировки, применяемых в зависимости от типов описываемых задач. По степени машинной ориентации весь ***комплекс программ управляющей системы*** целесообразно было делить на следующие ***три группы***:

- программы обмена информацией и машинные тесты;
- программы организации вычислительного процесса, управления задачами и данными (операционные системы реального времени);
- программы решения функциональных задач конкретной оборонной системы.

Наибольшей проблемной и наименьшей машинной ориентировкой обладают функциональные задачи оборонной управляющей системы. Программы этих задач имеют обычно наибольший удельный вес в комплексе программ

системы, составляя 70 – 80% общего размера программ. Их разработку целесообразно было производить на проблемно-ориентированном алгоритмическом языке высокого уровня (в данном случае на модификации языка Алгол-60).

Программы организации вычислительного процесса, управления задачами обмена информацией и тесты весьма сильно связаны с архитектурой вычислительной машины и с конкретной задачей управления вычислениями в разрабатываемой системе. Эти обстоятельства снижали эффективность использования алгоритмического языка высокого уровня и его применяемость для записи алгоритмов в различных системах. Для разработки этих программ целесообразно было иметь машинно-ориентированный язык типа автокода с макрокомандами.

Адаптация системы автоматизации программирования и отладки на логику систем команд специализированных ЭВМ должна была быть унифицирована и сделана независимой от типа и системы команд ЭВМ. Вся специфика, обусловленная системой команд конкретной ЭВМ, была вынесена в сменные или настраиваемые программные и информационные модули. Кроме состава, структуры и логики исполнения команд объектной ЭВМ, реализация заменяемых элементов зависит от методов адресации, объемов и структуры памяти различных типов, состава и типов внешних устройств, характеристик и способов обмена с внешними устройствами и абонентами. Настройка должна была осуществляться автоматизировано по формальному описанию логики команд и структуры мобильной ЭВМ, представленному на специальном языке.

Сменная (машинно-ориентированная) часть системы автоматизации программирования и отладки должна была обеспечивать, с одной стороны, машинную ориентацию процесса трансляции с каждого из входных языков на язык системы команд конкретной специализированной ЭВМ, а с другой – позволять исполнять программы этой ЭВМ на универсальной машине при отладке программ по тестам. При этом входные языки программирования, методы и правила структурного построения управляющих программ, язык отладки, состав и правила оформления документации на алгоритмы и программы должны быть едиными для всех систем управления. Единая технологическая база автоматизации должна в значительной степени обеспечивать инвариантность алгоритмических и программных разработок относительно изменения управляющих, специализированных вычислительных машин.

Должно было ***обеспечиваться автоматизированное сопряжение*** отдельных модулей и компонентов в большие комплексы управляющих программ. Для управляющих программ характерно мультипрограммное функционирование комплексов взаимодействующих функциональных компонентов объемом в десятки и сотни тысяч команд, использующих общую память глобальных переменных. ***Модульное, иерархическое построение*** должно было позволять

получать обозримое описание решаемых задач и их взаимосвязей в сложном комплексе программ, а также существенно облегчать автоматизацию всех этапов процесса разработки и в первую очередь контроля компонентов и их сопряжения в единый комплекс. Достаточно эффективным средством обеспечения сопряжения подпрограмм являлось использование ***единого описания глобальных переменных***, с применением которого транслируются или загружаются все функциональные компоненты программы системы.

Должен осуществляться ***автоматизированный расчет длительности исполнения*** функциональных компонентов и всего комплекса программ в реальном времени. Функционирование управляющих систем в реальном времени связано с возможностью реализовать исполнение комплекса всех необходимых программ в заданные интервалы времени. На конечных этапах разработки, когда функционирует вся управляющая система или основные функциональные задачи, длительность исполнения программ может быть получена непосредственным измерением времени исполнения каждого компонента программы и всего комплекса в процессе решения задач в реальном времени.

