60 к.

Академия наук Украины
Институт кибернетики имени В. М. Глушкова

Препринт 92-37

Л. П. Бабенко, Е. М. Лаврищева

ПРИОБРЕТЕНИЕ ЗНАНИЙ В ИНЖЕНЕРИИ ПРИЛОЖЕНИЙ

Киев 1992

УДК 681.3.06

Приобретение знаний в инженерии приложений / Бабенко Л. П., Лаврищева Е. М. — Киев, 1992. — 22 с. — (Препр. / АН Украины. Ин-т кибернетики им. В. М. Глушкова; 92-37).

Проведен анализ современных направлений в области автоматизации программирования и программной инженерии. Особое внимание уделено проблеме повторно используемых компонентов (ПИК), представляемых готовыми модулями и программами, а также ПИК, отображающих требования, постановки задач и спецификации приложений. Предложены концепции структурирования приложения для выделения ПИК и механизмы их адаптации к условиям применения. Сформулированы способы приобретения знаний в инженерии приложений.

Рассчитан на разработчиков приложений и специалистов, занимающихся методами и средствами автоматизации программирования.

Библиогр.: с. 20—21 (32 назв.)

Рецензент канд. физ.-мат. наук И. В. Криштопа

Утверждено к печати научным советом АН Украины по проблеме «Кибернетика»

Жизненный цикл разработки в инженерии приложений

Анализ современных исследований в области автоматизации программирования [1-14] позволяет сделать вывод об интенсивном становлении нового направления науки, которое может быть условно названо "инженерией приложений". Речь идет о процессе разработки программного обеспечения для различных проблемных областей как инженерной деятельности. В ней правила деятельности специалистов должны быть четко определены, скоординированы, регламентированы стандартами качества и основаны на интенсивном использовании готовых компонент и проектных решений.

Фактор повторного использования компонентов (ПИК) в производстве программного обеспечения для различных приложений приобретает все большее значение.

Трактовка фаз жизненного цикла (ЖЦ) программ, доминирующая в 80-х годах, сводилась (с некоторыми вариациями) к выделению таких стадий, как формулировка требований заказчика программных средств (ПС), спецификация, проектирование, кодирование, отладка, сопровождение. Такой процесс разработки ПС трудоемкий, доля предварительно накопленного продукта в виде ПИК в нем относительно невелика (по оценкам экспертов она составляет 25 % [7]).

Разрабатываемые ПИК представлялись в виде фрагментов исходного кода (процедуры языков программирования, макроопределения, описания структур данных и т. п.), готовых модулей и программ.

Адаптация последних к специфическим условиям применения осуществлялась двумя путями:

присвоением значений параметрам процедур и макроопределений с последующим их встраиванием в другой исходный код;

инженерией фабрик программ [10] (также опыт японских фирм по производству ПС), основанной на банках модулей и механизмах сборки готовых ПИК в интегрированные комплексы для различных приложений.

Метод сборки такого рода ПИК в готовые программные продукты достиг уровня программной инженерии [15,16]. Его основу составляют язык сопряжения, формализованное описание паспортных данных ПИК и средства автоматизированной поддержки процесса интеграции готовых модулей и программ в более крупные программные структуры.

Дальнейшим развитием и совершенствованием данного метода является создание моделей информационного сопряжения и управления готовыми ПИК, ориентированных не на создание жесткой программной структуры, а на создание интегрированной среды функционирования ПИК со средствами обмена данными и передач управлений [4].

Современная тенденция быстрого появления новых и новых тулов (по-английски TOOLs) ставит проблему разработки методов извлечения знаний о них неформализованного типа (свойства, характеристики, назначение, время, надежность работы ПИК и др.) для отображения их в базе знаний экспертной системы (ЭС) [17]. Это позволит конечному пользователю выявить средствами ЭС требуемые ПИК, определить модель интеграции их с другими тулами и сформировать среду их совместного функционирования.

Для извлечения знаний о готовых к употреблению ПИК сложилось три взгляда на ПИК [12], способствующих взаимодействию пользователя с ПИК:

исходное представление ПИК в языке программирования и набор его характеристик;

структурная модель ПИК;

модель данных, используемых в ПИК.

Каждому взгляду соответствует визуальное отображение в виде последовательности экранных карт. Эти карты позволяют проводить навигацию по исходному коду, указывая мышью на модуль, который необходимо вызвать для исполнения, либо произвести изменение в структурной схеме ПИК или в модели данных. Последние две функции работы с картами относятся к области reengineering (системное преобразование и настройка готовых модулей и программ к условиям применения), которое в настоящее время сформировалось как самостоятельное направление в современных CASE-системах [12].

Переход от программной инженерии к инженерии приложений предполагает накопление ПИК на стадии постановки требований и спецификации задач для проблемной области. Выделенная "родовая" информация в виде ПИК отображает спектр задач, к решению которых они могут адаптироваться.

