Е. М. Лаврищева

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ И ПРОГРАММНАЯ ИНЖЕНЕРИЯ

УЧЕБНИК ДЛЯ ВУЗОВ

Допущено Учебно–методическим объединением высших учебных заведений РФ по образованию в области прикладных математики и физики в качестве учебного пособия для студентов вузов, обучающихся по направлению «Прикладные математика и физика», а также по другим математическим и естественнонаучным направлениям

Москва ▪ Юрайт ▪ 2017

УДК 004(075.8)
ББК 32.973я73
Л13

***Авторы*:**

Лаврищева Екатерина Михайловна — доктор физико-математических наук, профессор Московского физико-технического института (государственного университета), главный научный сотрудник Института системного программирования РАН.

***Рецензенты*:**

Петренко А. К. — профессор, доктор физико-математических наук, заведующий отделом технологий программирования Института системного программирования РАН;

Петров И. Б. — доктор физико-математических наук, профессор, заведующий кафедрой информатики и вычислительной математики Московского физико-технического института (государственного университета), член-корреспондент РАН;

Провотарь А. И. — профессор, доктор физико-математических наук, заведующий кафедрой информационных систем факультета кибернетики Киевского национального университета имени Тараса Шевченко.

Лаврищева, Е. М.

Л13 Технология программирования и программная инженерия : учебник для вузов / Е. М. Лаврищева. — М. : Издательство Юрайт, 2017. — 432 с. — Серия : Университеты России.

ISBN 978-5-9916-8275-6

Серия «Университеты России» позволит высшим учебным заведениям нашей страны использовать в образовательном процессе учебники и учебные пособия по различным дисциплинам, подготовленные преподавателями лучших университетов России и впервые опубликованные в издательствах университетов. Все представленные в этой серии учебники прошли экспертную оценку учебно-методического отдела издательства и публикуются в оригинальной редакции.

Описаны основные положения технологии программирования и инженерии программных продуктов и систем. Изложены отечественные и зарубежные методы разработки сложных программных систем из готовых компонентов повторного использования (КПИ). Рассмотрены подходы к инженерии программных продуктов из КПИ и их вариантов в SPLE (Software Product Line/Product Family), GDM (Generative Programming), Grid и др. Изложен метод сборки программных систем из КПИ с учетом модели характеристик (Feature Model), изменяющей структуру программных продуктов и систем. Описан объектно-компонентный метод моделирования вариантов программных систем и семейств программных систем из КПИ. Излагаются онтология представления доменов, методы извлечения знаний о готовых системах, спецификациях КПИ и доказательстве систем из них. Рассмотрены методы экспертизы, верификации, тестирования и оценки качества систем. Описаны дисциплины SE и методы создания веб-систем в среде Semantic Web и требования стандарта SEMAT к применению теории и методов SE в разработке систем.

Для разработчиков и специалистов, которые интересуются современными технологиями изготовления отдельных программ и КПИ в разных языках программирования, их сборкой для получения вариантов программных систем и продуктов с обеспечением качества, а также студентов, магистров и специалистов по направлению подготовки «Теория и технология программирования».

УДК 004(075.8)
ББК 32.973я73

ISBN 978-5-9916-8275-6

Оглавление

Список принятых сокращений

АСУ — автоматизированная система управления
БД — база данных
ГоР — готовые ресурсы
ЖЦ — жизненный цикл
ИС — информационная система
КП — комплексы программ
КПИ — компоненты повторного использования
МХ — модель характеристик
ОКМ — объектно-компонентный метод
ООП — объектно ориентированные программы
ОС — операционная система
ПИ — программная инженерия
ПО — программное обеспечение
ПП — программный продукт
ППП — пакет прикладных программ
ППр — программирующая программа
ПрО — предметная область
ПС — программная система
СП — система программирования
СПС — семейство программных систем
ТД — тип данных
ТЛ — технологическая линия
ТП — технология программирования
ТПР — технология подготовки разработки
ЭВМ — электронно-вычислительная машина
ЯП — язык программирования
ACM — Association for Computing Machinery
CS — Computere Science
DE — Domain Engineering
DSL — Domain Specific Language
FM — Feature Model
GDM — Generative Domain Model
PIM — Platform Independed Model
PLE — Product Line Engineering
PSM — Platform System Model
SE — Software Engineering
SWEBOC — Software Engineering Body Knowledge

От автора

Теоретическими и прикладными аспектами автоматизации создания разных систем обработки информации на отечественных ЭВМ и на современных компьютерах автор занимается с самого начала своей научной деятельности. Исследования и разработки систем относятся к ТП и ПИ.

В период появления ЭВМ и разработки систем программирования для них в отделе Е. Л. Ющенко с 1960 г. проводились семинары по изучению синтаксического анализа ЯП в работах Н. Хомского, К. Самельсона, Ф. Бауэр, Р. Флойд и др. Стояла задача создать системы программирования для новых ЭВМ («Проминь», «М-20», «Днепр-1», «Днепр-2» и др.). В результате сформировался метод синтаксического контроля программ в разных ЯП. Метод реализован схематически в робототехнологических комплексах и системах программирования с языков Алгол, Кобол, Фортран, Автокод и др. Аналогичные разработки проводились в стране для других машин («МЭСМ», «БЭСМ», «Стрела», «Урал», «М-50» и др.). Созданы отечественные трансляторы ТА1—ТА4, которые вошли в состав системного математического обеспечения первых ЭВМ. Трансляторы с языков Алгамс и Кобол реализованы на машине «Днепр-2» и внедрены в ГДР в 1970-е гг. На их основе совместно со специалистами из Киева реализована АСУ ТП металлургической промышленности ГДР, которая проработала до 1992 г.

В 1970-х гг. в стране развернулись работы по автоматизации разного рода систем из программ (АПРОП, ПРОЕКТ, АЛЬФА и др.). Автор реализовала метод сборки модулей, записанных в разных ЯП (Алгол-60, Кобол, Фортран, PL/1, Assebler и др.), на единой системе ЭВМ. Подобные работы проводились специалистами в Москве, Ленинграде, Таллине и др. и в США. В АПРОП впервые реализована связь модулей в разных ЯП с помощью интерфейса (межмодульного и межъязыкового), осуществляющего обмен данными и их преобразование посредством интерфейсных функций (64) библиотеки в ОС ЕС. Система вошла в состав комплекса ПРОТВА и передана 52 организациям ВПК, а также в организации Прибалтики и Средней Азии. В результате сформировался новый вид программирования — сборочный. В составе специалистов комплекса ПРОТВА по технологии создания бортовых систем автор получила звание лауреата премии Кабинета министров СССР (1985).

Дальнейшим развитием сборочного программирования является метатехнология (ТПР) создания ТЛ производства программ в системе АИС «Юпитер» для Военно-морского флота СССР (1982—1991) в АН УССР. По пяти созданным линиям было изготовлено в АИС более 500 научных программ и программ обработки данных. Среди программ были такие, которые обеспечивали надежность технических и программных средств

АИС. Система передана заказчику. Идея линий получила развитие в проекте института SE USA (2001, http://www.sei.com/productline) и положила начало созданию вариабельных (изменяемых) продуктов семейств СПП (2004). Эти работы развиваются и в нашей стране рамках проекта Российского фонда фундаментальних исследований (РФФИ).

А. П. Ершов предвосхитил развитие сборочного программирования как доказательного (1986). В рамках фундаментальных проектов в отделе «Программная инженерия» Национальной академии наук Украины проведено изучение и развитие теории генерирующего (К. Чернецки), объектного (Г. Буч) и компонентного программирования (1992—2013). Создана теория объектно-компонентного моделирования объектных систем, которая апробирована в проекте информатизации НАН Украины. Разработаны новые идеи и концепции верификации, тестирования и оценивания качества объектов и ПС. По ним защищено пять диссертаций сотрудников отдела и шесть магистерских работ Московского физико-технического института (МФТИ) и Киевского национального университета имени Тараса Шевченко (КНУ). Дальнейшее развитие этого метода состоит в создании разных вариантов систем и продуктов, в проведении стандартизации GoP и reuses — КПИ в языке WSDL, в исследовании основных положений веб-семантики, веб-сервисов и онтологического представления ряда предметных областей (ЖЦ ПС, КПИ и вычислительная геометрия) с участием студентов КНУ, где автор преподавала ТП и ПИ с 1967 г., а также в МФТИ (2001—2016).

Современные ИТ-технологии Всемирной паутины и системы Skype ставят задачу по обеспечению взаимодействия программ, расположенных в разных точках планеты. В перспективе в информационном сообществе разрабатываются научные сервисы для поддержки *e*-коммерции, *e*-науки, *e*-обучения и т.п.

Развитие технических ресурсов ЭВМ идет быстрыми темпами по пути совершенствования и стандартизации микроэлементов (диоды, триоды, транзисторы и др.). Методом сборки из них изготавливают большие и малые компьютеры, очень малые настольные и карманные, а также мобильные телефоны и приборы для применения в медицине, космосе, авиации и т.п.

Такого прогресса еще не достигнуто в области изготовления сложных программных и информационных систем. ТП и ПИ находятся на стадии формирования готовых многоразовых ресурсов типа модуль, объект, компонент для конфигурационной сборки сложных систем. Эти объекты будут минимизироваться подобно техническим наноэлементам, и из них будут собираться большие системы в миниатюрном виде.

В данном учебнике нашли отражение основные положения ТП, сформировавшейся в СССР, и ПИ изготовления ПО и продуктов в разных вариантах. В нем более углубленно и с теоретической точки зрения рассматриваются методы проектирования отдельных компонентов систем, их верификация и тестирование, а также сборки готовых КПИ, GoP в сложные структуры с обеспечением их правильности и качества ПП в конфигурационные варианты систем семейства. Проблема изменяемости/вариабельно-

сти нашла отражение в отечественной технологии объектно-компонентного моделирования вариантов сложных систем из артефактов, reuses и КПИ.

Отдельные разделы учебника апробированы автором при чтении лекций в МФТИ (2001—2016) и Н. В. Лаврищевой в КНУ. Новые теоретические результаты в SE отображены в ряде дипломных работ студентов на сайте (http://www.sestudy.ua-univ.net, 2011), студенческой фабрики программ КНУ (http://www.programsfactory.univ.kiev.ua, 2012) и в статьях и докладах учеников автора на международных конференциях ICTERY (2011—2013), УкрПрог (1998—2014), TAAPSD'2008—2015, Science and Information — 2015, АПСПИ — 2015 и др.

На практических занятиях студенты изучают линии Product Line/ Product Family, Streaming assembly and conveyor К. Чернецки, Дж. Гринфилд, К. Ленц и др. С участием студентов и сотрудников отдела «Программная инженерия» реализована веб-онтология таких доменов, как вычислительная геометрия, КПИ и ЖЦ стандарта ISO/IEC 12207. Разработаны методы экспертирования, верификации и тестирования систем, предложен математический аппарат оценки характеристик качества продуктов. Ряд результатов исследований и разработок отражен на модифицированном сайте http://www.ispras.ru/lavrischeva/SE.

