ИНТУИТ
НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 10. Инженерия приложений и предметной области

Одна из характерных черт инженерной деятельности в промышленности – использование готовых решений и деталей. В программировании промышленное использование готовых решений и программных продуктов еще не стало повседневной практикой, а сформировались признаки этой инженерной деятельности [9.1,9.3].

Исследования и разработки в области инженерии программирования в направлении повторного использования компонентов (ПИК), готовых для применения в других областях человеческой деятельности привели к тому, что сформировалось два направления применения готовых ПИК [9.1][9.2] 9.3], [9.5]:

1. *прикладная инженерия* (*application engineering*) – процесс производства конкретных новых приложений из ПИК (модулей, программ, подпрограмм и др.), ранее созданных самостоятельно либо в среде программной системы или как отдельные элементы многоразового использования в инженерии другой ПрО;
2. *инженерия ПрО* (*domain engineering*) включает методы разработки, поиска, классификации, адаптации, сбора ПИК и создания из них или из готовых частей систем семейства домена, которые сохраняют наработанный опыт по реализации одного домена для применения его в другом крупном домене. Необходимое условие этой инженерии – системные инструментальные средства поддержки методов накопления ПИК и внедрения их в новые подсистемы семейства или самого домена.

Первое направление фактически характеризует создание одиночных ПС из разного рода ПИК, а второе ставит задачу создания программных систем домена и их совокупностей с выделением отдельных частей ПС при проектировании, обладающими общими свойствами и характеристиками и способными к многоразовому использованию в других доменах этой совокупности.

9.1. Инженерия повторного использования компонентов

Инженерия повторного использования компонентов (ПИК) – это систематическая и целенаправленная *деятельность по* подбору реализованных программных артефактов и представленных в виде ПИК, анализу их функций для добавления в качестве готовых в проектируемую систему и их *интеграция* с другими компонентами. Согласно стандарту ISO/IEC-12207 эта *деятельность* классифицируется как организационная и планируемая инженерная *деятельность*, которая заключается в выявлении общих и специфических черт компонентов для *принятия решений* об их использовании в разработке новых ПС [9.1][9.2][9.3], [9.4, 9.6-9.8].

При этом предполагается, что имеется каталог, с помощью которого можно понять, какие ПИК, как готовые детали, имеются и как их можно соединить в программную конструкцию. Именно эта сторона характеризует повторное использование как систематическую и целенаправленную *деятельность по* созданию и использованию каталога ПИК.

Систематическое повторное использование – это капиталоемкий подход, который предусматривает наличие двух процессов в ЖЦ разработки ПС.

Первый процесс – это создание ПИК путем:

- изучения спектра решаемых задач ПрО, выявление среди них общих свойств и функций;
- построения компонентов, реализующих выявленные функции в виде ПИК;
- разработка каталога для хранения изготовленных компонентов и организации поиск необходимых компонентов по запросам пользователей.

Для успешной реализации данного процесса необходимо иметь определенный *опыт* в решении нескольких подобных между собой задач, позволяющий определить заложенные общие черты и различия, чтобы найти решение для их реализации, а также разработать приемы настройки на характерные для каждой задачи особенности.

Второй процесс – конструирование новых систем из готовых компонентов путем:

- понимания сущности новой системы (домена), определения целей ее создания и предъявляемых к ней требований;
- поиска в каталоге готовых компонентов, которые кажутся подходящими для их использования в новой системе;
- сопоставления цели новой разработки с возможностями найденных ПИК и принятия решений о целесообразности и месте их применения в системе;
- интеграция ПИК в новую разработку с обеспечением интерфейса с подсистемами и другими компонентами.

Первый процесс требует вложения капитала, второй – получение прибыли за счет экономии трудозатрат от применения готовыхПИК. Инвестиции в повторное использование требуют оценки эффективности вложения капитала, прогнозирования сроков и объемов возврата этого вложения, оценки рисков и др. Бизнес повторного использования, как любой бизнес, требует специальных условий *по* менеджменту всей инженерной деятельности инженерии систем из ПИК. Критерии успеха такого бизнеса определяются следующими предпосылками:

1. повторное использование готовых компонентов требует меньших трудозатрат, чем разработка их как новых разовых продуктов;
2. поиск пригодных для использования компонентов требует меньше усилий, чем произвести реализацию необходимых функций для целей проектируемой системы;
3. настройка компонентов на новые условия среды применения должна обеспечиваться меньшими трудозатратами, чем новая разработка.

Основная *парадигма* ПИК – "писать – один раз, выполнять – много раз, где угодно". *Архитектура*, в которую встраивается готовый ПИК, поддерживает стандартные *механизмы* для работы с компонентами как со строительными блоками. Чтобы обеспечить высокий уровень использования ПИК, они должны обладать такими основными свойствами: функциональность, *удобство использования* и качество реализации.