На всех этапах разработки, значения длительности реализации компонентов и всего комплекса программ, могут быть получены аналитически, последовательным расчетом по блок-схемам алгоритмов или по готовым программам. Длительность реализации каждого линейного участка подпрограммы следовало рассчитывать непосредственно по операциям в программе или по операторам в блок-схеме. Вероятности условных переходов и количества прохождений циклов разработчик мог задать на основе представлений о функционировании программы в средних и в предельных условиях. При таких исходных данных аналитически должна была рассчитываться длительность реализации каждой программы.

Должен был производиться ***автоматизированный контроль корректности*** текстов, структуры компонентов программ и сопряжения всего комплекса программ. Во всех задачах контроля следовало иметь эталон – систему правил, которым должна соответствовать программа в целом и каждая ее часть. Несоответствие эталонным правилам надо регистрировать одной из систем контроля с указанием места нарушения правил (ошибки). Задача синтаксического контроля – проверка входного текста программы на соответствие формальному описанию синтаксиса языка программирования. Задачей семантического контроля должен быть анализ текстов программ на правильность применения конструкций входного языка для записи алгоритмов и соответствия правилам взаимосвязи элементов конструкций, главным образом между операторами и описаниями глобальных данных.

Структурный контроль построения компонентов должен был выявлять некоторые виды зацикливания, наличие тупиковых и лишних участков в алгоритмах и другие нарушения правил построения структуры компонентов программ. Структурный контроль взаимодействия функциональных

подпрограмм в комплексе программ в принципе подобен контролю компонентов, однако охватывает значительно больший объем программ, и каждая подпрограмма при этом участвует как единое целое. При этом должны контролироваться сопряжение подпрограмм по локальной информации, по управлению, и использованию глобальных переменных в динамике их записи и чтения.

Автоматизированная отладка автономных модулей и подпрограмм на уровне входного языка предназначена для локализации ошибок в системе, путем контроля процесса вычислений при заданном тесте с различной степенью подробности. Они должны были обеспечивать:

- обработку задания на отладку, представленного на языке, близком по уровню к языку программирования;
- трансляцию значений переменных и констант в составе теста в величины и масштабы данных, используемых в специализированной ЭВМ;
- исполнение задания на отладку при заданных исходных тестах;
- информирование разработчика о результатах исполнения отладочного задания.

Значительная часть отладки при программировании на алгоритмических языках, должна была предварительно проводиться с использованием метода компиляции, т. е. по программам, представленным в командах технологической ЭВМ. Это позволяло существенно повысить скорость исполнения программ и снизить общую трудоемкость отладки. Исполнение отладочных заданий на заключительных стадиях отладки должно было производиться ***методом интерпретации*** на уровне команд специализированной ЭВМ. Введение различных уровней детализации в системе отладки логики исполнения программ и изменения переменных должно было позволять селектировать необходимую для отладки информацию, в процессе исполнения задания и избавлять разработчика программы от избыточной информации. Тем самым каждая операция по локализации ошибки должна была содержать всю необходимую информацию.

Комплексная отладка систем управляющих программ реального времени должна была учитывать, что существует противоречие между стремлением обеспечить функционирование комплекса управляющих программ в условиях, максимально близких к реальным, и ограниченными возможностями управляющих ЭВМ для размещения и исполнения технологических программ, обслуживающих отладку. Кроме того, специализированные ЭВМ весьма ограничены по составу внешних устройств, необходимых для общения человека с машиной в процессе отладки комплекса программ в реальном времени.

Для обеспечения решения задач управления в реальном времени при одновременном решении задач: имитации информации внешней среды, накопления и статистической обработки результатов, информирования операторов о ходе процесса исполнения программ и других функций, обслуживающих отладку, следует использовать технологическую ЭВМ. В этом случае на управляющую ЭВМ дополнительно к ее основным функциональным задачам должно возлагаться исполнение операторов отладочного задания и выдача в технологическую ЭВМ результатов их реализации, которые осуществляются при весьма малых затратах памяти и производительности управляющей ЭВМ. Кроме того, это способствует единству методов и языковых средств в течение всего процесса разработки управляющих программ и использования средств автоматизации разработки программ без применения трудоемких ручных методов при переходе от одного этапа к другому.