Автор глубоко благодарна сотрудникам отдела (В. Н. Грищенко, Н. В. Лаврищева, Л. Г. Усенко, Е. И. Гришкевич, Т. В. Соколова, М. В. Семенюк, А. Т. Вишня, Д. С. Хоролец, Г. И. Коваль, Т. М. Коротун, Л. П. Бабенко, О. А. Слабоспитская, Н. Т. Задорожная, В. М. Зинькович, Е. И. Моренцов, Л. И. Куцаченко и др.), которые долгие годы развивали и реализовывали отдельные новые идеи в области ТП и ПИ. Семь сотрудников защитили кандидатские диссертации.

Выражаю признательность многим студентам (А. Островский, А. Аронов, А. Диденко, И. Радецкий, В. Бураков, В. Черный, Е. Гриценко, Т. Гончаренко, А. Поляченко, А. Колесник, А. Мороз, А. Бондаренко, М. Тарановский, Е. Баборико и др.), которые принимали активное участие в реализации поставленнях мною задач в области ТП и SE, писали по ним дипломы и магистерские работы. Особенно хочу поблагодарить студентов А. Колесник, А. Стеняшина, А. Шевченко, которые защитили кандидатские диссертации по современным вопросам создания изменяемых систем.

Хочу также поблагодарить В. П. Иванникову, А. К. Петренко, А. Н. Томилину, Л. Е. Карпову за признание моих работ в области ТП и ПИ, поддержавших проведение исследований и разработок вариабельных прикладных и операционных систем в рамках фундаментального проекта РФФИ.

Предисловие

С учетом направлений ТП, SE и SEMAT (SE Methods and Theory) в учебнике излагаются результаты теоретических исследований и разработок отечественных и зарубежных специалистов в области ТП и инженерии ПП и семейств СПС.

В области ТП сформировались сборочное программирование, метод моделирования объектных систем из КПИ, а также экспертирование, тестирование и оценивание качества ПС. Определен подход к обеспечению взаимодействия GoP между собой в среде Интернет. Представлена теория типизации данных (фундаментальных, общих GDT — стандарт ISO/IEC 11404–2007) и предложены новые механизмы обработки неструктурированных данных (больших данных) типа Big Data. Описаны методы онтологии ЖЦ, КПИ и вычислительной геометрии.

При освоении данного учебника обучающийся должен:

знать

- теорию модульного программирования систем из разнородных языковых модулей, интегрируемых вместе с помощью интерфейса;
- теоретические основы и методы проектирования вариабельных систем на основе модели характеристик, моделей систем и модели конфигурации;
- зарубежные методы формальной спецификации объектов предметной области, верификации и доказательства с использованием методов Флойда, Дейкстры, Турского и др.;
- теорию объектно-компонентного моделирования предметных областей с применением математических и логических операций;
- теорию формальных и общих ТД (FDT, GDT) и некоторые новые механизмы генерации неструктурированных больших данных типа Big Data структур FDT, GDT;
- онтологические основы представления научных и инженерных доменов (ЖЦ стандарта ISO/IEC 12207, домены КПИ и «Вычислительная геометрия»);
- парадигмы программирования (модулей, компонентов, сервисов, объектов, аспектов и др.) к виду отдельных самостоятельных элементов повторного использования КПИ и взаимодействующих между собой через интерфейс, определенный автором в 1970-х гг. и широко используемый во всех общесистемных инструментах (IBM, Microsoft, Intel, Apple и др.) и сегодня в фабриках программ;
- классификацию дисциплин ПИ и характеристику новых дисциплин в Semantic Web;
- новые теории и методы SEMAT;

уметь

- создавать операционные, прикладные бизнес-системы;
- анализировать новые подходы и методы, которые появляются в SEMAT;
- представлять итоги проделанной работы в виде отчетов, рефератов, статей с учетом стандартных требований с привлечением современных средств редактирования и печати;

владеть

- общенаучными методами исследований и творчески применять их при проведении разработки крупных программных проектов;
- методами проектирования современных вариабельных систем, широко используемых за рубежом;
- владеть методами представления данных, анализа и синтеза программной и системной информации, извлекать знания из готовых действующих систем методами Mining для последующего применения в других системах, а также формировать выводы из полученных знаний.

Материал данного учебника состоит из 12 глав. В конце каждой главы для студентов приводятся контрольные вопросы и задания, а также список используемой литературы.

Глава 1. Технология программирования программных систем. Изложены основы технологии программирования в СССР. Рассмотрены системы программирования и трансляторы с первых ЯП (Алгол, Кобол, PL/1, Алгол-68 и др.). Определены сформировавшиеся методы синтеза, сборки и тестирования программ. Представлена модульная ТП больших систем. Дано определение сборочной технологии, обеспечивающей интеграцию и объединение разнородных КПИ и ГоР.

Глава 2. Программная инженерия / Software Engineering программных продуктов. Представлена ПИ изготовления ПО, ПС и ПП средствами стандарта SWEBOK, включающего 10 областей знаний. Дано описание этих областей знаний. Изложены парадигмы программирования, сложившиеся в рамках SE. Рассмотрены модели ЖЦ, оформившиеся в процессе разработки ПО, ПС и ПП. Рассмотрены парадигмы программирования типа Agile.

Глава 3. Новые подходы к разработке изменяемых ПП из КПИ. Изложены методы производства ПП на Product Lines США с использованием КПИ и ГоР. Определена модель изменяемости (вариабельности) FM, предложенная К. Похлом. Рассмотрены другие подходы к конфигурационной сборке (К. Чернецки, К. Ленц и др.), а также Grid-системы и фабрики программ — AppFab. Представлена технология разработки вариантов систем методом конфигурационной сборки.

Глава 4. Модели и методы проектирования вариантов систем. Рассмотрена концепция формирования вариантов систем из КПИ, артефактов и продуктов. Определены внутренние и внешние характеристики КПИ и вариантные точки для внесения изменений. Задана технология управления вариантами в языке C#. Рассмотрены базовые модели (MDA, MDD, PIM, PSM, UML) моделирования систем. Представлены методы генерации, сборки и управления вариабельностью.

Глава 5. Определения характеристик систем методами анализа и извлечения знаний. Рассматривается моделирование функциональных характеристик с помощью методов анализа иерархий, байесовских сетей, методов Minning и извлечения знаний. Описаны инструменты анализа и извлечения знаний из готовых систем. Приведены предметные области, в которых используются методы извлечения знаний и реинженерии систем.

Глава 6. Формальные методы спецификации, верификации и доказательства правильности систем. Дана классификация методов спецификации моделей систем (Z, VDM, B, UML и др.) и теория доказательства программ с помощью утверждений, пред- и постусловий. Описаны методы статической и динамической верификации на основе онтологии OWL, а также представлена логика спецификации модели FM и средств анализа таких моделей.

Глава 7. Методы проектирования моделей доменов и систем. Рассмотрен метод ОКМ моделирования доменов из объектов с использованием логико-математического аппарата и построения графовой объектной модели из объектов и их интерфейсов. Определена компонентная теория разработки и изменения систем из объектов и компонентов. Изложен метод управления конфигурационной сборкой ПС из КПИ, хранящихся в репозиториях или библиотеках.

Глава 8. Применение теоретического аппарата фундаментальных и общих типов к неструктурным данным Big Data. Приведена аксиоматика ТД: фундаментальные, общие GDT (ISO/IEC 11404–2007) и неструктурированные данные в Big Data. Рассмотрены механизмы формального описания разных фундаментальных, общих ТД и данных, генерируемых с разных устройств. Определены функции преобразования разнотипных данных, передаваемых между разнородными системами и средами.

Глава 9. Онтологический метод концептуального моделирования доменов. Описан базис онтологии в языке OWL и подход к описанию систем средствами языка DSL. Показаны онтологии доменов — «Вычислительная геометрия», ЖЦ ПС и КПИ, сформированные в процессе обучения студентов МФТИ. Описана оригинальная концептуальная модель ЖЦ в DSLToolMS.Net и в языке Protégé, а также модель тестирования ПС, реализованная в среде Eclipse.

Глава 10. Методы экспертирования, тестирования и оценки качества ПС. Изложены подход к экспертизе процессов и программ, тестирование ПС и сбор данных об ошибках. Тестирование проводится с использованием модели вариантов и нового метода расчета оптимального времени тестирования ПС. Рассмотрена модель требований, прогнозирования безотказной работы ПС и оценки качества ПС и ПП.

Глава 11. Метод разработки веб-систем в Semantic Web. Дано определение понятий Semantic Web. Приведены основные компоненты и стандарты комитета W3C (BPMN, BPML, SOAP, OWL и др.) для описания веб-систем и онтологий. Рассмотрены веб-сервисы, модели SOA, SCA и набор композитных сервисов, используемых при решении бизнес-задач. Изложены методы организации вычислений ПС в Cloud Computing и с Big Data.

Глава 12. Основы Software Engineering Method and Theory (SEMAT). Изложены теоретические аспекты дисциплин SE и ЖЦ. Рассмотрены подходы к разработке новых методов и теории SEMAT, ориентированных на представление и получение теоретических знаний и навыков по специальности SE у студентов и разработчиков ПП.

Приложение 1. Моделирование систем диаграммами UML.

Приложение 2. Сборка систем в .NET

Приложение 3. Взаимодействие систем в Интернете.

Приложение 4. Первые всесоюзные конференции по технологии программирования (1968—1992).

Введение

Программирование возникло в связи с появлением первых ЭВМ как способ задания последовательности операций для решения математических и физических задач на ЭВМ. Первоначально программы описывались в языках, близким к машинному языку. Потом появились ЯП (Адресный, Автокод, Алгол, Кобол, Фортран, ПЛ/1 и др.), с помощью которых описывались разного рода программы. Для того чтобы программу понимала машина, необходимо было ее переводить на язык машины. Их перевод осуществлялся с помощью появившихся средств автоматизации — программирующих систем (трансляторов и интерпретаторов).

Отрабатывалась новая техника описания алгоритмов математических, физических и других задач в виде некоторой формальной схемы программы. Сформировались теории А. А. Ляпунова, В. М. Глушкова, А. П. Ершова, Е. Л. Ющенко и др. Эти теории были ориентированы на формализацию логических схем программ и машин и их переписывание в ЯП, для которых были разработаны программирующие программы (трансляторы, интерпретаторы, генераторы) на ЭВМ.

Разработкой трансляторов с разных ЯП для отечественных ЭВМ занимались научные школы программирования (ИПМ, М. Р. Шура-Бура, Москва; ЛГУ, С. С. Лавров, Ленинград; СОАН, А. П. Ершов, Новосибирск; ИК, В. М. Глушков, Киев и др.). Позже такие школы были созданы в Риге, Таллине, Минске, Кишиневе, Ереване и др. В этих школах сформированы направления отечественной ТП, основу которых составляли ЯП, методы и средства автоматизации программ на разных ЭВМ. ТП охватывала не только трансляцию программ, но и их отладку и организацию вычислений под управлением специально созданных программ операционных систем для каждой ЭВМ.