Разновидности ПИК. В качестве ПИК могут использоваться формализованные артефакты деятельности разработчиков ПС, которые отражают некоторую функциональность для применения в новых разработках. Под *артефактом* понимается реальная порция информации, которая может создаваться, изменяться и использоваться при выполнении деятельности, связанной с разработкой ПС различного назначения.

Артефактами могут быть:

промежуточные продукты процесса разработки ПС (требования, постановки задач, архитектура и др.);

описания результатов процесса разработки ПС (спецификация, модели, каркас и т.п.)

готовые компоненты ПС или отдельные части системы;

продукции, фреймы, диаграммы, паттерны и т.п.

К компонентам ПИК выдвигаются такие требования, как независимость от конкретной платформы, наличие стандартного интерфейса и параметров настройки на новую среду, возможность их взаимодействия в системе без внесения в них изменений.

Разработке ПС с помощью ПИК соответствует модель ЖЦ со следующими общими этапами:

анализ объектов и отношений реализуемой ПрО для выявления ПИК, обладающих общими свойствами, присущими группам объектов этой области;

адаптация имеющихся в базе репозитария ПИК, разработка новых функциональных компонентов, не представленных в этой базе и доведение их до уровня ПИК;

разработка интерфейсов компонентов и их размещение в репозитарии интерфейсов системы;

интеграция ПИК и их интерфейсов с другими элементами создаваемой системы и формирование конфигурации этой системы.

Повторные компоненты могут быть прикладными и общесистемными. *Прикладные компоненты* выполняют отдельные задачи и функции прикладной области деятельности домена (бизнесдомены, коммерция, экономика и т.п.), которые могут использоваться в дальнейшем, как готовые в качестве прикладных систем в других доменах с аналогичными функциями.

К *общесистемным компонентам* относятся компоненты общего и универсальные назначения, а также общесистемные сервисные средства, которые обеспечивают системное обслуживание и предоставляют разные виды сервисов для многих создаваемых программных систем разного назначения. К компонентам общего назначения относятся: трансляторы, редакторы тестов, системы генерации, интеграции, загрузчики и др. Они используются всеми прикладными системами в процессе их проектирования и выполнения. Универсальные системные компоненты обеспечивают функционирование любых (в том числе и прикладных) компонентов, *обмен данными* и передачу сообщений между всеми видами систем и компонентов, расположенных в разных средах и платформах компьютеров. К ним относятся ОС, *СУБД*, сетевое обеспечение, электронная *почта* и др.

Связь между прикладными и общесистемными средствами осуществляется через стандартные интерфейсы, обеспечивающие взаимодействие разных типов компонентов через *механизмы* передачи данных и сообщений.

На современном рынке программных продуктов циркулируют следующие виды готовых компонентов:

процедуры и функции на ЯП высокого уровня;- алгоритмы, программы;

классы объектов и абстрактные классы;

структуры данных и часто используемая информация (например, информационные ресурсы Интернет);

API, IDL модули в библиотеках C++ (например, GUI, графика и др.);

Web-ресурсы и сервисы;

средства развертки систем и компонентов в операционной среде (например, CORBA, COM, .NET) [9.6];

готовые решения в виде абстракций – паттерны, фреймы и др.

Все многообразие видов и типов готовых компонентов требует от разработчиков их поиска и изучения для использования в новых ПС. Процесс разработки новых ПС с помощью разных видов ПИК – капиталоемкий, в нем в качестве капитала выступают готовые ПИК, на применение которых затрачивается меньше средств, чем на повторную их разработку.

9.2. Спецификация ПИК

В качестве ПИК могут быть объекты, созданные в рамках объектно-ориентированного программирования (например, огромная библиотека повторно используемых классов в C++) с наследованием их реализации. В компонентном программировании наследуется реализация компонента и его интерфейсы.

Пример ПИК – *контейнерные классы*, которые хранят структуры данных с правилами их запоминания или выдачи очередного элемента, входящего в *контейнер*. Механизм контейнеров реализован в C++ в виде так называемых шаблонов (*templates*) и их библиотек ПИК этого типа.

Каждый *компонент* имеет такие аспекты и свойства:

связь компонентов через интерфейсы на этапах разработки системы;

инкапсуляция компонента, как "черный ящик" без возможности вмешательства в выходной код);

наследование интерфейсов, их изменение и настройка на применение;

повторное использование исходного или выходного кода.

С общей точки зрения *компонент* определяется поразному в зависимости от среды и функций, приведем некоторые их них [9.1, 9.2].