В инженерии приложений информационные технологии основываются на существенно других стадиях ЖЦ разработки ПС (рис. 1.), где создаваемое программное обеспечение для некоторой сферы человеческой деятельности, образующей проблемную область (ПрО), осуществляется с помощью следующих фаз разработки:

анализ объектов и отношений рассматриваемой ПрО, выявление универсальных и стабильных для ПрО знаний, обладающих "родовыми" свойствами, т.е. присущих группам объектов ПрО, и материализация их в виде базы ПИК для данной ПрО;

решение конкретных задач заказчика путем адаптации имеющихся в базе ПИК и достраивания компонентов ПС, не охватываемых базой ПИК;

анализ возможности преобразования вновь достроенных для конкретных задач компонентов на предмет их общности для рассматриваемой ПрО или других ПрО; при обнаружении таковой - преобразование их в ПИК и пополнение, таким образом, базы ПИК.

Основой в данном ЖЦ являются репозиторий (хранилище) ПИК и база метазнаний о всех создаваемых и помещаемых в репозиторий ПИК.

Процесс разработки с вышеописанным ЖЦ можно назвать капиталоемким, если считать капиталом знания о ПрО, материализованные в виде ПИК и готовые к использованию в будущих разработках.

Рассмотрим ключевые проблемы в разработке и использовании ПИК и подходах к их разрешению.

Разработка ПИК для конкретной ПрО включает следующие процессы:

1) распознавание общих закономерностей ПрО, "родовых" объектов и связей между ними, характерных для ПрО правил и ограничений поведения объектов. Этот процесс - присущая человеку когнитивная деятельность по выявлению "повторно используемых ", т.е. типовых, общих приемов и методов решения задач ПрО и фиксации выделенных общностей в виде знаний, допускающих представление их в ЭВМ;

2) представление ПИК в виде, пригодном как для хранения в базе данных и доступа к ним потенциальных пользователей, так и для связывания их в спецификации решения конкретных задач с последующей трансформацией (человеко-машинной или автоматической) в соответствующие программы. Этот процесс должен сопровождаться третьим процессом;

3) представление знаний о функциональных целях, смысле, ограничениях, интерфейсах и правилах включения ПИК в новые разработки, позволяющих пользователю запрашивать сведения о наличии

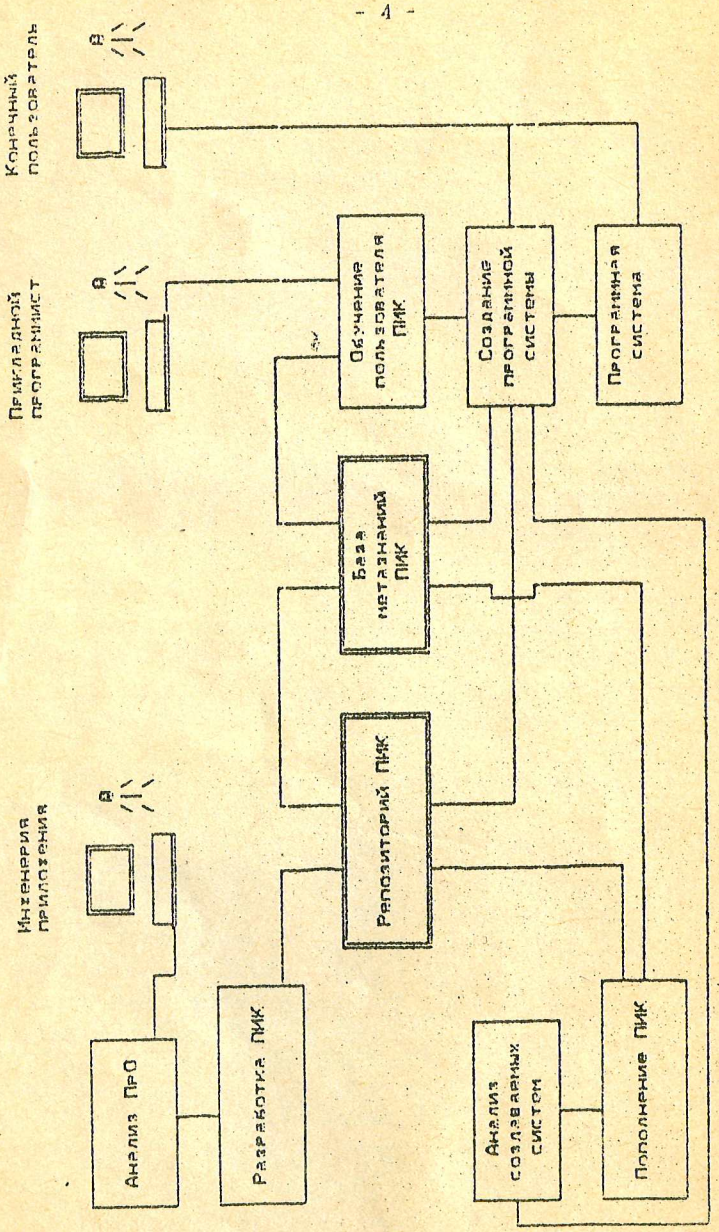

Рис.1. Жизненный цикл в инженерии приложений

требуемых ПИК и обучаться их использованию для удовлетворения своих информационных потребностей.