Для координации и финансирования работ по ТП в СССР был создан Госкомитет по науке и технике (ГКНТ). Он поддерживал разработки и развитие научных основ вычислительной техники и ТП. В рамках ГКНТ создавались государственные комиссии с привлечением известных специалистов страны для приемки новых средств вычислительной техники и технологий программирования, изготовленных в разных организациях страны.

В 1970-х гг. проводилась массовая разработка программных модулей для решения разных задач на ЭВМ. ГКНТ ставит задачи автоматизации разных отраслей промышленности, крупных военно-промышленных комплексов (ВПК), АСУ, АСУ ТП, САПР и др. ГКНТ финансировал системы автоматизации модульного программирования («АПРОП», «ПРОЕКТ», «АЛЬФА», «СИРИУС» и др.), технологический комплекс «РТК»

с *Р*-графическим языком задания программ и документов АСУ, систем «ПРОТВА», «РУЗА» (В. В. Липаев) по изготовлению больших программ для ВПК и др. Главным элементом разработки систем стал модуль, выполняющий некоторую функцию и описанный в стандартном виде. По решению ГКНТ для хранения готовых отдельных модулей и программ общего назначения созданы государственные и республиканские Фонды алгоритмов и программ (1970—1992). В стране был создан в Калинине (ныне Тверь, 1980) первый завод для сборки ПО АСУ из готовых программ разных фондов. После двух лет работы этот завод перестал существовать из-за отсутствия необходимого набора готовых программ и КПИ.

Созданные системы автоматизации «АПРОП», «ПРОТВА», «РТК» и др. внедрялись в разные организации страны и служили способом автоматизации разных задач. Кроме того, были созданы системы АСУ в нашей стране, а также в Болгарии, ГДР и др.

Для обсуждения проблем автоматизации разных систем в стране ГКНТ проводил всесоюзные конференции по ТП. Первая Всесоюзная конференция по программированию прошла в Киеве (1965), вторая — в Новосибирске (1978) и последующие конференции по ТП проводились в 1979, 1982—1992 гг. В работе каждой конференции принимали участие тысячи специалистов, вовлеченных в автоматизацию разных задач. В организации этих конференций принимали участие видные специалисты страны в ТП: В. М. Глушков, А. П. Ершов, Е. А. Жоголев, Л. Н. Королев, Э. Х. Тыугу, М. Р. Шура-Бура, Е. Л. Ющенко, и др.

Инженерия ПО (Software Engineering; SE) сформировалась для первых зарубежных ЭВМ, аналогично ТП, и была представлена П. Науром на конференции НАТО (1968). В США проводились разработки ПО систем общего назначения для разных ЭВМ (IBM, Intel, Apple, Sun и Microsoft), а также прикладных систем в ЯП для этих ЭВМ.

Были созданы международные комитеты ISO/IEC, ACM и IEEE для проведения работ по созданию средств автоматизации ПО ЭВМ и систем. Комитет выпустил стандарт разработки систем: SWEBOK (2001, 2004, 2007, 2014); ISO/IEC 12207 — ЖЦ ПО, оценки качества ПО, стандарт описания многоразовых КПИ (Reusebility, 1987—2012) и др.

SWEBOK (Software Engineering Body Knowledge) сформировался в США (www.swebok.com). Это процесс разработки ПО в виде областей знаний, отражающих последовательность процессов проектирования ПО систем. Для поддержки областей знаний разработаны специальные CASE-инструменты (Rational Rose, COM, CORBA, UML и др.), а также стандарт ISO/IEC ЖЦ 12207, регламентирующий процессы разработки систем согласно областям знаний SWEBOK. В его последних версиях определены новые дисциплины SE (теория, экономика, менежмент и др.) и новые парадигмы программирования: объектно ориентированные; клиент-серверные, компонентные, сервисные, системы реального времени, бизнес-системы и др.

Начиная с 2000 г. многие крупные фирмы и отдельные специалисты за рубежом проводят работы по обеспечению изменяемости (вариабельности) систем и ПП. Так, в SEI USA разработан метод производства ПП —

Product Line/ Product Family со сборкой КПИ (компонентов, reuses, assets и др.) по заказу потребителей. Теоретическую основу концепции вариабельности сформулировали зарубежные специалисты (К. Похл, К. Чернецки, Т. Бёгнер и др.). Определена модель характеристик FM (Feature Model), включающая внешние характеристики (свойства) создаваемого ПП. Разработаны новые методы верификации FM и конфигурационной сборки ПП из КПИ и других ГоР. Представлены отдельные аспекты Semantic Web и создания ПП с помощью сервисов Интернет.

Основные разработчики SE постоянно развивали новые теоретические методы изготовления ПО систем. В 2009 г. вышел новый стандарт SEMAT, ориентированный на дальнейшее развитие теории и методов изготовления ПП и систем, в том числе логического и математического моделирования систем и моделей (http://www.semat.org). Для формального доказательства этих моделей необходимы математический аппарат и формальные языки спецификации типа VDM, RSL, Z, UML и доказательства программ методами верификации моделей и систем. Появились новые методы анализа характеристик систем средствами иерархических и байесовских сетей, а также методы анализа и извлечения знаний Minning.

Развитию теории проектирования ПС из КПИ и ГоР способствовали сформулированные автором формальные дисциплины SE (теория, инженерия, экономика, менеджмент и производство), теория ОКМ, а также стандарт ICO/IEC PMBOK, нацеленный на управление программными проектами и обнаружение рисков в разработке качественных систем.

Глава 1
ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ ПРОГРАММНЫХ СИСТЕМ

Под **технологией программирования** понимается совокупность методов, средств и инструментов для проведения процесса программирования задач предметной области с целью получения ПП с заданными свойствами и функциями. Методами могут быть: модульный, объектный, компонентный и др. К средствам относятся ЯП, а также Analytic, Domain Specific Language, VDM, SADT, SSADM, OMT и др. Инструменты — это системы программирования, отладчики, редакторы и CASE-системы (UML1, UML2, Rational Rose и др.) и ТЛ разработки вариантов некоторого продукта, в которых отображается регламентированная последовательность действий над объектом разработки, начиная от требований, задания схемы (графа) системы из ее элементов и действий по их реализации в конкретной технологической среде. Каждое действие поддерживается системным или специальным инструментом, на вход которого подается промежуточный результат и на выходе которого получается готовый выходной код.

В ТЛ указываются оборудование, на котором надо работать, и технология его использования с заданным порядком выполнения операций и действий на линии. Выходной результат линии — продукт и документация, а также инструкции по ее применению, получению вариантов продуктов и проведению модификаций ПП в процессе эксплуатации и замены непригодных инструментов или функций новыми, более совершенными. Технологический процесс — это общее базовое понятие в индустрии ПП, компоненты которого могут применяться по-разному, в разных вариантах, в зависимости от требований и условий использования. Таким образом, на практике долгие годы в СССР использовалась технология программирования.

В данной главе рассматриваются фундаментальные основы ТП, системы программирования, технология модульного программирования, широко используемая в СССР для производства программных продуктов и систем различного назначения.

1.1. Технология программирования сложных систем

Программирование вычислительных задач начало развиваться на цифровых ЭВМ с помощью языков представления алгоритмов и программ, их преобразования в коды машин средствами ППр или систем программирования (трансляторов, интерпретаторов) с языков описания программ (Адрес-

ный, Алгол-60, Фортран, Кобол, Алгол-68 и др.). Трансляторы разрабатывались для отечественных ЭВМ («БЭСМ», «М-20», «М-50», «Минск-32», «Стрела», «СМ1-3», «Днепр-2» и др.) в рамках ТП. Создавались системы автоматизации методами синтезирующего, сборочного программирования, технологического комплекса «РТК» с Р-графическим языком задания программ и документов и технологии В. В. Липаева[1] по изготовлению больших программ для ВПК и др.

1.1.1. Создание систем программирования

Первые советские монографии по программированию — «Элементы программирования» А. И. Китова, И. А. Криницкого (1956), «Быстродействующие электронные машины АН СССР» С. А. Лебедева (1952), «Общее описание БЭСМ и методика выполнения операций» С. А. Лебедева и В. А. Мельникова (1958) и др. — освещали аспекты создания программ решения разных математических задач на ЭВМ. Они открыли эру алгоритмизации вычислительных задач с помощью языков, близких ЭВМ. Это также работы «Алгоритмы и машинное решение задач» (В. А. Трахтенгерц, 1957), «Начальные сведения о решении задач для ЭВМ» (А. А. Ляпунов и Г. А. Шестопалов, 1957), «Граф-схемное программирование» (А. А. Ляпунов, А. П. Ершов), «Адресное программирование» (Е. Л. Ющенко, 1962) и др.

Основные элементы отечественной ТП

Программирующая программа. Первая идея ППр возникла в 1953 г. у А. А. Ляпунова, который читал курс по теории представления алгоритмов с помощью операторных граф-схем в МГУ. На языке операторных схем программа представлялась схемой, в виде управляющего графа и совокупности спецификаций его операторов. В основе входного языка каждой из ППр лежал общий концептуальный базис, фиксирующий типы операторов и их спецификации. Программа ППр-1 сделана в 1954 г. с участием Э. З. Любимского, А. П. Ершова в МГУ под руководством М. Р. Шура-Бура. Это первая ППр в мировой практике. ППр-1 стала прототипом для проектирования других ППр на машинах «БЭСМ», «Стрела», «М-20» и др. В результате были созданы фундаментальные алгоритмы трансляции и положено начало системному и теоретическому программированию. Важную роль сыграли теоретические работы: А. П. Ершова «Введение в теорию программирования» (1977), В. М. Глушкова, Г. Е. Цейтлина и Е. Л. Ющенко «Алгебра. Языки. Программирование» (1974), М. Р. Шура-Бура, Э. З. Любимского «Транслятор ТА2 на языке АЛМО» (1970) и др.

Идея ППр развивалась и в ВЦ АН УССР в период создания машины МЭСМ А. А. Лебедевым. В. М. Глушков в статье «Об одном методе автоматизации программирования» (журнал «Проблемы кибернетики». 1957. № 2. С. 181—189) и А. А. Стогний (там же, с. 190—199) в статье «О прин-

[1] См.: *Липаев В. В.* Программная инженерия. Методические основы. М. : ТЕИС, 2006; *Его же*. Программная инженерия сложных заказных программных продуктов. М. : Изд-во института системного программирования, 2014.

ципах построения специализированной ППр» определили библиотечный метод алгоритма решения систем дифференциальных уравнений. ПП использовались на разных машинах: «УМШН», «Проминь», «Днепр», «Днепр-2» и др. Трансляторы для этих машин были реализованы в адресном языке Е. Л. Ющенко и в специализированных языках типа Автокод.

Библиотеки стандартных программ. Е. А. Жоголев (МГУ) разработал стандартную программу, которая осуществляла статическую загрузку и связывание программ из библиотеки численных методов. Интерпретирующая система ИС-2 (М. Р. Шура-Бура) использовалась на машинах «М-20», «БЭСМ» и др. В ней реализованы метод динамического вызова библиотечных подпрограмм, способы связывания, подкачки и замены. ИС-2 стала неотъемлемой частью разных ОС ЭВМ, поставляемых пользователям[1].