Определение 1. *ПИК* – это некоторая *функция* с определенными атрибутами, обеспечивающая функциональность, взаимодействие со средой и поведение.

Определение 2. *Готовый ПИК* – это совокупность методов определенной сигнатуры и типов данных, которые передаются и возвращаются после выполнения метода.

Определение 3. *Компонент типа ПИК* – это самостоятельный программный элемент, который удовлетворяет определенным функциональным требованиям, требованиям архитектуры, структуры и организации взаимодействия в заданной среде, имеет спецификацию, помогающую пользователю его использовать и объединять с другими компонентами в интегрированную систему.

Определение 4. *Программный компонент* – это независимый от языка программирования, самостоятельно реализованный *объект*, который обеспечивает выполнение определенной совокупности прикладных сервисов, *доступ* к которым возможен только с помощью интерфейсов, указывающих функции и порядок обращения к нему операций.

Эти и другие определения приведены в публикациях, они отображают разные тонкости определения или использования компонента. Мы остановимся на двух последних определениях компонента, модель спецификации которого имеет следующий вид:

$$\text{ПИК} = (T, I, F, R, S),$$

где T – тип компонента,

I – множество интерфейсов компонента;

F – функциональность компонента;

R – реализация, скрытая часть – программный код;

S – сервис для взаимодействия со средой или набор правил развертывания.

Каждый из элементов спецификации компонента представляет собою видимую или скрытую от пользователя часть его абстракции.

В зависимости от сложности ПИК их можно разделить на следующие группы:

простые компоненты (функция, модуль, класс и пр.);

объектыкомпоненты, имеющие интерфейс, функцию и реализацию на любом ЯП, а также возможность дополнять спецификации шаблоном развертывания и интеграции;

готовые к использованию ПИК (например, beans компоненты в Java, *AWT* компоненты, классы и др.);

сложные ПИК типа каркасов, паттернов с элементами группирования нескольких простых ПИК в законченную функциональность и их взаимодействие между собой при решении одной общей задачи системы.

Большое количество готовых компонентов требует от разработчиков и пользователей задания метаинформации о том, какие классы совместимы с заведомо определенными семантическими ограничениями спецификации ПИК. Метаинформация включает информацию, касающуюся сведений относительно:

интерфейсов, которые реализуют компоненты;

механизмов повторного использования;

среды развертывания компонента;

сервиса, поддерживаемого компонентом;

ролей, которые выполняют компоненты в системе;

формализованных языков описания ПИК.

Современная технология применения ПИК базируется на таких особенностях:

отображение способности ПИК анализировать самого себя и предоставлять свои возможности динамично во время выполнения, а не во время компиляции, т.е. управлять большинством свойств, событий и методов, встроенных в компонент;

стандартизованное описание компонента для удобного анализа и понимания его посторонним лицом, а не самим разработчиком;

способности обеспечивать изменения в нем, не затрагивая целевую направленность и добавления новых параметров;

способности компонента к рефакторингу, т.е. к трансформации компонента с сохранением функциональности, но с возможным изменением структуры и исходного кода;

сохранение параметров конфигурации (шаблонов отладки) в постоянной памяти для использования в нужное время;

регистрация сообщений о событиях, полученных от других объектов через ссылки (например, beans компоненты и инструментарий поддержки архива в технологии JAVA), сообщения, а также группирование компонентов в JAR файле для дальнейшего повторного использования;

использование компонента в разных языковых средах;

адаптация ПИК к разным контекстам их использования и выделение свойств, которые мешают повторному использованию и модификации для применения в конкретных целях.Компоненты в отличие от объектов могут изменяться и пополняться новыми функциями и интерфейсами. Они конструируются в виде некоторой программной абстракции, состоящей из трех частей: информационной, внешней и внутренней.

Информационная часть содержит такие сведения: назначение, дата изготовления, условия применения (ОС, платформа и т.п.), возможности ПИК, тип *среды окружения* и др.

Внешняя часть – это *интерфейс*, который определяет взаимодействие компонента с внешней средой и с платформой, на которой он будет выполняться и включает характеристики типа:

интероперабельность – способность взаимодействовать с компонентами других сред;

переносимость – способность компонента выполняться на разных платформах компьютеров;

интеграционность – объединение компонентов с помощью интерфейсов в более сложные структуры ПС;

нефункциональность – требование к безопасности, надежности и защиты компонентов и данных.

Внутренняя часть компонента – это программный фрагмент кода, системная или абстрактная структура, представленные в виде каркаса компонента, спецификации и выходного кода (табл. 9.1). Данная часть компонента состоит из полей:

интерфейса (interfaces);

реализации (implementation);

схемы развертки (deployment).