Процессы 1) и 2) относятся соответственно к первой и третьей стадиям жизненного цикла разработки приложений, процесс 3) - ко второй.

Вышеуказанные процессы являются характерными фазами общего процесса приобретения знаний в экспертных системах, играющего, по общему признанию специалистов, роль "бутылочного горлышка" в сдерживании использования систем, основанных на знаниях [17-18].

Декомпозиция задач и приобретение знаний о ПрО

Всякий процесс приобретения знаний предполагает декомпозицию их на некие порции - концепты. Поскольку в рассматриваемом нами случае объектом когнитивного анализа является ПИК, рассмотрим вопросы их спецификации.

В основе ПИК должна лежать некоторая абстракция и аппарат конкретизации, позволяющие детализировать указанную абстракцию путем настройки на конкретное применение. Другими словами, ценность ПИК определяется адаптируемостью к изменениям окружения. Сложившееся понятие мобильности программ относительно "исполняющих" их технических средств должно быть расширено за счет учета таких нестабильных факторов программного окружения, как операционная система, система управления данными (то, что принято называть платформой исполнения), пожелания к интерфейсам как с пользователями, так и с программными средствами, система понятий предметной области, функции преобразования данных и др.

Для повышения мобильности программ относительно упомянутых факторов естественно выдвинуть требование, чтобы каждый из выявленных факторов возможной "дестабилизации" функционирования программы был максимально локализован с тем, чтобы его модификация стоила минимальных усилий. В соответствии с этим требованием выполняется декомпозиция задачи на такие части, каждая из которых относится лишь к одному из указанных факторов и отображает связанные лишь с ним аспекты решения задачи. Спецификация програмы делится на автономно определяемые части, каждая из которых представляет аспекты соответствующего фактора.

Формализация знаний и выявление ПИК, относящихся к некоторому фактору, должны осуществляться без учета возможного влияния

остальных, и все факторы рассматриваются как "взаимно ортогональные". Как следствие средства спецификации приобретают четко выраженную модульную структуру, в основе которой лежит концепция функционального обрабатывающего модуля (ФОМ), введенная впервые в языке КОБОЛ. Каждый ФОМ представляет собой совокупность языковых конструктов, в общем случае текстуально не локализованных, относящихся к явно выделенным аспектам постановки задачи, соответствующим одному из выше упомянутых факторов. Конструкты отдельного ФОМ могут быть изменены безотносительно к другим.

Сложившаяся технология автоматической обработки информации позволяет назвать в качестве таких ортогональных направлений следующие [19]:

экстракция информации из хранилищ (могут быть файлы, базы данных, базы знаний) и представление ее в соответствии с теми ассоциациями, упорядочением, группированием и формой, которые удобны для ее обработки при решении конкретной задачи;

структуры управления и функции обработки;

система понятий ПрО, используемая в спецификациях;

интерфейс с другими ПИК и человеком, если предусмотрено управляющее воздействие последнего на функционирование ПИК.

Большинство современных CASE-систем в той или иной степени поддерживают отдельные из упомянутых направлений для декомпозиции не столько ПИК, сколько целевых задач в целом (следует заметить, что практически преимущественной разновидностью использования ПИК являются ПИК исходного кода). Анализ рекламных проспектов коммерческих CASE-систем, например [20], показывает, что большинство из них поддерживают спецификацию задачи как совокупность следующих компонентов: единый репозиторий данных; диаграммы потоков данных; диаграммы потоков управления; ERA-диаграммы; диаграммы взаимозависимости и взаимодействия объектов.

Многие из них поддерживают средства спецификации интерфейса с пользователем.

Каждый из вышеперечисленных видов диаграмм специфицирует один из указанных аспектов декомпозиции, однако эти спецификации не интегрируются в единое представление задачи, во многом дублируют друг друга и отражают желание разработчика CASE-системы угодить пользователям, предоставив каждому то технологическое средство, к которому он привык до использования CASE-системы.

В то же время выделенные направления декомпозиции задачи при

спецификации могут служить эффективным средством как классификации ПИК в процессе приобретения знаний о них, так и интеграции их в единую целевую программную систему, поскольку задача интеграции ПИК также может включать в себя следующее:

1. Единая понятийная база выражения информационных потребностей, предполагающая разрешение коллизии имен путем разделения пространства имен и организации соответствующей справочной службы.