Автоматизированный выпуск и корректировка технической документации на комплекс программ необходимы, для того чтобы успешно разработать, отладить и ввести в эксплуатацию сложный программный продукт. Документация на комплексы управляющих программ должна была строиться по иерархическому принципу и состоять из нескольких уровней детализации. Такая структура должна позволять в удобной и наглядной форме проводить анализ программ как от общего к частному, так и от частного к общему. Наибольший объем документации соответствует представлению комплекса программ в детальном виде на уровне текстов модулей и компонентов программ на разных языках и описаний переменных и констант. Значительный объем документов представляют также блок-схемы с расшифровкой операторов, блоков и подпрограмм. Вся необходимая исходная информация для этой части документов должна быть в архивах и библиотеках системы автоматизации программирования и отладки на технологической ЭВМ.

Вторая часть документов должна представлять комплекс программ в обобщенном и укрупненном виде, описывать общие принципы их функционирования и иерархическую схему связей компонентов. Эта документация не может быть изготовлена автоматически, так как для обобщений необходимо творческое участие разработчиков, и укрупненная исходная информация отсутствует в технологической ЭВМ. Положение упрощалось тем, что объем такой документации относительно невелик и находится в пределах 10 – 20% общего объема.

Кроме того, должен был обеспечен оперативный автоматический выпуск документов, отражающих изменения в программах, переменных и константах с корректировкой всей документации на завершающих этапах комплексной отладки, а также в процессе эксплуатации и при модернизации системы. Автоматический выпуск основной документации на программы позволяет исключать множество ошибок, как в документах, так и непосредственно в

управляющих программах, и существенно влияет на качество и эксплуатационные характеристики программного продукта.

Автоматизированный контроль процесса разработки и технологических характеристик комплекса управляющих программ необходим руководителям проекта для учета объективных характеристик и тенденций изменения состояния комплекса. В зависимости от степени детализации и аспекта анализа, эти характеристики должны служить основой для принятия различных решений по корректировке программ, технологии их разработки и распределению усилий специалистов с целью повышения качества проектирования и снижения его трудоемкости. Исходные данные для получения большинства частных и обобщенных характеристик содержатся в архивах и библиотеках системы автоматизации программирования и отладки и могут быть выделены, упорядочены и отредактированы. Однако ряд величин является промежуточными, не регистрируются и теряются, если не предусмотрены специальные программные средства для их фиксирования в процессе выполнения функциональных задач по программированию и отладке.

3. Структура системы автоматизации программирования и отладки ЯУЗА-6.

Система ЯУЗА-6 делилась на ***три крупных компонента*** (рис. 1) [2]:

- организующую систему;
- систему автоматизации программирования;
- систему автоматизации отладки.

Эти три системы использовали развитую информационную систему – ***базу данных***, которая включала архив символьной информации исходных текстов программ, библиотеку паспортов модулей и компонентов, архив оттранслированных программ на языке программирования и в машинных кодах, тесты и результаты отладки, технологические и эксплуатационные документы программных продуктов.

Организующая система была предназначена для формирования режимов функционирования САРПО, для управления хранением и обработкой данных, для подготовки системы к эксплуатации, а также для сбора и подготовки информации контроля процесса разработки. В соответствии с назначением она состояла из четырех частных систем.

Монитор служил для связи пользователей с системой ЯУЗА-6 и являлся управляющей системой, обеспечивающей общую организацию прохождения работ, выполняемых САРПО ЯУЗА-6 на технологической ЭВМ БЭСМ-6 с ОС ДИСПАК. Монитор обеспечивал ввод и контроль информации, и вызов отдельных компонентов системы ЯУЗА-6.

Система управления данными проекта обеспечивала запись, корректировку и считывание, распечатку и каталогизацию входной символьной информации архивов САРПО, организованных на магнитных лентах или дисках БЭСМ-6.

Система автоматизации адаптации обеспечивала выполнение комплекса процедур, формировавших версию САРПО ЯУЗА-6 для конкретной специализированной, управляющей ЭВМ и конкретного проекта автоматизированной системы. Для адаптации пользователь использовал, так называемую ***базовую систему,*** представлявшую собой совокупность машинно-независимых программ и блоков САРПО ЯУЗА-6 (95% всей системы). Машинно-зависимая часть САРПО была сосредоточена в ограниченном числе машинно-зависимых программных модулях.