Операционные системы. Для «БЭСМ-6» была сделана операционная система (ОС) в Институте точной механики и вычислительной техники под руководством В. П. Иванникова, которая долгое время выполняла общие функции управления памятью, задачами и данными. Дальнейшее развитие ОС получила в вычислительной системе «Электроника ССБИС» (1985—1991). Ее цели и задачи опубликованы в ряде статей, в том числе и в итоговой статье[2]. В ней представлены ОС и ее ядро. В это ядро входят стандартные средства ОС, средства взаимодействия с IBM и Эльбрус, набор протоколов обслуживания программ и система программирования с ЯП (Фортран, С, Паскаль и др.). Базовая система программирования включает новые средства поддержки абстрактных ТД, кластерного сборочного конвейера для загрузки модулей в ОС и в библиотеку программ трансляторов, а также отладчик программ на ЯП и сервисные средства обслуживания пользователя.

Системы программирования

С 1960 г. в стране появилось много языков ЯП: Адресный, Алгол-60, Фортран, Кобол, Пролог, Ада и др.

Одним из первых языков, используемых для программирования математических задач и ПП, был Адресный язык.

Адресный язык (Е. Л. Ющенко, Б. В. Гнеденко, В. С. Королюк) был близок к языку математики. Алгоритм в этом языке записывался набором строк операторов, в каждой из которых — одно или несколько действий — формул. Для задания порядка операторов используются метка и операторы безусловного перехода. Переменные обозначались буквами и соответствовали ячейкам машины. Содержимое некоторого адреса отмечалось указателем, адресом второго ранга и использовалось для многократного перехода. Этот язык также применялся для описания программ вычислительного характера и реализации ПП для машин «Киев», «Урал» и «Днепр».

---

[1] См.: *Жоголев Е. А.* Система стандартных подпрограмм / Е. А. Жоголев [и др.]. М. : ГИФМЛ, 1958.

[2] *Иванников В. П., Гайсорян С. С., Томилин А. Н.* Системное программное обеспечение вычислительной системы «Электронника ССБИС. SORUCOM.2014» // Труды Межд. конференции «Развитие ВТ и ее ПО в России и в странах бывшего СССР. История и перспективы», 15—17 октября 2014 г., Казань. С. 117—125.

Язык Алгол-60. Он возник в США как универсальный ЯП научных и инженерных задач[1]. Этот язык в ВЦ АН СССР был принят в качестве языка описания научных задач. Сформировалось следующие направления разработки трансляторов с языка Алгол (ТА): ТА1 — С. С. Лавров (ЛГУ, 1962), ТА2 — М. Р. Шура-Бура и Э. З. Любимский (ИПМ, 1965), ТА-3 (Альфа-система) в русской версии языка Алгол-60 (СО АН СССР, 1966), ТА-4 — Е. Л. Ющенко, Е. М. Лаврищева — для УВК «Днепр-2» (ИК АН СССР, 1967). Позднее, в 1970-х гг. делались еще трансляторы: с языка Алгол-68 (А. Н. Терехов, ЛГУ), с языка Кобол (Л. П. Бабенко для «Днепр-2» и О. Н. Романовская для «Минск-32») и др.[2]

На основании этого языка в трансляторе ТА-1 реализованы простая схема трансляции, стековый подход к трансляции выражений, процедур без рекурсий. В трансляторе ТА-2 разработан оригинальный алгоритм реализации процедур, механизм управления памятью (оперативной и внешней) и метод таблично-управляемой генерации кода. В ТА-3, АЛЬФА-трансляторе[3] осуществлена оптимизация (выражений, циклов, процедур, памяти и др.) для повышения эффективности выходного кода, который был близок коду, созданному вручную. В ТА-3 реализованы также операции над многомерными значениями и комплексными ТД. В трансляторе Д-АЛГАМС (ТА-4) — СМ-метод табличного представления БНФ и набора семантических программ их трансляции[4]. Новым подходом к трансляции системы программирования «Днепр-2» было создание общего арифметического блока для двух языков Алгол и Кобол.

Трансляторы ТА1—ТА3 и ОС для новых ЭВМ («М-20», «СТРЕЛА», «БЭСМ») были реализованы в машинном коде этих машин, а транслятор ТА4 — на языке, близком Адресному языку, — Автокод «Днепр-2». Трансляторы АКД, Д-АЛГАМС и Кобол входили в состав общесистемного ПО машины «Днепр-2». Трансляторы с этих языков были сданы в составе ОМО (ОС и систем программирования АКД, Д-АЛГАМС, Кобол) УВК «Днепр-2» Госкомиссии СССР в 1967 г. Комиссию возглавлял А. В. Дородницын. В ее состав входили Л. Н. Королев, В. М. Глушков, А. Н. Кухарчук и др. Эта машина была приобретена ГДР в 1971 г., ПО машины было внедрено в вычислительном центре немецкой металлургической фирмы BMHW и работало до 1991 г. С помощью этих систем совместно специалистами ГДР и Украины разрабатывалась АСУ ТП металлургическими процессами. Работала она там до 1992 г. Следует отметить, что одновременно для «Минск-32» делался транслятор с языка Кобол сотрудниками ИНЭУМ Белоруссии.

---

[1] См.: Сообщение об алгоритмическом языке Алгол-60 / под ред. А. П. Ершова. М., 1960.

[2] См.: *Лаврищева Е. М., Грищенко В. Н.* Связь разноязыковых модулей в ОС ЕС. М. : Финансы и статистика, 1982.

[3] См.: *Бабецкий Г. И.* Система автоматизации программирования АЛЬФА // Журнал вычислительной математики и математической физики. 1965. Т. 5. № 2.

[4] См.: *Yushchenko E. L., Lavrishcheva E. M.* A method of analyzing programs based on a machine language // Springer. 1972. Vol. 8. №. 2. P. 219—223; и др.

Системы программирования включали трансляторы, отладчики и редакторы программ. Э. З. Любимский (1963) предложил промежуточный язык — АЛМО — для перевода теста в любом ЯП в этот язык, а затем теста АЛМО в код ЭВМ[1]. АЛМО — это некоторая абстрактная машина, отражающая особенности класса ЭВМ в СССР. Этот и другие языки (Эпсилон, Сигма) стали языками-посредниками при трансляции программ с различных ЯП. Их смысл состоял в замене трансляции с m входных языков в n машинных языков, т.е. «из m в один» и «из одного в n». АЛМО реализован для основных машин того времени («М-20», «БЭСМ-6», «Минск-2», «Урал-1») и проверен для трансляторов с Алгол-60 и Фортран[2].

Одной из систем программирования с ЯП была система с языка Алгол-68 (Г. С. Цейтин, А. Н. Терехов, ЛГУ, 1968—1991), переведенного на русский язык в 1979. Многие годы разрабатывались варианты трансляторов в ЛГУ, Минске, Новосибирске (Бета). Однако наиболее реальный транслятор был сделан под руководством А. Н. Терехова для первых ЭВМ — СМ1, СМ2, а потом для ОС единой серии ЭВМ. НИЦЕВТ финансировал эту работу. Это была одна из самых сложных и трудных разработок в стране. В работе «Процессы идентификации и структура транслятора с языка Алгол-68» А. Н. Терехов описал наработанные многими членами рабочей группы по Алгол-68 вопросы реализации транслятора и программ перевода языковых конструкций сначала в промежуточный язык, а потом в код машины. История развития и создания системы с языка Алгол-68 описана в статье «Алгол-68 и его влияние на развитие программирования в СССР в России».

Опыт разработки программ на Алгол-60 и первые трансляторы обсуждались на Первой Всесоюзной конференции по программированию в Киеве (1965), Второй конференции в Новосибирске (1978) и на последующих конференциях по ТП (1979, 1982—1992). В работе каждой конференции принимали участие более 1000 человек. Это объясняется большой популярностью данной проблематики в стране. По линии ГКНТ (Госкомитет по науке и технике), ГКНТИ (ГКНТ и информатике) СССР в Алуште под руководством проф. Е. Л. Ющенко проводились Всесоюзные конференции (1981—1990) по разным аспектам ТП. На них обсуждались новые теоретические и прикладные аспекты проектирования, разработки, тестирования различных видов систем для ЭВМ, а также задачи эксплуатации и сопровождения ПО. В развитие отечественной ТП внесли вклад ведущие специалисты академических институтов АН СССР: В. М. Глушков, И. В. Вельбицкий, А. П. Ершов, Э. З. Любимский, В. С. Лавров, М. Р. Шура-Бура, К. Л. Ющенко, Э. Х. Тыугу и др. ГКНТ СССР финансировал работы по разработке средств автоматизации ТП. Были специальные постановления Кабинета министров СССР, направленные на развитие средств вычис-

---

[1] См.: *Шура-Бура М. Р., Любимский Э. З.* Транслятор с языка Алгол-60 // Журнал вычислительной математики и математической физики. 1964. Т. 4. № 1.

[2] См.: *Жоголев Е. А.* Система стандартных подпрограмм; *Лавров С. С.* Основные понятия и конструкции ЯП. М. : Финансы и статистика, 1982; *Лаврищева Е. М.* Транслятор с языка Д-АЛГАМС для УВК «Днепр-2» / Е. М. Лаврищева [и др.]. Киев : Изд-во ИК АН УССР, 1970; *Лаврищева Е. М., Грищенко В. Н.* Связь разноязыковых модулей в ОС ЕС.

лительной техники и технологии изготовления ПП и сдачи их в фонды алгоритмов и программ для использования другими программистами.

Р-технология. Для конструирования структур произвольных программ с помощью визуальных *Р*-графов и схемной реализации программ разработано устройство синтаксического контроля программ за несколькими отечественными патентами[1]. Выполнена структурная интерпретация синтаксиса и семантики ЯП с использованием этого языка. Графический стиль программирования использовался во многих организациях военно-промышленного комплекса СССР. Затем был создан технологический комплекс программиста для проведения синтаксического анализа программ в *Р*-языке.

Ведение новых СП в эксплуатацию

Созданные в стране СП (трансляторы с входных языков типа Алгол (ТА1-ТА4), средства отладки программ в системе) передавались в организации, где находились новые ЭВМ, вместе с документацией, включающей:

- описание входного языка;
- руководство программиста;
- руководство системного программиста;
- инструкцию по запуску и сопровождению.

Эти системы проходили опытную эксплуатацию в течение нескольких лет на примерах задач (вычислительных, экономических, планирования, управления и др.). Находились не только собственные ошибки в программах, но и в СП и ОС. Системные ошибки устранялись разработчиками систем, которые потом передавали новую версию организациям пользователей. В качестве примера можно привести ОС 360 IBM, которая выдала организациям-пользователям более 200 версий на протяжении двух десятков лет. Потом был сделан стандарт ОС IBM 370. Аналогично было с отечественными СП. Появлялись другие языки ЯП (Ада, Фортран, Пролог и др.), СП которых проходили длительную отработку, пока они получали статус промышленного применения. Документация на системы размещалась в фондах алгоритмов и программ для приобретения любыми пользователями. Позднее Кабинет министров СССР принял решение о межведомственной сдачи ОС и СП. В состав комиссий по приемке систем входили видные специалисты страны в области ЭВМ. Однако стандартной версии ОС и СП почти никто из разработчиков систем не сделал. На некоторых системах в закрытых организациях страны до сих пор работают отдельные ОС и СП.