2. Интеграция по данным, обмен данными, синхронизация процессов доступа и обновления. Можно выделить ряд методов совместного использования информации: прямая передача значений данных; доступ к одним и тем же файлам; передача данных на основе коммуникаций (для открытых систем и распределенных сред); передача через репозиторий, предполагающая преобразование используемых данных в единую концептуальную модель (чаще всего в этой роли известна ERA-модель). При этом поддерживаются служба запросов к метамодели и метаданным, средства организации "точки зрения" на данные (view) и обмен данными между традиционными файлами и репозиторием.

3. Интеграция по управлению, позволяющая независимо созданным компонентам извещать друг друга о событиях, активизировать процессы, разделять функции.

4. Интеграция по функциям, призванная обеспечить согласованность по типам входных и выходных параметров, однозначность понимания содержания функций и смысла параметров.

5. Интеграция по интерфейсам - правила, стандартизованные как для взаимодействия между программными компонентами, так и для взаимодействия между пользователем и программной компонентой, в том числе и для прямого манипулирования, т.е. взаимодействия с программой методом "укажи и нажми " (выбор из меню с использованием клавиатуры или устройства типа "мышь" и т.д.).

Каждый из вышеприведенных аспектов интеграции обеспечивается соответствующим сегментом знаний о ПИК. Исходя из этого принципы декомпозиции задачи при её спецификации предлагается распространить на принципы декомпозиции знаний о ПИК и соответственно на процесс их приобретения.

Структуризация знаний о ПИК

Анализ современных методов приобретения знаний позволяет

разделить их на две категории. К первой категории можно отнести те, в которых знания получаются от эксперта в виде неструктурированных фрагментов (обычно текстов), и задачей метода ставится выделение концептов (смысловых единиц), связей между ними, т.е. внесение структуризации в получаемые от эксперта знания.

Разработанные методы репертуарных решеток, получения знаний по аналогиям, грубой индукции моделей, формирования понятий и другие [21] эффективны, по нашему мнению, в тех слабо формализованных ПрО, где эксперты не знакомы или мало знакомы с компьютерными методами представления и обработки знаний.

Ко второй категории отнесем методы, основанные на внесении структуризации в ПрО. Выделение допустимых типов концептов и связей до работы с экспертом по знаниям и соответствующее ознакомление последнего "драматически повышает" [2] эффективность процесса приобретения знаний.

Рассмотрим ряд типов концептов, присущих широкому спектру ПрО:

объекты управления;
элементарные и составные действия;
элементарные и составные ситуации;
элементарные и составные понятия;
факты;
условные правила;
безусловные правила.

Среди множества отношений, известных в различных ПрО, также можно выделить набор универсальных:

специальный случай;
обобщение;
объясняет;
подкрепляет;
ослабляет;
классифицирует;
несовместим;
подвергается;
производит;
составляет;
состоит из;
комментарий.

Вышеперечисленные типы концептов и отношений между ними

достаточно универсальны, однако отражают лишь общие закономерности в ПрО; их состав должен быть расширен специфическими для каждой ПрО типами концептов и отношений в процессе анализа ПрО.

Рассмотрим подходы к представлению знаний о ПИК. Будем придерживаться парадигмы объектно-ориентированного подхода, механизм наследования которого открывает широкие возможности для представления ПИК. Действительно, при построении ПИК используются две конкурирующие стратегии: стремление к адаптивности и композиционности. В первой из них акцент делается на свойства, позволяющие получать варианты ПИК, во второй - на спецификацию инвариантных свойств.

Планирование адаптации на ранних стадиях создания ПИК требует представления соответствующих знаний. Механизм наследования - прямой инструмент адаптации, в том числе извлечения знаний из одного домена и применения их в другом.

Следуя [22], будем использовать формулу выражения наследования при построении ПИК:

<новый класс > = IS - PLUS - BUT - EXCEPT

Здесь IS - класс, который наследуется (исходный класс).

PLUS - новые свойства, которые приписываются новому классу и которыми не обладает исходный класс.

EXCEPT - свойства исходного класса, которые не наследуются новым классом.

BUT - свойства исходного класса, которые наследуются только при определенных пред - и постусловиях.

Для каждого класса могут быть предложены его информационные характеристики:

1. Основные дескрипторы класса:
   имя класса;
   ключевые слова, определяющие функциональные свойства класса;
   неформальное описание функциональных свойств класса.
2. Стуктурные характеристики класса:
   ключевые слова, определяющие функции методов класса;
   типы входных параметров;
   типы выходных параметров.
3. Характеристики среды:
   язык реализации;
   технология программирования;
   платформа оборудования.

4. Отношения между классами:
наследование;
включение как подсистемы;
вызывает (вызывается).