Система контроля разработки обеспечивала сбор, обобщение и редактирование информации о текущем состоянии разработки комплекса программ проекта и об их характеристиках. Эта информация являлась исходной для руководителей при контроле и управлении ходом разработки комплекса программ.

Система автоматизации программирования обеспечивала получение синтаксически, семантически и структурно-корректных записей компонентов и комплекса программ и описаний переменных, на входных языках САРПО ЯУЗА-6 и в машинных кодах программ, а также технической документации на них и временных характеристик исполнения.

Транслятор описаний глобальных переменных. В системе ЯУЗА-6 составление описания глобальных переменных, которыми пользовались (без дополнительных описаний) все или часть программных компонентов, было выделено в отдельный этап. Описания глобальных переменных и/или глобальные константы были объединены в структуры в различных формах представления, принятых для специализированной ЭВМ. Язык описаний глобальных переменных являлся фрагментом алгоритмического языка ЯУЗА. Транслятор описаний состоял из двух просмотров и обработки входного текста, предварительно прошедшего лексический контроль, и перекодированного во внутренний код транслятора.

ЯУЗА-6 имел ***три взаимосвязанных языка программирования***: автокод, макроязык и алгоритмический язык. В качестве базового языка был использован модифицированный и русифицированный Алгол-60, дополненный средствами раздельной компиляции, возможностями указания типов данных с ограничениями и конкретным представлением их в памяти ЭВМ.

Трансляторы с алгоритмических языков для специализированных ЭВМ состояли из пяти просмотров и решали следующие задачи:

- лексический и синтаксический контроль входного текста;
- распределение памяти переменных;

- семантический контроль;
- масштабирование;
- оптимизация программы;
- формирование последовательности автокодных команд или операторов макроязыка.

Транслятор с макроязыка строился по обычной схеме макрогенераторов. При этом макроязык рассматривался как макросредства над автокодом.

Транслятор с автокода для определенной специализированной ЭВМ отличался выбором при адаптации изобразительных средств (буквенных и символьных) для составляющих элементов команд. Глобальные переменные в модуле программы не описывались, они считались объявленными во всем комплексе программ. Трансляция с автокода состояла из двух просмотров.

Система структурного контроля осуществляла контроль построения компонентов на соответствие общим правилам, предъявляемым к структуре подпрограмм и вложенных модулей, а также производила контроль размещения переменных в памяти специализированной, управляющей ЭВМ. Система подразделялась на две подсистемы: построения модели программы (операторной схемы) и собственно контроля структуры на отсутствие зацикливаний, тупиков и неподключенных участков в компонентах программы.

Система расчета временных характеристик позволяла получать средние и предельные значения длительностей исполнения отдельных компонентов и комплекса взаимосвязанных программ без их непосредственного исполнения. Объектом анализа для системы являлась взвешенная графовая модель программы, автоматически построенная по модели программы и приведенная к эквивалентному ациклическому виду.

Система выпуска документации была предназначена для выпуска технологической и эксплуатационной документации по информации, хранящейся в базе данных системы ЯУЗА-6. Такими документами являлись: распределение памяти ОЗУ и ДЗУ; паспорта программ; макроописания; каталоги библиотек и характеристики проекта. Все перечисленные документы печатались на стандартных бланках.

Система автоматизации отладки (система детерминированного тестирования) была предназначена для исполнения программ управляющей ЭВМ на технологической машине БЭСМ-6 с использованием программного интерпретатора, который моделировал работу специализированной ЭВМ. Система ***состояла из трех компонентов***.

Транслятор отладочных заданий и тестов производил обработку отладочного задания на программу, представляющую собой последовательность операторов отладки и тесты.

Интерпретатор команд специализированной ЭВМ осуществлял моделирование покомандного исполнения программы на БЭСМ-6 в кодах специализированной ЭВМ в условном масштабе времени. Память специализированной ЭВМ (ОЗУ и ДЗУ) отображалась однозначно на память БЭСМ-6, загрузка которой всеми программами и константами отлаживаемого комплекса программ производилась из архива САРПО. В результате формировалась информация о типе и адресе исполненной команды, адресе команды, которая будет исполняться следующей, и содержании всех моделируемых регистров управляющей ЭВМ.