1.1.2. Системы синтеза, композиции и сборки программ

Система «ПРИЗ» разработана академиком ЭССР Э. Х. Тыугу для синтеза программ в *языке Утопист* и ЯП на основе задания семантической модели ПрО решения математических задач в ППП. Метод синтеза семан-

[1] См.: *Вельбицкий И. В.* Технология программирования. Киев : Техника, 1984; *Lavrischeva E. M.* Developmentof Domestic Programming Technology // Journal Cybernetics and Systems Analyses. 2014. Vol. 50. № 3. May. P. 452–264.

тических программ в PL/1, Фортран, Assembler реализован методом подстановки прикладным задачам семантики их реализации в ЯП и операциями системы «ПРИЗ» в ОС единой серии[1].

Композиция программ. Композиционное программирование[2] служило способом объединения функций и данных такими цепочками: «данные — функция — имя», «функции — композиция — дескрипция». Программной поддержкой этих цепочек являлась система «ДЕФИПС», которая обеспечивала построение программ из функций, заданных на некотором множестве именованных данных, дескрипций и денотатов (значений). Семантика задается ординарными функциями обработки операций, интерфейсных функций, арных функций и именованных данных. Операции композиции — это подкласс стандартных композиций и декомпозиционных функций.

Сборка программ. Индустриальный выпуск сложных программ В. М. Глушков определил как сборочный конвейер Форда. По его мнению, фабрика оборудуется линиями программирования разных программ и их интеграции в новую программную структуру. Базисом сборки были модули — главный элемент программирования. Обмен готовыми модулями потребовал разработки *паспорта*, который содержал описание входных и выходных данных, а также операторов вызова (Call, RMI и др.) других модулей. Модули описывались в разных ЯП и имели паспорт, задающий связь (интерфейс) с другими модулями. Это описание упрощало процесс сборки разноязыковых модулей, записанных в широко используемых ЯП (Алгол, Кобол, Фортран, ПЛ-1, Модула и др.). Позднее (1994) для описания интерфейса были созданы язык IDL (Interface Definition language) и брокер объектных запросов (ORB) в системе CORBA в рамках объектного подхода. Описание интерфейса через посредников *stub* и *skeleton* стало основным элементом сборки любых модулей, объектов и компонентов.

Вопросом сборки модулей занимались Е. П. Жоголев, Е. М. Лаврищева, В. И. Орлов, В. Г. Волховер и др.[3] В 1980-х гг. бурно развивалось направление автоматизации прикладных систем и ППП. В Институте кибернетики Академии наук УССР был выполнен ряд проектов, финансируемых ГКНТ СССР. Это проекты: система автоматизации программ — «АПРОП» (Е. М. Лаврищева); комплекс программиста «ТКП» (И. В. Вельбицкий); система алгоритмических алгебр «Мультипроцессист» (Г. Е. Цейтлин) и др. Средства автоматизации разработаны и в ряде других научных центров СССР: система «ПРИЗ» (ЭССР, Э. Х. Тыугу), системы «Альфа», «Бета» (Новосибирск, А. П. Ершов), система автоматизации математических задач (ИПМ АН СССР, Д. Н. Корягин), система модульного программирования (МГУ, Е. А. Жоголев), система послойного проектирования ПО (Ростов-на-Дону, В. И. Фуксман) и др.

---

[1] См.: *Кахро М. И., Тыугу Э. Х.* Инструментальная система программирования на ЕС ЭВМ (ПРИЗ). М. : Финансы и статистика, 1981; *Тыугу Э. Х.* Концептуальное программирование. М. : Наука, 1984.

[2] См.: *Редько В. Н.* Композиции программ и композиционное программирование // Программирование. 1978. № 5. С. 17—26.

[3] См.: *Волховер В. Г., Иванов Л. А.* Производственные методы разработки программ. М. : Финансы и статистика, 1983.

При выполнении научных проектов в СССР сформировались разные аспекты ТП для производства прикладных систем, АСУ и ППП с использованием готовых модулей, названных компонентами повторного использования (КПИ). Выходило постановление Кабинета министров СССР «Программы — продукты производственного назначения» (1974) и постановления ГКНТ СССР о том, что программы общего назначения должны были сохраняться в фондах алгоритмов и программ республиканского значения (1976—1992). В рамках Всесоюзной программы по ТП проводилась разработка стандартов ГОСТ 9126 и качества ПС.

Отработка разных аспектов создания сложных программ и ППП осуществлялась на всесоюзных конференциях по ТП (1968, 1979, 1980, 1986, 1987, 1991, 1992 и др.). Наиболее распространенной тематикой были технология модульного создания систем и интерфейс связей модулей и программ. Эти вопросы рассматривались на Международной конференции «Интерфейс СЭВ — 1987». Краткая характеристика работ на конференциях по ТП в СССР приведена в приложении 4.

1.1.3. Технология программирования систем для ЭВМ

В 1960—1970-х гг. для решения разных народно-хозяйственных задач применялись новые отечественные ЭВМ («БЭСМ», «МЭСМ», «М-20», «Днепр-1», «Днепр-2», «Стрела» и др.). Для каждой их них сформированы ТП, включающие набор методов и средств для разработки сложных систем. К ним относились СП и средства отладки программ (на правильность синтаксиса и семантики). Позже появились методы тестирования отдельных программ и систем. С их помощью находились функциональные ошибки, отказы в связи с недоработками как ОС, так и изготовляемых систем.

Определение программной системы и методы ее построения

Под **системой** понимается совокупность взаимодействующих элементов, работающих совместно для достижения заданных целей. Система первоначально была иерархической структурой, в которой в качестве элементов могли быть подсистемы, каждая из которых функционировала самостоятельно и связывалась с другими для обмена данными. Так как работа системы зависит от аппаратуры ЭВМ, то формировалась системотехника как технология создания систем, охватывающая все аспекты создания и модернизации программных и технических средств этих систем.

Результаты, полученные при системном тестировании систем, оценивались на надежность и качество. На основе опыта программирования и тестирования систем были опубликованы монографии В. В. Липаева — «Надежность ПС» и «Качество ПС» и «Проектирование комплексов программ». В это же время вышла в свет работа А. Ф. Кулакова «Качество программ ЭВМ». Появилась теоретическая возможность оценивать качество технических средств и программ. Методы тестирования программ и систем независимо от используемой ЭВМ и видов программ проводили проверку правильности их работы. Сформировался определенный порядок проведения разработки систем — так называемые модели ЖЦ (итерационная, спиральная и др.), которые включают следующие этапы создания системы:

• определение целей и задач в техническом задании на разработку системы;

• проектирование структуры системы и разработка отдельных ее структурных элементов;

• тестирование элементов и систем;

• испытания системы на множестве данных и тестов;

• подготовка к тиражированию системы для передачи потребителям;

• эксплуатация системы и ее сопровождение;

• снятие с эксплуатации (утилизация).

В 1990-х гг. появится стандарт ЖЦ (1996), в котором были регламентированы этапы ЖЦ для разработки разных сложных программ, а в 2007 г. — окончательный вариант стандарта ЖЦ ICO/IEC 12207.

ТП специализированных ЭВМ

В рамках ВПК разрабатывались (в строгой секретности) специализированные ЭВМ («ПРА-6.0», «МАПА», «АРГОН», «АОУ6» и др.), а также военная техника (5Э89, Э26Б 5Э51 и др.) для решения функциональных задач космических кораблей, подводных лодок, самолетов и др.[1] Они использовались как радиолокационные средства:

• для обнаружения воздушных целей;

• управления средствами противовоздушной обороны;

• телекоммуникационных средств передачи данных на командные пункты и средств поражения противоракетной техники (ПРО и ПВО) и стратегии перехвата закрытых данных;

• оценки надежности систем обработки информации в режиме реального времени и др.

Было создано более 300 образцов военной техники, которые надо было поставить на решение конкретных задач ВПК. Для этого при Министерстве радиопромышленности СССР был создан комитет под руководством главного конструктора В. В. Липаева. Главная его задача состояла в том, чтобы на специализированных ЭВМ создать ПО, основанное на принципах разработки языков и программ как для универсальных ЭВМ. Для этих целей разрабатывались системы «ЯУЗА», «РУЗА» и «ПРОМЕТЕЙ». В НИЦЕВТ в бортовую машину «АРГОН» включалась система команд машин ЕС ЭВМ. Затем потребовалось разрабатывать новые интерпретаторы и кросс-системы, с помощью которых усовершенствовались процессы разработки функциональных программ на бортовых ЭВМ в ВПК. На этих машинах создавались специальные средства разработки, отладки и испытания программ в режиме разделения времени.

Разработка системы «ПРОТВА» проводилась по ТП, с помощью которой решались следующие задачи:

• планирование трудоемкости и длительности создания ПО;

• прогнозирование и оценивание реального состояния качества в зависимости от обнаруженных ошибок;

---

[1] *Липаев В. В.* Фрагменты истории развития отечественного программирования для специализированных ЭВМ в 50—80-е годы. М. : СИНТЕГ, 2003.

• оценка ресурсов ЭВМ по памяти и производительности для реализации ПС;

• прогнозирование и оценка состояния качества ПС, в зависимости от длительности и характеристик обнаруженных ошибок.

В качестве языков использовались: автокоды, макроязыки и ЯП. Для них строились СП и системы проверки правильности по аналогии с большими ЭВМ. Для программистов вводились рекомендации по размеру программного модуля (100—200 строк текста) и тестам проверки до 10 строк. Было сформировано правило: число условий в тестах для покрытия тестами структуры модуля пропорционально квадрату строк текста программы. Было получено ограничение на размер программного модуля не более 300 строк. Эти вещи делались для того, чтобы снизить сложность и повысить качество системы в целом.

При работе с проектом «ПРОМЕТЕЙ» наибольшее внимание уделялось вопросам взаимодействия модулей. Для этого в рамках проекта «ПРОМЕТЕЙ» использовалась система «АПРОП», которая реализовала интерфейсные связи разноязычных модулей. Она была адаптирована в среду системы «ПРОМЕТЕЙ». В ней были предусмотрены средства тестирования интерфейсов связываемых модулей и методы системного тестирования. Для системы «ПРОМЕТЕЙ» была разработана документация согласно требованиям ГОСТа. Система «ПРОМЕТЕЙ» была сдана государственной комиссии и передана в ЕРНУЦ (Ереван). Она успешно применялась на многих предприятиях ВПК и после эксплуатации представлена к премии Кабинета министров СССР (1985).

1.1.4. Развитие технологии сборочного программирования

Технология программирования развивалась по трем направлениям, сформулированным в статье В. М. Глушкова «Фундаментальные основы и технологии программирования» (Программирование. 1980. № 2):

• модульная система автоматизации производства программ («АПРОП») из стандартизированных программных заготовок в сложные системы[1];

• метод формализованных технических заданий для проектирования сложных программных комплексов «ПРОЕКТ»[2];

• *Р*-технология программирования для автоматизации проектирования систем средствами графического *Р*-языка для представления структур программ и данных в АСУ[3].