Дескрипторы классов, предопределенные типы концептов и отношений между ними могут служить исходным состоянием для процесса приобретения знаний. Начальными его шагами могут быть:

стандартизация понятий и терминологии на всем информационном пространстве функционирования системы, нормализация лексики и устранение возможных лексических разночтений;

фиксация и поддержание внешнего (ориентированного на эксперта ПрО) представления знания о системе понятий, адекватной решаемым задачам, в виде иерархии объектов, тезаурусов, отражающих их понятийные связи и агрегирование по использованию в отдельных функциях;

обеспечение эксперта ПрО сведениями об уже накопленных знаниях, о принятой терминологии и установленных связях между объектами;

создание концептуального уровня представления знаний для композиции автономно приобретенных знаний в единую базу знаний;

обеспечение возможности образования проекций баз знаний и баз данных для автономных подсистем ПрО;

предоставление справок об источниках и истории приобретения знаний;

терминологическая экспертиза знаний, поставляемых экспертом ПрО.

Построение справочника знаний является первой фазой процесса приобретения знаний, общей для всех разновидностей знаний и всех ПрО. Справочник знаний - инструмент поддержки метазнаний, т. е. знаний о знаниях, и соответствующей настройки на их основе всех используемых механизмов взаимодействия с пользователями и программами.

Дальнейший анализ ПрО для выявления ПИК целесообразно провести по установленным нами ранее аспектам декомпозиции. Рассмотрим их подробнее.

Выявление понятий ПрО в виде ПИК

Понятие - это обобщение представлений о предметах некоторого

класса по существенным, специфическим для этого класса признакам, которые помогают выделению, распознаванию, отождествлению. На основе понятий осуществляются классификация, обобщение, структурирование объектов. Входящие в понятие признаки делятся на объединительные и разъединительные. Над понятиями допустимы следующие операции: объединение (в родовое понятие), пересечение, сложение (обобщение), вычитание (исключение).

Использование понятий в автоматизации ПрО имеет давние традиции; для представления знаний о них имеются удобные формализмы (продукции, фреймы и др.). Установление понятийной базы ПрО, гибкий аппарат адаптации поддерживаемой системы понятий к конкретным доменам предметной области, динамическая поддержка текущего содержания понятий (объем заключенных в них знаний) позволяют выявлять ПИК на стадии спецификации задач ПрО и автоматически трансформировать их в ПИК исходного кода.

ПИК экстракции информации

В процессе анализа ПрО выявляются характерные артефакты, правила их представления в виде структур данных и связей между ними, для отображения которых могут использоваться ERA-модель данных, диаграммы потоков данных [23], нотация Буча [24,25]. Выявление ПИК экстракции информации основывается на характерных для ПрО методах конструирования "виртуальной точки зрения" на данные, позволяющих автоматически организовать сбор, хранение и предъявление артефактов в удобном для решения задачи ПрО виде, т. е. в соответствии с удобными для неё ассоциациями, упорядочением, группированием и формой представления. Такие наборы ПИК могут быть стандартизованы как средства получения внешней, ориентированной на конечного пользователя ПрО модели данных (терминология известного отчета ANSI-SPARC), экранирующей особенности структурирования и распределения данных в базах данных.

ПИК управления и функций

В общем случае эти виды ПИК связаны между собой, их выявление и представление - наиболее творческий процесс в инженерии знаний, поскольку "родовые" свойства функционального содержания ПИК в значительной степени зависят от выбранной формы представле-

ния информации и базового набора понятий. Стандартизация ПИК экстракции информации облегчает стандартизацию ПИК функций. Объединение их в единую структуру образует ПИК класса объектов. Библиотеки классов в [25] содержат около 500 ПИК.

ПИК конструирования интерфейсов с пользователем

Проектирование "дружественного" интерфейса ПС (тула) с пользователем относится к существенному компоненту современной технологии програмирования.

На эту цель, по оценкам [27], тратится до 40 - 50% усилий по разработке; обеспечение интерфейса отвлекает примерно половину ресурсов памяти ПС и объема программного кода. Таким образом, затраты на обеспечение интерфейса вполне сравнимы с затратами на обеспечение функциональных свойств ПС. Этим объясняется интенсивное развитие технологии конструирования интерфейса.

Системы конструирования интерфейсов (СКИ) - это инструментальные системы для разработки и исполнения человеко-машинных интерфейсов в интерактивных системах ПО. Они помогают в спецификации, проектировании, прототипировании, реализации, исполнении, оценках, модификации и поддержке этих интерфейсов. Это один из инструментов CASE-систем. Его назначение: повышение продуктивности труда создателей интерфейсов; "не программистские" средства, не требующие кодирования; привлечение профессионалов в прикладных областях к проектированию интерфейсов, что приближает к потребностям конечного пользователя.

Выделим [28] ряд поколений систем управления интерфейсом пользователя (СУИ).