Интерфейс – компонента содержит обращения к другим компонентам через описание параметров средствами языков *IDL* или *APL*. В нем указываются типы данных и *операции* передачи параметров для взаимодействия компонентов друг с другом. Каждый *компонент* может реализовывать целую совокупность интерфейсов. *Интерфейс* – видимая неизменная и обязательная часть спецификации компонента. Например, система Inspector Components изменяет некоторые параметры интерфейса компонента без вмешательства в его код.

Параметры интерфейса определяются типом ПИК и включают *инвариант* спецификации с указанием типа и имени компонента, его входных и выходных параметров, методов компонента и др. В языке *JAVA*, например, типами компонентов могут быть: проекты, формы (*AWT*-компоненты), beans-компоненты, CORBA-компоненты,

RMI-компоненты, стандартные классы-оболочки, *БД*, JSP-компоненты, сервелети, XML-документы, DTD-документы и т.п.

Таблица 9.1. Структурные части компонента

Свойства элементов структуры компонента		
Интерфейс	**Реализация**	**Схемы развертывания**
Один или несколько	Одна или несколько	Типовость процедуры развертывания
Уникальность именования	Ориентация на платформу и среду	Конфигурация
Клиентский или серверныйй	Выбор конкретной реализации.	Управляемость
Сигнатура операции	Поддержка интерфейсов компонента	Настраивание на операционную среду
Методы взаимодействия		Модификация

Реализация – это код, который будет использоваться при обращении к операциям, определенных в интерфейсах компонента. *Компонент* может иметь несколько реализаций, например, в зависимости от операционной среды или от модели данных и соответствующей системы управления базами данных, которая необходима для функционирования компонента.

Развертка – это физический *файл* или *архив*, готовый к выполнению, который передается пользователю и содержит все необходимые *операции* и инструкции для создания, настройки и функционирования компонента.

Компонент описывается в ЯП, которое не зависит от операционной среды (например, виртуальной машины *JAVA*) и от реальной платформы (например, в системе *CORBA*), где он будет функционировать.

Расширение понятия компонента есть *паттерн – абстракция*, которая содержит описание взаимодействия совокупности объектов в общей кооперативной деятельности, для которой определены роли участников и их ответственность. *Паттерн* – повторяемая часть программного компонента, его схема или взаимосвязь контекста описания пути решения проблемы.

9.3. Репозитарий компонентов

ПИК и другие компоненты многоразового применения размещаются в разных хранилищах.

Ими могут быть библиотеки, репозитарии компонентов и ресурсов в *Интернет* (например, GreedStone, Matlab, библиотека повторно используемых классов C++ и др.).

Ими можно пользоваться многократно всем желающим для реализации своих целей, в том числе при построении программных систем различного назначения. Репозитарий типа библиотеки GreenStone и Matlab представляют огромное множество готовых программ научного и математического типов, т.е они ориентированы на математиков, биологов, физиков и других *специалистов предметных областей*.

В общем случае репозитарий – это система средств для хранения, пополнения наработанных ПИК, включает инфраструктуру разработки ПС из компонентов, организацию доступа к содержащимся в нем ПИК для последующего их применения в новых проектах.

С функциональной точки зрения репозитарий работает *по* принципу информационно-поисковой системы, объектами хранения которой – разные типы документов, текстов и др. Система ставит в соответствие ключевым понятиям, словам, правилам доступа и др. *запрос* пользователя к требуемому ресурсу, который содержится в ее коллекции документов.

В отличие от таких систем, в репозитарии компонентов, кроме ПИК размещается семантическая *информация* в виде поискового образа, созданного на основе описания информационной модели каждого компонента. Эта модель – средство построения поискового образа для каталога ПИК, ориентированного на осмысление человеком функций ПИК и возможности сопоставления их с собственными потребностями.

Поисковый образ ПИК в репозитарии может включать:

- список ключевых слов, наиболее часто упоминаемых в тексте ПИК;
- ссылку на предварительно построенную онтологию домена проблемной области, к которой этот ПИК относится.

Задание поискового образа ПИК на основе информационной его модели обеспечивает систему хранения, поиска и сопоставления ПИК. Репозитарий виртуально разделен на *разделы*, соответствующие отдельным

ПрО, перечень которых представляет *классификатор* первого уровня. Классификаторами следующих уровней могут быть отдельные понятия ПрО.

Поисковый образ может включать паспортные данные компонента (имя и *адрес* разработчика, способ приобретения, цену и т.п.), сведения о среде реализации (ОС, ЯП, *СУБД* и т.п.), описание аппаратных ресурсов, имени ПрО в *системе классификации*, а также описание нефункциональных требований к создаваемой системе (*безопасность*, *конфиденциальность*, *показатели качества* системы и др.).