Модульная система автоматизации программ «АПРОП» разрабатывалась исходя из тезиса В. М. Глушкова — конвейерная сборка разнородных моду-

---

[1] См.: *Глушков В. М.*, *Стогний А. А.*, *Лаврищева Е. М.* Система автоматизации производства программ (АПРОП). Киев : Изд-во ИК АН УССР, 1976; *Лаврищева Е. М.*, *Грищенко В. Н.* Связь разноязыковых модулей в ОС ЕС; *Лаврищева Е. М.* Вопросы объединения разноязыковых модулей в ОС ЕС // Программирование. 1978. № 1.

[2] См.: *Глушков В. М.*, *Капитонова Ю. В.*, *Летичевский А. А.* О применении метода формализованных технических заданий к проектированию программ обработки структур данных // Программирование. 1978. № 6. С. 5—12.

[3] См.: *Вельбицкий И. В.* Технология программирования; *Вельбицкий И. В.*, *Ходаковский В. Н.*, *Шолмов Л. И.* Технологический комплекс автоматизации программ на машинах ЕС ЭВМ и БЭСМ-6. М. : Статистика, 1980.

лей и интерфейсов (Е. М. Лаврищева) и в системе «ПРОЕКТ» (Ю. В. Капитонова, А. А. Летичевский). На этой основе создавались ППП (И. Н. Молчанов); ППП математического и статистического типов (И. В. Сергиенко, В. Н. Редько, А. С. Стукало); *Р*-технология (И. В. Вельбицкий), «ТЕРЕМ» (Н. М. Мищенко). Этими работами был внесен весомый вклад в сборочное создание ПП на ЕС ЭВМ.

Система АПРОП разрабатывалась по договору с Институтом приборостроения (Москва) в составе технологии создания программ для бортовых систем «ПРОТВА» в рамках Министерства радиопромышленности СССР, реализованной под руководством В. В. Липаева[1]. В эту систему были внедрены интерфейс и библиотека интерфейсных функций преобразования нерелевантных ТД, описываемых в модулях на разных языках и в разных платформах. Было разработано стандартное описание интерфейса, содержащее описание входных и выходных параметров связываемых между собой модулей. Это описание стало средством интеграции модулей и КПИ. Система «АПРОП» вошла в состав комплексов «ПРОТВА» и «РУЗА», которые реализовали технологию создания бортовых систем для ВПК. Комплекс «ПРОТВА» был представлен на премию СМ СССР (1985).

В ТП сформировался новый вид — сборочное программирование для объединения разнородных модулей (программ) в более сложные структуры. Эта технология развивалась и в отделе В. М. Глушкова для семейства трансляторов с широко используемых ЯП. Основные положения этого вида сборки базировались на идее выделения общих средств в ЯП ОС ЕС, системной реализации компонентов языковых процессоров и сборки из них трансляторов (СПТ ТЕРЕМ). В СПТ разработаны общие компоненты языковых процессов в классе языков ОС ЕС. СПТ включала ЯП ОС ЕС (Алгол, Кобол, ПЛ1 и др.) и построена для МВК с макроконвейерной организацией вычислений и средствами внесения изменений в программы ЯП. Более половины универсальных модулей для ЯП реализованы и вошли в ядро СПТ «ТЕРЕМ», которая сдана Госкомиссии в 1989 г.

В системе «ПРОЕКТ» реализован **метод формализованных технических заданий** для проектирования дискретных систем с применением теории дискретных преобразований. Основу метода составляет формальное описание функций системы с помощью языка L2, аналогичного языку Аналитик, и доказательства правильности функций с помощью дискретных преобразователей и с применением алгебраических операций и средств обработки разных структур данных. К структурам данных относятся простые (числа, символы) и сложные данные (массивы, указатели). Вводится понятие схемы программы, содержащей множество переходов и состояний. Дается описание алгоритма проектирования системы по данному методу и системы обработки данных с применением метода К. Флойда. Метод прошел внедрение на практике при разработки ПО системы «МАЯК».

---

[1] См.: *Липаев В. В.* Технология проектирования комплексов программ. М. : Радио и связь, 1982; *Липаев В. В.*, *Позин Б. А.*, *Штрик А. А.* Технология сборочного программирования. М., 1992.

***P*-технология** создана под руководством И. В. Вельбицкого для конструирования структур программ с помощью визуальных *P*-графов и схемной их реализации. Было разработано устройство синтаксического контроля программ, запатентованное и за рубежом. В нем выполнена структурная интерпретация синтаксиса и семантики ЯП с использованием *P*-языка. Затем был создан ТКП для проведения анализа программ в *P*-языке и комплекс РТК для машин «БЭСМ-6», «ЕС ЭВМ» и «СМ ЭВМ». РТК использовался в различных закрытых организациях СССР для обработки текстовой информации и генерации средств производства программ для АСУ, САПР и ИС.

1.1.5. Интерфейс и метод сборки в технологии программирования

Концепция интерфейса в рамках системы АПРОП включала межмодульный, межъязыковой и технологический. Первое определение понятия *интерфейса* и языка его описания сформулировано в проекте системы в 1976 г.[1] Идея интерфейса намного опередила зарубежные разработки. Был создан язык MIL (*Module Interface Language*) для интерфейса модулей. В зарубежной литературе интерфейс появился позднее. К нему относились программный интерфейс API (*Application Programs Interface*), языковый интерфейс IDL (*Interface Definition Language*), научный интерфейс SIDL (*Scientifically IDL*) и др.

Межмодульный интерфейс — это связующее звено между двумя объединяемыми программными объектами, выполняет функции передачи, приема и преобразования разнотипных данных. Ему затем соответствовал брокерный Stub, Skeleton в системе CORBA (1994)[2] , используемый и сейчас.

Межъязыковой интерфейс — средства представления и преобразования структур данных в ЯП с помощью примитивных функций системных библиотек, которые обеспечивают взаимно однозначную передачу данных между разнородными программами (например, преобразование матрицы по строкам в Фортране в матрицу по столбцам в ПЛ/1 и обратно). Библиотека таких функций была создана в рамках системы «ПРОТВА» (1982—1987) для среды ОС ЕС (ПЛ/1, Algol, Fortran, Cobol, Assembler) и передана в 52 организации СССР по актам внедрения.

Технологический интерфейс — совокупность методов и средств для поддержки процессов ТЛ, включая нормативные, методические документы и формы преобразования данных на процессах ЖЦ. Данный интерфейс включал операции контроля процессов, оценки заданных в требованиях показателей качества, изменения состояния продукта на процессах ЖЦ и др. Методика создания ТЛ (1987) проверена на шести линиях заказа АИС «Юпитер-470» и стала первой работой по формализации ТЛ разра-

---

[1] См.: *Глушков В. М.*, *Стогний А. А.*, *Лаврищева Е. М.* Система автоматизации производства программ (АПРОП).

[2] См.: *Эммерих В.* Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и Java RMI. М. : Мир, 2002.

ботки ПС. Альтернативой ТЛ явились продуктовые линии (ProductLines/ ProductFamily, 2004) института SEI США (http://www.sei.com.edu/ productlines). Концепция технологического интерфейса автора, Г. И. Коваль и Т. М. Коротун соотмечена грамотами европейского конкурса «Интерфейс-СЭВ» (1987)[1].

Созданная нами концепция интерфейса модулей была автоматизирована. Роль генератора интерфейсных модулей-посредников выполняла подсистема реализации интерфейса в системе «АПРОП» (1975—1982). Она описана в монографии «Связь разноязыковых модулей в ОС ЕС» под авторством Е. М. Лаврищевой и В. Н. Грищенко. Эти средства использовались многими в стране и способствовали сокращению объема работ при сборке разноязыковых объектов.

Таким образом, было сформировано сборочное программирование, объектами которого являются любые готовые программы, модули и КПИ, а также интерфейс как способ объединения, комплексирования (по В. В. Липаеву), сборки ПС и семейств ПС из готовых объектов. Он занял свое место среди других методов программирования. Как показали исследования[2], сборка и сейчас остается главным инструментом на многочисленных фабриках программ и сервисов, работающих в среде операционных сред (MS.Net, CORBA, Java, VSphere, Unix, Linex и др.).

Академик А. П. Ершов считал, что «сборочное программирование эффективно, поскольку готовые запрограммированные модули позволяют быстро решить любые задачи из определенной проблемной области для ЕС ЭВМ и мини-, микро- и макро-ЭВМ»[3]. Сборка стала важным технологическим решением в индустрии создания продукции производственно-технического назначения.

Метод сборки разноязыковых программ:

- опубликован в зарубежной прессе *(Lavrishcheva E. M.* Modular design of large programs. Springer. 1980. Vol. 16. № 2. P. 244—249);
- защищен Е. М. Лаврищевой в докторской диссертации «Методы, средства и инструменты сборочного программирования» (1989);
- описан в монографиях «Связь разноязыковых модулей в ОС ЕС», «Сборочное программирование» (Е. М. Лаврищева, В. Н. Грищенко, 1991) и в книге «Технология сборочного программирования» (В. В. Липаев, Б. А. Позин, А. А. Штрик, 1992);
- развит в ОКМ моделирования ПС из объектов и компонентов и обеспечивает сборку КПИ и ПС (описывается в гл. 7 данного учебника);
- представлен в статье Е. М. Лаврищевой «Парадигмы программирования сборочного типа в программной инженерии» (УкрПРОГ. 2014. № 2—3. С. 121—133).

---

[1] *Коваль Г. И., Коротун Т. М., Лаврищева Е. М.* Технологический интерфейс для разработки систем // Труды Межд. конф. «Интерфейс-СЭВ», 1987. С. 97—110.

[2] В частности: *Лаврищева Е. М., Грищенко В. Н.* Сборочное программирование.

[3] *Ершов А. П.* Опыт интегрального подхода к актуальной проблеме ПО // Кибернетика. 1984. № 4. С. 11—21. См. также: *Его же.* Два облика программирования // Кибернетика. 1982. № 6. С. 122—125.

1.2. Технология модульного проектирования систем

Технология модульного программирования основывается на понятии модуля, который осуществляет преобразование множества исходных данных X во множество выходных данных Y и задается в виде отображения:

$$M : X \to Y.$$

На множества X, Y и отображение M накладывается ряд ограничений и дополнительных условий, позволяющих отделить модуль как самостоятельный программный элемент. Модуль обладает множеством свойств и характеристик. Эти свойства проявляются на трех основных процессах разработки системы: проектировании, разработке и выполнении[1].

Проектирование ПС включает определение следующих свойств модулей:

- выполнение одной или более взаимосвязанных функций;
- логическая законченность функции;
- независимость одного модуля от других (внутренняя логика данного модуля не связана с логикой работы других);
- замена отдельного модуля без нарушения структуры;
- вызов других модулей и возврат данных вызвавшему модулю;
- уникальность именования модуля и др.

Процесс разработки модулей основан на следующих свойствах:

- описание модулей на одном из ЯП и представление в виде КПИ;
- обязательное описание характеристик модуля в паспорте;
- задание ограничений на размер модуля.