Первое поколение обеспечивало моделирующее прототипирование и управление выводом, ориентированное на программистов, проектирование сценариев диалога в меню-ориентированной форме, конструирование входных и выходных экранов заполнения полей, иерархические меню, обучение. Однако связать эти средства с результатами программных вычислений можно было только прямым программированием.

Второе поколение характеризуется усилиями по поддержке стадии исполнения.

Часто используются диаграммы переходов состояния, ограниченные стилем интерфейса. Характер диалога не зависит от семантики приложения. Его представлением, например, может быть:

1) абстрактная форма, элементами которой являются средства коммуникации между пользователем и приложением;

2) командный язык в терминах меню, функциональных кнопок, локальных устройств, вычислителей значений, по которому автоматически генериуется Paskal-код;

3) редакторы экранов и икон (под которыми понимаются изображения на экране информации в виде пиктограмм), представляемые библиотекой процедур интерфейса, не интегрируемой с другими инструментами CASE-систем.

Третье поколение характеризуется развитием средств поддержки проектирования и исполнения. Широко используются средства прямого манипулирования (и его орудия - мышь, палец), возрастает функциональность, разнообразие типов стилей интерфейса и устройств (окна, мышь и т.д.). Тип диалога изменяется от последовательного до прямоманипулируемого, асинхронного со сложной графикой. Появляются новые подходы: интерфейс посредством демонстрирования; объектно-ориентированный и "оконный" подходы, как например, интерфейсы в стиле MAKINTOSH; окна; всплывающие меню; радиокнопки; боксы выбора. Появляются средства определения " горячих пятен" экрана, доступ к которым осуществляется по кнопкам мыши. Получают распространение библиотеки икон со встроенной семантикой.

Жизненный цикл имеет не "водопадную" (каскадную), а "звездную" структуру с фазой исполнения в центре.

Четвертое поколение можно характеризовать использованием подходов искусственного интеллекта к конструированию СУИ.

Разработчики компьютерных систем традиционно рассматривают средства организации интерфейса пользователя и ПС как логически отдельную компоненту ПО, хотя чаще всего физически они не локализованы в программе как независимо созданные заменяемые модули.

В настоящее время физическое обособление компонента обеспечения взаимодействия прикладных программных систем от компонентов обеспечения их функциональных требований осознано вычислительным сообществом и как возможное, и как желательное. Такое разделение имеет целый ряд преимуществ: создание прототипов пользовательских интерфейсов на ранних стадиях ЖЦ; возможность иметь в рамках одного приложения несколько персонально ориентированных интерфейсов; возможность создания единого ("родового") интерфейса для многих приложений; возможность модификации интерфейса по изменившимся требованиям пользователя.

Идеи отчуждения интерфейса от приложения нашли развитие в очередном глобальном проекте фирмы IBM, получившем название системной архитектуры приложений (System application architecture - SAA). Как образно отмечается в [29], объявление этого проекта, систематизировавшего и развившего современное состояние СУИ, является "третьим крупнейшим призом, выигранным IBM" (первым автор считает переход от индустрии счетных машин к индустрии ЭВМ, вторым - объявление операционной системы OS/360).

Цель проекта SAA - достижение совместимости прикладных систем при переходе к различным исполнительским средам и платформам и тем самым достижение независимости прикладных систем от архитектурных особенностей ЭВМ и операционных систем.

Концептуально проект поддерживает идею капиталоемких технологий - капитал вкладывается в небольшой ряд согласованных операционных систем для всех типов ЭВМ, поддерживаемых IBM (OS/370, AS/400, PS/2), для которых обеспечиваются шесть основных целей:

- длительность использования;
- архитектурная поддержка эволюционного роста;
- способность связывать разнородные компоненты в единые автоматизированные системы управления;
- согласованность сообщений;
- способность взаимодействия на уровне загрузочных компонентов;
- сохранение приобретенных навыков использования среды при смене платформы - исключение фактора "переучивания" для персонала.

Акцент делается на передачу "обученности" при переходе от среды к среде.

Средством достижения поставленных целей является выделение интерфейса как замкнутой функции, отчуждаемой от приложения. При этом определяется согласованное множество базовых функций, покрывающих возможности существующих общесистемных средств и нивелирующих различия операционных систем и платформ.

В проекте SAA предусматриваются три компонента: общий доступ пользователя (Common User Acces - CUA); общий программный интерфейс (Common Programming Interface - CPI) и общая поддержка коммуникаций (Common Communication Support - CCS). Рассмотрим более подробно назначение этих компонентов.

CUA - набор правил для проектирования интерфейса программного средства и пользователя - человека. Вводится ряд стандартов взаимодействия с пользователем: одинаковый формат и представление

экрана; стандартное местоположение заголовков, командных строк и области подсказки системного меню; команды начала сеанса, запуска прикладной системы, выбора ситуации должны быть согласованы по именам, синтаксису, семантике; вводится единая терминология для всех прикладных систем; объявляется прозрачность локальных и удаленных объектов (пользователь не обязан знать, является ли объект локальным или удаленным).