Для отображения ПИК в репозитарии проводится их классификация и каталогизация, аннотирование и *размещение*.

Классификация компонентов ориентирована на унификацию представления информации о компонентах для последующего их поиска и отбора из среды репозитария. Она проводится с учетом следующих факторов:

- компонент типа модуль, класс и т.п.;
- компоненты, которые имеют интерфейс (входные и выходные параметры, условия выполнения), функциональность и реализацию на ЯП со спецификацией шаблона развертывания;
- готовые к употреблению ПИК;
- сложные ПИК типа каркасы и паттерны, которые обеспечивают взаимодействие простых ПИК.

Каталогизация направлена на физическое *размещение* кодов ПИК в репозитарии для извлечения их при необходимости *встраивания* нового программного проекта ПС. Для выбранных компонентов осуществляется их настройка на условия среды функционирования.

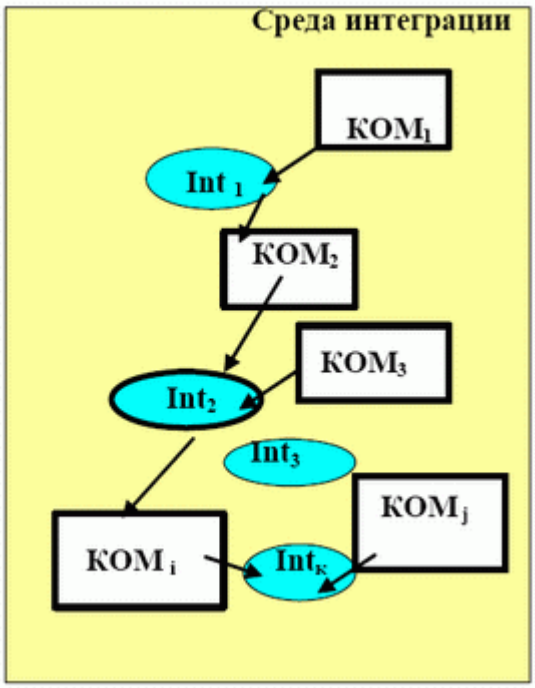
Поисковый образ упрощает *поиск* и сокращает сроки разработки ПС за счет:

- отображения базовых функций и понятий компонента;
- скрытия представления данных, операций обновления и получения доступа к этим данным;
- обработки исключительных ситуаций, возникающих в процессе выполнения и др.

Разработка новых ПС может осуществляться *по* технологии с использованием компонентов $(KOM_1, ..., KOM_k)$ и ПИК (рис. 9.2).

Если компоненты написаны на разных ЯП, в процессе создания ПС формируются интерфейсные модули $(Int_1, ..., Int_k)$, в которых преобразуются типы передаваемых данных между компонентами.

1. Разработка компонентов (КОМ) на ЯП
2. Выбор ПИК
3. Разработка интерфейсов (Int) для КОМ и ПИК
4. Генерация интерфейсов пары КОМ на ЯП
5. Разработка среды и репозитария КОМ
6. Типизация и классификация компонентов
7. Тестирование КОМ, интерфейсов, ПС
8. Интеграция разных видов компонентов
9. Загрузка ПС в среде выполнения
10. Сопровождение компонентной ПС
11. Эволюция компонентной ПС



увеличить изображение

Рис. 9.2. Технология инженерии компонентов в разработке ПС

В соответствии с терминологией *UML* лица, которые обеспечивают выбор необходимых ПИК из репозитария и построение из них новых ПС *по* типу сценария – это акторы.

9.4. Язык описания интерфейса компонентов в распределенной среде

Для объединения компонентов программную структуру необходимым условием является наличие для них формально определенных интерфейсов в языках *IDL* и *APL*, а также механизмов динамического контроля

связей между компонентами.

Спецификация интерфейса в *API* и *IDL* включает описание функциональных свойств компонентов, их типов и порядка задания операций передачи аргументов и результатов для взаимодействия компонентов. Описание интерфейса представляет собой интерфейсный посредник между двумя объектами.

ПС, построенная из компонентов и предназначенная для функционирования в распределенной среде, имеет некоторые особенности в структуре, а именно: она состоит из двух частей, каждая из которых может выполняться на разных процессах и взаимодействовать друг с другом через вызов интерфейсных функций. Первая часть – серверная *программа*, а вторая – клиентская (далее просто *сервер* и клиент).

В функции интерфейсного модуля клиента входят:

- подготовка внешних данных клиента (параметров),
- набор вызовов этих процедур или обращение к сервису сервера,
- обработка разных ошибок, возврат данных от сервера к клиенту.