Процесс выполнения ПС основывается на следующих требованиях к свойствам характеристик модуля:

- возможность использования КПИ в различных точках вариативности ПС;
- передача данных между модулями через вызов CALL;
- изменение передаваемых данных и др.

Не все свойства одинаково важны в процессе разработки ПС. Однако для модульного проектирования характерны две тенденции — усиление *внутренних связей* в модуле и ослабление *внешних связей*. Хорошо спроектированный модуль обладает значительно более сильными внутренними связями, чем внешними. При выполнении этого условия процесс сборки модулей можно рассматривать как сочетание неделимых программных единиц с определенными свойствами без учета их внутренней структуры. С учетом этого выделены свойства модулей, наиболее важные для процесса сборки:

1) логическая законченность процесса проектирования;
2) заменяемость отдельного модуля в ПС;
3) возврат управления вызывающему модулю;
4) обращение одного модуля к другим;
5) раздельная компиляция модуля как внешнего;
6) использование модуля в различных точках ПС;
7) спецификация паспорта и текста модуля в ЯП.

---

[1] См.: *Лаврищева Е. М., Грищенко В. Н.* Связь разноязыковых модулей в ОС ЕС.

Приведенные свойства позволяют выбрать аксиоматический подход к определению модуля.

Модулем в рамках метода модульного программирования называется программный объект, характеризующийся свойствами 1—7 и реализующий отдельную функцию Это определение задает границы применения понятия «модуль».

Определение типов связей модулей

Под *типом связи* будем понимать: 1) связь по управлению; 2) связь по данным.

1. **Связь по управлению** характеризуется наличием или отсутствием среды ЯП и механизмом вызова.

Среда ЯП определяется как совокупность программных средств окружения модулей с ЯП, включающих:

- системные средства процессов выполнения библиотечных программ связи с ОС, вычисление стандартных функций, обработки внутренних структур данных и т.д.;
- внутренние структуры данных, точки передачи и управления.

Механизм вызова через CALL в ЯП.

Связь по управлению описывается следующей функцией:

$$CP = K_1 + K_2,$$

где K_1 — коэффициент механизма вызова; K_2 — коэффициент перехода от среды ЯП вызывающего модуля к среде ЯП вызываемого.

Для стандартного механизма вызова $K_1 = 1$, для нестандартного $K_1 = 1 + a$ ($a > 0$). Здесь a зависит от количества отличных от стандарта характеристик вызова.

К характеристикам вызова относятся: способ определения точки входа в вызываемый модуль; механизм передачи управления; формирование адреса возврата в вызывающий модуль; доступ к списку параметров; метод сохранения и восстановления регистров вызывающего модуля.

Коэффициент K_2 зависит от количества операций, необходимых для перехода от среды вызывающего модуля к среде вызываемого и наоборот. Аналитического выражения для K_2 не существует, но можно указать параметры, от которых он зависит. К ним относятся: количество библиотечных модулей, входящих в среду; количество выполняемых ими функций; количество структур данных, входящих в среду; ЯП вызывающего и вызываемого модулей.

Если вызывающий и вызываемый модули написаны на одном ЯП, то $K_2 = 0$. Для остальных случаев $K_2 > 0$.

2. **Связь по данным** между модулями. Различают регулярную и нерегулярную связь. Регулярная характеризуется целенаправленным обменом определенного множества данных при каждой активизации вызываемого модуля с помощью оператора вызова CALL. Нерегулярная — непостоянством обмениваемых данных через посредников.

К этому типу связи относится обмен данными с глобальными именами или внешними файлами. Эта связь характеризует *сложность* ПС, описываемую следующей функцией:

$$CI = \sum_{i=1}^{n} K_i F(x_i), \qquad (1.1)$$

где K_i — весовой коэффициент для i-го параметра; $F(x_i)$ — функция количества элементов простых типов для параметра x_i.

Коэффициенты $K_i = 1$ — для простых переменных и $K_i > 1$ — для сложных ТД. $F(x_i) = 1$, если x_i — простая переменная, и $F(x_i) > 1$ — для сложных ТД. Необходимо отметить, что в формулу (1.1) входят данные, принадлежащие к регулярной и нерегулярной информационной связи. Определение простых и сложных ТД будет приведено ниже.

1.2.1. Интерфейсы модулей и их функции

В модульном программировании важное место занимает интерфейс, под которым понимается взаимосвязь программных объектов (модулей, программ, языков и т.п.), обеспечивающая условия их совместного функционирования.

Интерфейс разрабатывается как самостоятельный ПП. В основе построения интерфейсов лежат:

- стандартизация объектов, их свойств и характеристик;
- разработка формализованных правил (механизмов) связи стандартизованных объектов;
- средства автоматизации механизмов связи объектов.

При этом имеет место несколько видов интерфейсов: межъязыковый, межмодульный, межпрограммный, пользовательский.

Межъязыковый интерфейс — это связь различных ЯП по типам содержащихся в них данных, методам их организации и способам отображения этих средств соответствующими системами программирования.

Основными функциями межъязыкового интерфейса является решение следующих четырех задач.

1. Обеспечение перехода от среды функционирования одного ЯП к среде функционирования другого. При связи разноязыковых модулей необходимо осуществить переход от среды ЯП вызывающего модуля к среде ЯП вызываемого. Перед возвратом управления необходимы обратные операции.

2. Передача управления между разноязыковыми модулями. Решение данной задачи связано с решением предыдущей.

Если реализованы средства перехода от одной среды функционирования к другой, то передача управления между разноязыковыми модулями не отличается от передачи управления между одноязыковыми. В этом случае задача для межъязыкового интерфейса сводится к контролю за последовательностью обращения от вызывающего модуля к вызываемому.

Необходимо отметить, что решение данной задачи предполагает механизм непосредственного взаимодействия модулей без дополнительных средств обеспечения передачи управления.

3. Обеспечение доступа к общим данным через механизмы нерегулярной информационной связи в зависимости от места расположения информации. Поэтому при модульном подходе основным механизмом обмена

является аппарат передачи данных через параметры оператора вызова CALL ЯП высокого уровня.

4. Реализация механизма передачи данных через параметры вызова. Задачей межъязыкового интерфейса в этом случае будет преобразование данных из списка фактических параметров вызывающего модуля к представлению, согласующемуся с описанием формальных параметров вызываемого модуля.

Преобразование направлено на устранение различий, как языковых, так и связанных с реализацией данных системами программирования. Оно выполняется перед и после выполнения вызываемого модуля.

Межмодульный интерфейс — это обеспечение связи модулей, записанных в разных ЯП, и управление модульными структурами:

1) разработка межмодульного интерфейса основана на решении задачи сопряжения пар модулей и задачи управления модулями;

2) описание разнородных модулей;

3) представление модулей в системах программирования с ЯП.

1. **Различия** в языковых средствах ЯП являются следствием неодинаковости синтаксического и семантического представления ТД ЯП, их функциональных возможностей. К ним относятся:

- механизм конструирования новых ТД, который отсутствует в языках Фортран, ПЛ/1, Кобол и имеется в языках Паскаль, Ада, Симула-67, Модула-2, Си, CLU;
- некоторые предопределенные типы, которые отсутствуют в некоторых ЯП (например, нет символьного типа в Фортране);
- формат представления разный (логический тип в языке ПЛ/1 представлен как битовая строка);
- динамические ТД отсутствуют в Фортране и Коболе и имеются в языках Паскаль, Ада, Симула-67 и др.;
- организация внешних файлов в ЯП различна;
- дескрипторы для представления структурных ТД имеются в некоторых ЯП и не требуются в Фортране и Коболе;
- представление некоторых структурных типов различается в различных ЯП (массивы в Фортране располагаются по столбцам, в других ЯП — по строкам).

2. **Проблемы сопряжения** связаны с описаниями модулей, вызваны несоответствием задания формальных и фактических параметров, и состоят в следующем:

- описания ТД, областей значений переменных, индексов массивов и т.д. задаются неоднозначно;
- с одним формальным параметром сопоставляется несколько фактических и наоборот;
- изменение порядка следования параметров.

3. **К проблемам реализации** модулей относятся:

- особенности передачи управления (наличие среды функционирования);
- различия во внутреннем представлении однородных ТД для различных СП;
- различия в структуре и организации внешней памяти для однородных файлов.

Из приведенного выше следует, что проблема передачи управления между разноязыковыми модулями носит не принципиальный характер, а является следствием реализации конкретных СП с ЯП. Это подтверждается также тем, что аналогичные проблемы для ЯП, реализованных на мини- и микроЭВМ, практически отсутствуют или незначительны.

Управление построением и обработкой модульных структур программ как одной из задач межпрограммного интерфейса основывается на реализации следующих четырех функций.

1. Комплектование программных средств различного уровня сложности. Этот процесс выполняется несколькими процессами. На каждом приходится иметь дело с программами различного уровня сложности, готовности и отлаженности. Комплексирование (сборка) разнородных программ происходит в рамках задачи межпрограммного интерфейса.

2. Обеспечение построения различных видов программных структур (простая и динамическая структуры). Построение этих программных структур и составляет вторую задачу управления модульными структурами.

3. Анализ правильности функционирования модульной структуры. Для выполнения операций над модульными структурами необходимо исследовать модульные структуры объектов для обеспечения правильности их функционирования. В частности, эти исследования состоят в выполнении операций определения доступности модулей и их именования, анализа циклов в последовательностях обращений между модулями и др. Поэтому в управлении обработкой модульных структур заключается третья задача межпрограммного интерфейса.

4. Проведение отладки и тестирования межмодульных переходов способствует построению модульных структур. При решении этих задач используются средства межъязыкового интерфейса и средства управления модульными структурами. Средства сопряжения модулей позволяют обеспечить тестирование передаваемых параметров, а средства управления модульными структурами — отслеживание цепочек выполнения модулей.

К задачам управления модульными структурами относятся:

1) выполнение операций над модульными структурами;
2) построение новых структур на основе модулей;
3) построение отладочной среды модульной структуры;
4) сборка модулей в агрегат с учетом решения задач 1)—3).

При решении первых трех задач четвертая является тривиальной, и ее обычно выполняет специальный компонент ОС — редактор связей в ОС и др.

1.2.2. Определение модульной структуры ПС

Для представления ПС из модулей используется математический аппарат теории графов, в котором графом G называется пара объектов $G = (X, \Gamma)$, где X — конечное множество, называемое множеством вершин, а Γ — конечное подмножество прямого произведения $X \cdot X \cdot Z$, называемое множеством дуг графа. Из данного определения следует, что граф G фактически является мультиграфом, так как две его вершины могут быть соединены несколькими дугами. Для отличия таких дуг они нумеруются целыми положительными числами.

Программная структура, состоящая из модулей, описывается ориентированным графом. В нем каждая дуга соответствует оператору CALL и соединяет вершину, соответствующую вызывающему модулю, с вершиной, соответствующей вызываемому модулю.

Определение 1.1. Модель модульной структуры ПС — это объект, описываемый тройкой $T = (G, Y, F)$, где $G = (X, \Gamma)$ — ориентированный граф, являющийся графом модульной структуры; Y — множество модулей, входящих в программный агрегат; F — функция соответствия, ставящая каждой вершине X-графа элемент множества Y.