Выделяются базовые функции управления (способ взаимодействия с пользователем) и шаблоны (templates) - наборы связанных функций управления для создания панелей.

Выделяются стандартные типы окон, основными элементами которых являются заголовок, скроллинг-информация. Поддерживаются следующие функции управления: клавиатурой, редактированием текста, состоянием, списком ("ролик списка"), меню - главное или скроллинг-меню, скроллингом, размером окна, заголовком. Образец стандартного формата окна для OS2 показан на рис. 2.

Объекты и команды интерфейса базируются на объектах и командах приложения, а не на используемых физических графических устройствах. Выделяются общие для всех сред элементы интерфейса: клавиатура; "мышь"; скроллинг (перемещение текста вдоль экрана сверху вниз, снизу вверх, слева направо и справа налево); листание текста (browsing); цвет; яркость; ввод данных и сообщений; оказание помощи; кнопки выбора (единственный или множественный). Панели интерфейса строятся как композиции указанных элементов.

CPI - набор правил межпрограммного взаимодействия на основе единого унифицированного взгляда на функцию прикладной системы, безотносительно к тому, выполняется ли она рабочей станцией (ПЭВМ) или центральным процессором большой машины. CPI стандартизирует применение широкого круга языков программирования: ФОРТРАН, Си, КОБОЛ, РПГ, REXX, языки генератора приложений Cross System Product. Стандартизируются также правила применения типовых системных служб: баз данных, организации диалога, запросов, представлений. Рассматриваются две файловые модели данных - потоко- и запись-ориентированные; они стандартизированы в рамках одного языка программирования.

CPI решает задачу обеспечения CUA в SAA, обеспечивает переносимость прикладных систем и навыков работы с ними относительно используемых платформ операционных систем.

CCS определяет правила распределенной обработки. Рассматри-

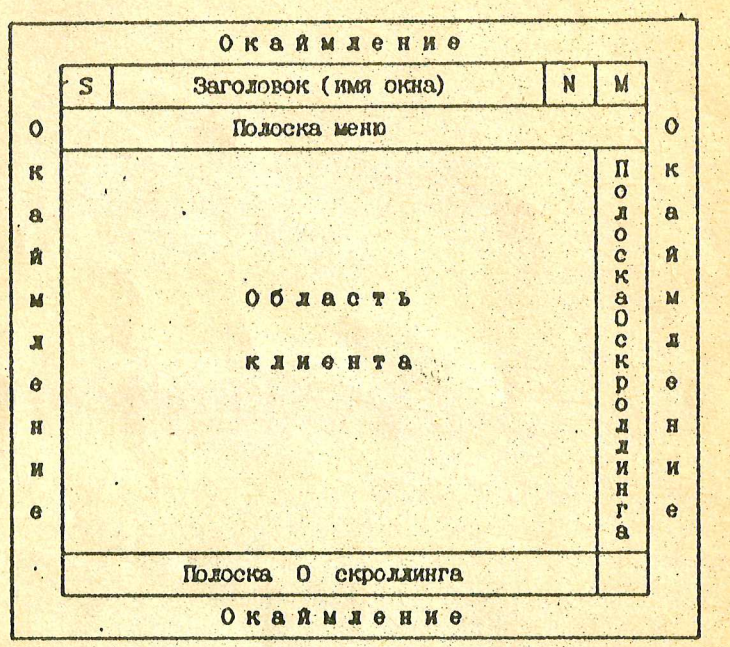

Рис. 2. Стандартный формат окна:

S - системное меню;

M - блок максимизации;

N - блок минимизации;

O - точки для указания на полосках скроллинга

ваются три способа распределения данных:

1) совместная обработка данных, размещенных в файлах центральной машины посредством доступа от рабочих станций на ПЭВМ. Как правило, создается виртуальный файл, отображающий потоко ориентированный файл рабочей станции в файл с записями фиксированной длины на центральной машине;

2) локальные сети без центральной машины. Все рабочие станции считаются равноправными и могут взаимно использовать такие ресурсы, как время процессора, память, принтер и др. Одна из машин может специализироваться по определенному ресурсу;

3) распределенные базы данных, когда доступ и обновление допускаются на разных платформах. Оба процесса допускаются в одной транзакции и пользователь не заботится о том, что и где хранится, при этом обеспечивается защита от несанкционированного доступа.

Учитывая вышеизложенное, следует отметить, что проект SAA создает платформу-оболочку, "единый мир со стандартным лицом", позволяющий старые продукты исполнять на новых операционных системах и новом оборудовании. Заметим, что большинство ведущих разработчиков ПС мира объявило о своей поддержке концепции SAA.

Компоненты SAA можно считать общесистемными ПИК, независимо от ПрО. Следующим шагом накопления ПИК для конструирования интерфейсов следует считать ПИК, специфические для ПрО.