Общие функции интерфейсного модуля сервера содержат: – ожидание сообщений клиента и их обработку; *запуск* удаленной процедуры и передачу ей параметров клиента;

возврат результатов процедуры клиенту, уничтожения уда ленной процедуры и др.

Структура интерфейсного модуля не зависит от ЯП взаимодействующих объектов и в целом одинакова для всех. Это связано со стандартизированной его структурой и общим *языком спецификации* интерфейса, *синтаксис* которого представлен ниже в форме БэкусаНаура:

```
<интерфейс объекта>::=
    object <имя_Объекта> :{<множество исходных интерфейсов>};
                         {<множество входных интерфейсов>}
    end;
<множество входных интерфейсов>::= <множество интерфейсов>;
<множество выходных интерфейсов>::= <множество интерфейсов>;
<множество интерфейсов>::= ∅ | <интерфейс>; <множество интерфейсов>;
<интерфейс>::=
    interface <имя_интерфейса> :{<множество функций>}
    end;
<множество функций>::= ∅ | <функция>; <множество функций>;
<функция>::=
    function <имя_функции>: <множество параметров>
    end;
<множество параметров>::= <параметр> | <параметр>, < множество параметров>;
<параметр>::= <тип> (<вид параметра>);
<вид параметра>::= in | out | inout.
```

Тип данных описывается средствами языков программирования (C++, *Pascal* и т.п.) и обеспечивает взаимодействие между процессами, а параметры `<вида параметра>` могут быть: `in` –*входной параметр*, `out` – *выходной параметр*, `in-out` – совместный *параметр*.

9.5. Основные аспекты инженерии приложений и предметной области

Базис инженерии программирования, основанной на использовании ПИК, – прикладная инженерия и инженерия ПрО, которые используют готовые ПИК, программы, а также отдельные части систем многоразового применения [9.3, 9.4, 9.10].

Прикладная инженерия – это инженерия ПИК и процесс создания ПС из готовых компонентов и ПИК.

Инженерия ПрО ориентирована на создание архитектуры ПрО – каркаса (framework), включающего ПИК, компоненты многоразового применения из семейства программ разных доменов и их интерфейсов, а также готовые приложения прикладной инженерии.

Основные этапы *инженерии ПрО* это:

- анализ ПрО и выявление объектов и отношений между ними;
- определение области действий объектов ПрО;
- определение общих функциональных и изменяемых характеристик, построение модели характеристик, устанавливающей зависимость между различными членами семейства, а также в пределах членов семейства системы;

создание базиса для производства конкретных программных членов семейства с механизмами изменчивости независимо от средств их реализации;

подбор и подготовка готовых компонентов многократного применения;

описание аспектов выполнения задач ПрО;

генерация отдельного домена, члена семейства и ПС в целом.

В основе генерации модели ПрО для семейства ПС лежит модель характеристик и набор компонентов реализации задач ПрО. Используя данную модель, знания о конфигурациях и спецификации компонентов участвующих в этом процессе, можно автоматизировано сгенерировать отдельный член семейства, а также *ПО* для всей ПрО.

Инженерия ПрО включает в себя следующие вспомогательные процессы:

корректировку процессов для разработки решений на основе ПИК;- моделирование изменчивости и *зависимостей компонентов* многоразового использования, фиксации их в модели характеристик и в справочнике информации об изменении моделей (объектных, Use Case и др.).

Фиксация зависимостей между характеристиками модели избавляет разработчиков от некоторых конфигурационных операций, выполняемых, как правило, вручную;

разработку инфраструктуры ПИК – описание, хранение, поиск, оценивание и объединение готовых ПИК;

создание репозитария ПИК и компонентов многоразового использования в классе задач ПрО (рис. 9.3);

обеспечение безопасности, защиты данных, изменений;

обеспечение синхронизации и взаимодействия ПИК.

увеличить изображение

Рис. 9.3. Структура репозитария в интегрированной среде ПрО

Стандартизация процессов доменной инженерии. В стандарте ISO/IEC 12207: 2002 дано описание процесса доменной инженерии (*Domain engineering process*) как нового процесса в модели процессов ЖЦ. Согласно этому стандарту процесс доменной инженерии охватывает ряд видов деятельности.

Анализ домена состоит:

в определении границ домена и связей с другими доменами;

в выявлении, формальном описании общностей и отличительных особенностей внутри домена (постоянных и переменных требований) и в их включении в модели домена;

в формировании словарей для описания основных понятий в домене и взаимосвязей между активами в домене;

в классификации и документировании моделей;

в *оценке моделей* и словарей домена с учетом выбранной методологией моделирования. *Архитектурное проектирование домена* (*Domain* design) – это *определение* архитектуры домена на основе программных компонентов – специфичных активов/ресурсов.