Информирующая система отладки была предназначена для получения и выдачи на экран и/или на печать информации о результатах исполнения отлаживаемых программ. Система опиралась на план отладки, в котором были зафиксированы данные, интересующие пользователя, и на результаты моделирования исполнения каждой команды, выданные интерпретатором. Используя паспорта программ, производился перевод адресов в символьное представление имен величин, а содержимое ячеек (регистров) переводилось с учетом типов и масштабов величин в значения в терминах входного языка.

4. Заключение

При применении ЯУЗА-6 был налажен и апробирован ***процесс накопления и использования наборов готовых испытанных программных и информационных модулей и компонентов*** для формирования новых комплексов программ и/или их модернизированных версий. Библиотеки наборов таких компонентов и унифицированные межмодульные интерфейсы обеспечили ***эффективное конфигурационное управление*** при создании из них новых комплексов программ и при модернизации эксплуатируемых версий программных продуктов.

Первая версия **САРПО была** ***передана для эксплуатации разработчикам в середине 1975-го года***, адаптированной на ***ЭВМ 5Э26*** для разработки комплекса программ радиолокационного узла «Основа», а затем системы ПВО «Байкал». Эта массовая высокопроизводительная ЭВМ для мобильных оборонных систем была создана в ИТМ и ВТ [1]. Было произведено около 1,5 тысячи этих машин, с программными комплексами, разработанными с использованием САРПО ЯУЗА-6. Для обеспечения работ по отладке системных и пользовательских ***программ в реальном времени*** впоследствие был создан имитационный комплекс на БЭСМ-6 в многомашинном комплексе АС-6.

Полная версия САРПО ЯУЗА-6 эксплуатировалась в ряде организаций почти 20 лет с 1979 года (в одной организации до 2000 года) и имела суммарный объем около 400 тыс. слов БЭСМ-6. В разработке этой версии системы принимали участие около 60 специалистов при средней производительности труда около 5 команд в день на человека. Процессы разработки компонентов системы производились сверху вниз по техническим

заданиям и спецификациям требований небольшими группами специалистов на языке БЕМШ. Регистрировались затраты на различных этапах и была произведена оценка общей трудоемкости разработки САРПО ЯУЗА-6, составившей около 300 человеко-лет. Для системы было выпущено техническое описание, инструкции по адаптации и эксплуатации.

Наиболее активно САРПО ЯУЗА-6 применялась для различных типов бортовых ЭВМ класса «Аргон», а также в следующих организациях, где была адаптирована для приведенного количества типов специализированных ЭВМ:

МНИИ приборной автоматики – 6 типов;

НИИ автоматической аппаратуры – 5 типов;

НПО автоматизации приборостроения – 11 типов;

НИ электромашиностроительный институт – 2 типа;

НПО ЭЛАС (г. Зеленоград) – 3 типа;

НПО им. Лавочкина – 2 типа;

НПО ПРОГРЕСС (г. Самара) и другие.

В сумме это определило использование ЯУЗА-6 более чем в 13 организациях, для свыше 30 типов мобильных, специализированных ЭВМ. Общий ***объем разработанных программ с применением ЯУЗА-6 к 1985 году превысил 5 млн. команд***. В НПО АП ЯУЗА-6 использовалась, в частности, для разработки программ: орбитальной станции САЛЮТ-7; межпланетных станций Венера и Марс, спутников Экран, Радуга, Горизонт и ряда стратегических ракет. Эксплуатация ЯУЗА-6 прекращалась в конце ***90-е годов***, в основном, вследствие технического старения машин БЭСМ-6.

Литература

[1] Липаев В.В. Отечественная программная инженерия: фрагменты истории и проблемы. – М.: СИНТЕГ. 2007. 312 с.

[2] Липаев В.В., Серебровский Л.А., Филиппович В.В. Система автоматизации программирования и отладки комплексов программ управления (ЯУЗА-6). «Программирование», 1977, № 3.