Функция F отображает X на Y:

$$F: X \to Y. \tag{1.2}$$

В общем случае элементу из Y может соответствовать несколько вершин из X, что характерно для динамической структуры ПС.

Определение 1.2. Модульной структурой ПС называется пара $S = (T, \chi)$, где T — модель модульной структуры ПС; χ — характеристическая функция, определенная на множестве вершин X графа модулей G.

Значение функции χ определяется следующим образом:

- $\chi(x) = 1$, если модуль, соответствующий вершине $x \in X$, включен в состав ПС программного агрегата;
- $\chi(x) = 0$, если модуль, соответствующий вершине $x \in X$, не включен в состав ПС и к ней имеются обращения из других модулей.

Определение 1.3. Две модели модульных структур $T_1 = (G_1, Y_1, F_1)$ и $T_2 = (G_2, Y_2, F_2)$ тождественны, если $G_1 = G_2$, $Y_1 = Y_2$, $F_1 = F_2$. Модель T_1 изоморфна модели T_2, если $G_1 = G_2$, между множествами Y_1 и Y_2 существует изоморфизм φ, а для любого $x \in X \Rightarrow F_2(x) = \varphi(F_1(x))$.

Определение 1.4. Две модульные структуры $S_1 = (T_1, \chi_1)$ и $S_2 = (T_2, \chi_2)$ тождественны, если $T_1 = T_2$ и $\chi_1 = \chi_2$. Модульные структуры S_1 и S_2 называются изоморфными, если T_1 изоморфна T_2 и $\chi_1 = \chi_2$.

Понятие изоморфизма модульных структур и их моделей необходимо для введения уровня абстракции, на котором определяются операции над модульными структурами. Для изоморфных объектов операции будут интерпретироваться одинаково без ориентации на конкретный состав модулей. Данные операции определяются над парами (G, χ).

Введенные выше определения описывают модульные структуры общего вида. Далее рассматриваются структуры специального вида, особенности которых состоят в следующем:

1) граф модульной структуры G имеет один или несколько компонентов связности, каждая из которых представляет ациклический граф, т.е. не содержит ориентированных циклов;

2) в каждом компоненте выделена единственная вершина, которая называется корневой и характеризуется тем, что не существует входящих в нее дуг и соответствующий ей модуль программного агрегата выполняется первым (для данного компонента связности);

3) циклы допускаются только для случая, когда соответствующий некоторой вершине модуль имеет рекурсивное обращение к самому себе. Обычно такая возможность реализуется компилятором с соответствующего ЯП и данный тип связи не рассматривается межмодульным интер-

фейсом. Поэтому такие дуги не включаются в граф. Исключение из рассмотрения других типов циклов связано с тем, что некоторые модули должны будут помнить историю своих вызовов, чтобы правильно вернуть управление. А это противоречит свойствам модулей;

4) пустой граф G_0 соответствует пустой модульной структуре.

В дальнейшем под *графом модульной структуры* понимается схема модулей, удовлетворяющая указанным выше условиям. Типичный пример графа приведен на рис. 1.1.

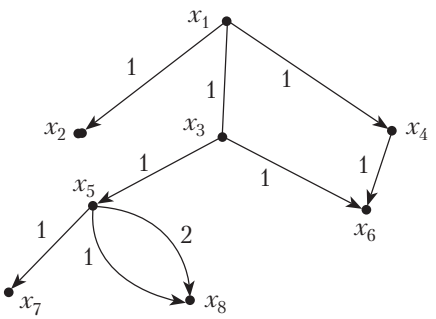

Рис. 1.1. **Пример графа модульной структуры программы**

Вершины x_1, x_2, ..., x_8 составляют множество X. Все дуги пронумерованы. Из модуля, соответствующего вершине x_5, имеются два обращения (два оператора вызова) к модулю, соответствующему вершине x_8. Множество дуг графа имеет вид $\Gamma = \{(x_1, x_2, 1), (x_1, x_3, 1), ..., (x_5, x_8, 1), (x_5, x_8, 2)\}$. В дальнейшем этот граф будет использоваться для иллюстрации операций над модульными структурами. Так, на рис. 1.2 показана структура графа, соответствующая модулям x_5 и x_6, а на рис. 1.3 — модульные структуры для трех видов сегментов.

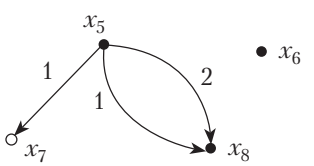

Рис. 1.2. **Граф структуры модулей x_5 и x_6**

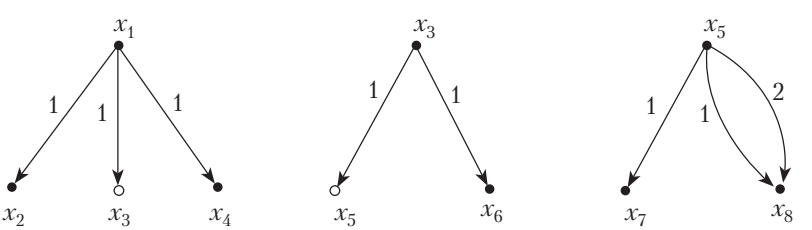

Рис. 1.3. **Графы модульных структур для трех видов сегментов**

1.2.3. Матричное представление графов из модулей

Для определения основных операций над модульными структурами используем матричное представление их графов. Матричные представления используются в различных работах. Например, матрицы вызовов и матрицы достижимости эквивалентны матрицам смежности ориентированных графов.

В настоящей работе в качестве матричного представления используется матрица вызовов. Элемент матрицы m_{ij} определяет количество обращений (операторов вызова) из модуля, соответствующего индексу i, к модулю, соответствующему индексу j. Кроме матрицы вызовов для дальнейшего анализа используется характеристический вектор, для каждого компонента i которого $V_i = \chi(x_i)$. Для графа модульной структуры, приведенной на рис. 1.1, характеристический вектор и матрица вызовов имеют вид

$$V = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Проведем анализ матриц вызовов и характеристических векторов для графов модульных структур, соответствующих различным типам программных агрегатов. Для модулей с графами, представленными на рис. 1.2, векторы и матрицы имеют следующий вид:

$$V_5^m = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, M_5^m = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$V_6^m = (1), M_6^m = (0).$$

Только один элемент характеристического вектора равен единице и только в одной строке матрицы находятся ненулевые элементы. На рис. 1.3 векторы и матрицы записываются в таком виде:

$$V_3^s = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, M_3^s = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; V_1^s = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$M_1^s = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}; V_5^s = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, M_5^s = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Для программы с графом, представленным на рис. 1.1, характеристический вектор и матрица вызовов совпадают с V и M соответственно и определяются в формуле (1.2). Все элементы V равны единице.

В комплексе программ характеристический вектор и матрица вызовов имеют следующий вид:

$$V^c = \begin{pmatrix} V_1^p \\ V_2^p \\ \dots \\ V_n^p \end{pmatrix}, M^c = \begin{pmatrix} M_1^p & 0 & \dots & 0 \\ 0 & M_2^p & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & M_n^p \end{pmatrix}.$$

Здесь V_i^p и M_i^p $(i = \overline{1, n})$ обозначают характеристический вектор и матрицу вызовов для графа i-й программы, входящей в комплекс.

В дальнейшем матричное представление используется для анализа операций над модульными структурами.

1.2.4. Отношение достижимости модулей графов

Пусть $G = (X, \Gamma)$ — граф модульной структуры, x_i, x_j — вершины, принадлежащие X. Если в графе G существует ориентированная цепь от x_i к x_j, то вершина x_j — достижима из вершины x_i. Справедливо следующее утверждение: если вершина x_j достижима из x_i, а x_l — из x_j, то x_l достижима из x_i. Доказательство этого факта очевидно. Рассмотрим бинарное отношение на множестве X, которое определяет достижимость между вершинами графа. Введем обозначение $x_i \to x_j$ для достижимости вершины x_j из x_i. Отношение транзитивно. Обозначим через $D(x_i)$ множество вершин графа G, достижимых из x_i. Тогда равенство

$$\overline{x}_i = \{x_i\} \cup D(x_i)$$

определяет транзитивное замыкание x_i по отношению достижимости.

Докажем следующую теорему.

Теорема 1.1. Для выбранного компонента связности графа модульной структуры любая вершина достижима из корневой, соответствующей данному компоненту, т.е. выполняется равенство (x_l — корневая вершина)

$$\overline{x}_l = \{x_l\} \cup D(x_l) = X.$$

Доказательство. Предположим, вершина x_i $(x_i \in X)$ недостижима из x_1. Тогда $x_i \notin \overline{x}_l$ и множество $X' = X \setminus \overline{x}_l$, непусто. Поскольку выбранный компонент графа связанный, то существуют вершина $x_j \in \overline{x}_1$ и цепь $H(x_i, x_j)$, ведущая от x_i к x_j. Исходя из ацикличности графа G, в X' должна существовать простая цепь $H(x_l, x_j)$, где в вершину x_l не входят дуги (данная цепь может быть пустой, если X' состоит только из x_i). Рассмотрим цепь $H(x_l, x_j) = H(x_l, x_i) \cup H(x_i, x_j)$. Это означает, что модуль x_j достижим из вершин x_1 и x_i и обе вершины не содержат входящих дуг. А это противоречит определению графа модульной структуры с единственной корневой вершиной. Теорема доказана.

Основное требование для обеспечение достижимости — это отсутствие неориентированных циклов в графе. Исходя из графа на рис. 1.4, отмечаем, что граф содержит ориентированный цикл и модули, соответствующие вершинам x_4, x_5, x_6, никогда выполняться не будут. Таким образом, результаты теоремы 1.1 усиливают требование отсутствия ориентированных циклов в графе модулей.

Для анализа матричного представления отношения достижимости графа модульной структуры рассмотрим граф матрицы достижимости, приведенной на рис. 1.1.

Коэффициент $a_{ij} = 1$, если модуль, соответствующий индексу l, достижим из модуля, соответствующего индексу i. Следующие результаты основаны на известной теореме из теории графов.

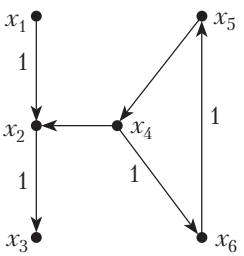

Рис. 1.4. **Граф с ориентированным циклом**

$$A = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Теорема 1.2. Коэффициент m_{ij} l-й степени матрицы смежности M определяет количество различных маршрутов, содержащих l дуг и связывающих вершину x_i с вершиной x_j-ориентированного графа.

Рассмотрим следующие три следствия из этой теоремы[1].

Следствие 1.1. Матрица $\bar{M} = \sum_{l=1}^{n} M^t$, где M — матрица смежности ориентированного графа с n вершинами, совпадает с точностью до числовых значений коэффициентов с матрицей достижимости A.

[1] Доказательство этой теоремы приводится в статье: *Лаврищева Е. М., Грищенко В. Н.* Сборочное программирование.