Архитектура домена – каркас для ПИК, активов и формально определенных интерфейсов – должна согласовываться с моделью домена, стандартами организации и оцениваться на соответствие выбранной методологии архитектурного проектирования.

Технология доменной инженерии базируется на новом процессе в модели ЖЦ (ISO/IEC 12207) и включает стандартизированные подпроцессы:

формирование ресурсов (Asset *provision*), т.е. разработка или приобретение ресурсов (активов), которые могут использоваться при компоновки новых программных систем или подсистем.

разработка базы ресурсов (asset-based development), в основе которой лежит концепция повторного использования (*software reuse*) – ПИК, обеспечивающая компоновку программных продуктов домена;

сопровождение ресурсов (Asset maintenance) – модификация и эволюция модели, архитектуры и продуктов домена за счет готовых ресурсов типа ПИК.

Данная технология предполагает разработку методик и инструментов для эффективного ее выполнения, а также для генерации системы из ПИК и компонентов многоразового применения на основе спецификаций требований к системе.

В результате применения технологии доменной инженерии в софтверной организации будет создаваться, поддерживаться и развиваться *архитектурный базис* из *множества* ПИК, хранящийся в репозитарии и учитывающий общие и специфические особенности разных сторон деятельности в доменах.

Основное требование к инженерии ПрО – обеспечение многоразового применения используемых решений для семейства ПС, а в инженерии приложений – производство (линейка) одиночной системы из ПИК *по* требованиям к ней.

9.6. Структура линейки программных продуктов

На базе использования готовых к употреблению ПИК в рамках инженерии ПрО формулируются требования к семейству систем, причем инвариантные свойства, присущие каждому члену ПИК, отделены от свойств, специфических для отдельных представителей семейства готовых компонентов.

Свойства ПИК могут быть обязательные, необязательные и *альтернативные*. К обязательным свойствам относятся такие, которые обязательно присутствуют в каждом из представителей семейства систем, а их реализация может иметь некоторые отличия. К альтернативным свойствам относятся свойства, которые отображают особенности выбора представителя семейства как многократно используемого. Необязательные свойства отражают некоторые специфические особенности или могут отсутствовать.

Создание семейства программных продуктов в инженерии ПрО проводится на определенной платформе, включающей в себя архитектурные особенности и совокупность готовых компонентов и программ для генерации отдельных членов семейства ПС. Такими готовыми для применения ресурсами могут быть не только компоненты, ПИК и готовые программы семейства, но и определенные правила спецификации требований и архитектуры системы.

Американский институт программной инженерии *SEI* предложил *линию продуктов* или линию семейства продуктов как производство систем из *множества* программ, ПИК и ПС, которые удовлетворяют специфическим потребностям некоторого рынка программной продукции и показателям качества. В результате множество компонентов и систем образуют семейство продуктов, если они имеют общие свойства, а каждый член семейства имеет свои индивидуальные свойства [9.10].

Понятие линейки программных продуктов (Framework for Product *Line Practice*) сформировалось как *поддержка* инженерии ПрО, в задачу которой входит применение подходов и методов для автоматизированного построения разных видов программных продуктов на линейке продуктов.

При этом в рамках домена исследуются рынок и потребности покупателей, строится производственный план, процессы и определяется организация их взаимодействия. На основе анализа потребностей рынка строится технология линии продукта, в которую включаются методы разработки, тестирования и *оценки процессов* и продуктов линейки.

Институт *SEI* предложил также инфраструктуру разработки линейки продуктов, в которую входят различные методы и средства программной инженерии, необходимые для построения и эксплуатации линеек продуктов (рис. 9.4) [9.10], а также определены соответствующие руководящие материалы и методики.

Рис. 9.4. Инфраструктура построения линейки продукта

Построение конкретной линейки для разработки программного продукта для некоторого представителя (члена) семейства домена определяется:

ограничениями, свойственными продуктам линейки;

образцами и каркасами, которые могут использоваться на ЛиНИИ;

производственными ограничениями, стратегиями и методами;

набором средств и инструментов для разработки продукта на линии.

На основе этих данных определяются *область действия* линейки и набор базовых средств, строится план создания продукта на линейке, который учитывает сроки, *стоимость* и требования к управлению производством продукта путем:

- контроля плана работ и отслеживания хода построения продукта;
- выявления рисков и управления ими в процессе исполнительской деятельности на процессе проектирования семейства;
- прогнозирования стоимостных и технических ресурсов проекта;
- применения технологии управления конфигурацией;
- измерения и оценки качества продукта.

Данное направление развивается за счет использования готовых компонентов, накапливаемых в репозитариях доменов в целях выбора готовых функций, компонентов ПИК для использования на производственной линейке.

9.7. Оценивание стоимости системы из компонентов

Инженерия программирования ПС из компонентов, которые вновь разрабатываются или берутся готовые, основа на ПИК и компонентах многоразового использования. Она включает оценку стоимости новой разработки с целью получения вложенных затрат на создание продукта из совокупности взаимосвязанных компонентов, реализующих функции ПрО.

Общую *стоимость* создания компонентной системы считаем состоящей из таких составных элементов:

$$C = C_1 + C_2 + C_3 + C_4,$$

где C_1 – *стоимость* анализа функций ПрО;

C_2 – *стоимость* подбора ПИК из репозитария или библиотеки методов с учетом вновь разработанных компонентов;

C_3 – *стоимость* интеграции всех компонентов в систему;

C_4 – *стоимость* определения и обработки данных ПС.

Рассмотрим в отдельности каждую составную единицу стоимости ПС.

Стоимость анализа функций ПрО имеет вид

$$C_1 = \sum_{i=1}^{M} b_i^1 C_{1i} F_i(D_i)$$

где D_i – данные i –функции, M – количество функций F в системе,

$$b_i^1 = \begin{cases} 1, \text{ когда функция реализована компонентом,} \\ 0, \text{ в противном случае.} \end{cases}$$

Стоимость поиска и исследования возможностей применения ПИК из репозитария для реализации некоторой определенной функции ПрО вычисляется с помощью выражения

$$C_2 = \sum_{j=1}^{N} \sum_{i=1}^{M} a_{ji}^2 C_2(F_{ji}) + C_2(PF_{ji})$$

где $C_2(F_{ji})$ – *стоимость* поиска ПИК для функции F_i, сформулированной на этапе анализа ПрО, N – количество новых компонентов и ПИК, $C_2(PF_{ji})$ – *стоимость* разработки некоторых типичных программных компонентов,

$$a_{ji}^2 = \begin{cases} 1, \text{ когда } j\text{-компонент используется функцией } F_i \\ 0, \text{ в противном случае.} \end{cases}$$

Стоимость композиции компонентов определяется так:

$$C_3 = \sum_{j=1}^{N} \sum_{i=1}^{M} \sum_{r=1}^{R} d_{jir}^2 C_3(I_{jr})$$

где $C_3(I_{jr})$ – *стоимость* создания интерфейсных модулей пары компонентов K_i и K_r,

$$d_{jir}^2 = \begin{cases} 1, & \text{когда } rR \text{ соответствует количеству параметров} \\ & X = (X_1, \ldots, X_r)\text{-входных для } j\text{-компонента}, \\ & r\text{-функции } (r = 1, \ldots, R) \\ 0, & \text{в противном случае.} \end{cases}$$

Таким образом, конечный результат оценивания стоимости ПС получается путем суммирования $C = C_1 + C_2 + C_3 + C_4$ (расчет C_4 громоздкий, поэтому не приводится) и имеет вид:

$$C = \begin{cases} \sum_{i=1}^{M} b_i^1 C_{1i} F_i(D_i) + \sum_{j=1}^{N} \sum_{i=1}^{M} a_{ji}^2 C_2(F_{ji}) + C_2(PF_{ji}) + \\ + \sum_{j=1}^{N} \sum_{i=1}^{M} \sum_{r=1}^{R} d_{jir}^2 C_3(I_{jr}) \end{cases}$$

Основное ограничение данного выражения – это факт реализации заданных функций в ПС, наличие средств интеграции пар компонентов K_i и K_r, которые могут быть заданы в любых современных ЯП в заданной среде функционирования, количество компонентов R соответствует заданным функциям для решения задач ПрО.

Расчет стоимости для компонентных систем является трудоемким процессом. Общая *стоимость* уменьшается, если описание компонентов выполнено на одном из ЯП, за счет отсутствия интерфейсных модулей преобразования данных в системе.

Таким образом, *программная инженерия* характеризуется степенью использования накопленной программной продукции в виде ПИК и компонентов ПрО многоразового использования. Инженерия предполагает не только их подбор для применения в новых разработках ПС, но соответствующие инженерные оценки показателей качества, стоимости и риск приобретения ПИК. Определяются трудозатраты на разработку с учетом полученных выгод (а также потерь при изменениях и адаптации готовых ПИК) от использования произведенного программного изделия и т.п.

Контрольные вопросы и задания

1. Определите базис инженерии приложения.
2. Определите базис инженерии предметной области.
3. Приведите модель спецификации ПИК.
4. Назовите функции репозитария.
5. Определите интерфейс компонентов.
6. В чем суть доменной инженерии?
7. Назовите назначение линеек производства программ.