Перечислим функции, которые присущи таким ПИК: поддержка мыши и других устройств оборудования; проверка правильности ввода пользователя; обработка его ошибок и аварийных выходов; обеспечение обратной связи с пользователем; визуализация его манипуляций; помощь и подсказка; визуализация изменений значений данных; редактирование экранов. Для спецификации конкретных требований к ПИК интерфейса используются специальные языковые средства. Рассмотрим подходы к их организации.

Примем модель интерфейса, предложенную в [30], суть которой представлена на рис. 3.

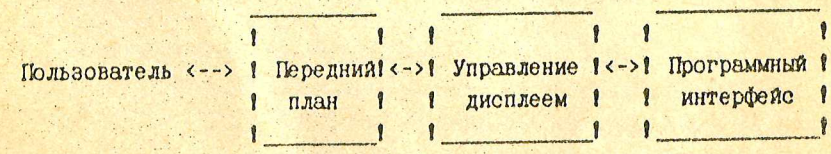

Рис. 3. Модель интерфейса

Назначение интерфейса - отображать на дисплее внешнее воздействие пользователя и сообщения, посылаемые функциональным компонентам приложения (ФКП), а также данные, извлекаемые из ФКП.

При этом внешним воздействием пользователя может быть нажатие клавиш панели, движение мыши и (или) нажатие ее клавиш (эти действия называются прямым манипулированием).

В модели интерфейса, следуя [30], выделим три компонента: передний план - программа-посредник между пользователем и дисплеем; управление дисплеем - компонент, ответственный за появление объектов на экране; программный интерфейс - взаимодействие компонента управления дисплеем и ФКП.

Для каждой ПрО могут быть специфицированы объекты экрана, поведение которых предопределено. Среди них различают:

1) пассивные объекты, изображение которых появляется на экране, но не порождает при этом управляющих событий и не требует реакции пользователя. Примерами таких объектов могут служить рамка указанной толщины вокруг другого объекта, статическое изображение, неизменяемая строка или фрагмент текста;

2) интеракторы - объекты, связанные с событиями, которые пользователь может порождать прямым манипулированием. Состояние интерактора связывается с сообщением, которое посылается ФКП. Примерами интеракторов являются меню, кнопки выбора, окна диалога.

Стандартные виды интеракторов определены в проекте SAA, однако для конкретной ПрО могут определяться специфические для нее виды объектов, как статических, так и интеракторов. Так, для ПрО, в которых профессионалы имеют дело с показаниями приборов, например в кабинах самолетов или на пультах управления электростанциями, в качестве ПИК интерфейса могут быть специфицированы соответствующие объекты, имеющие образ в виде шкалы прибора, вертикальной или круговой, и состояние в виде показателя значения. Такой объект может быть определен и как статический, состояние которого запрашивается от ФКП, и как интерактор, на который оказывает воздействие пользователь.

Для объекта экрана могут при этом определяться его визуальное представление, так называемая икона; состояние по умолчанию; допустимые для объекта воздействия пользователя; значения атрибутов экрана, соответствующие определенным воздействиям; сообщения, которые посылаются ФКП.

Система конструирования интерфейса может иметь в качестве

инструмента генератор икон [31]; при спецификации объектов дисплея иконы могут быть определены как родовые объекты с параметризованными атрибутами, например для шкалы, минимальное и максимальное значения, шаг деления, цена шага, шрифты подписей и др.

Наборы таких ПИК разработаны, например, для систем, использующих электронную почту [32]. Для спецификации адаптивных компонентов интерфейса используются методы, основанные на знаниях.

Цель спецификаций - обеспечить отображение команд пользователя в набор команд конкретной системы электронной почты, в то же время не обременяя пользователя знанием специфики используемой конкретной системы; на основе спецификаций обеспечивается также возможность проверки правильности команд пользователя и обнаружение семантических ошибок до передачи задания ФКП.

На основе анализа ряда действующих систем электронной почты была создана ее модель, представляющая обобщенные команды электронной почты, определение их функций, побочных эффектов и обратимости, возможные стратегии отката, т. е. возврата ФКП в известное состояние при обнаружении непредвиденной ситуации, диагностические сообщения и их смысл, стратегии по умолчанию, используемые ФКП.

База знаний, специфицирующая интерфейс, представлена как сеть переходов от состояния к состоянию, узлы которой являются состоянием ФКП, а дуги представляют команды, вызывающие переходы из состояния в состояние. Для представления этой сети используется логика предикатов первого порядке. Поддерживаются также знания о состояниях ФКП по умолчанию, о прогнозируемых ошибках пользователя, соответствующих им сообщениях и стратегиях отката. При необходимости перехода от одной конкретной системы электронной почты к другой достаточно замены только соответствующей базы знаний об интерфейсных ПИК, при этом переучиваться пользователям не требуется.

Выводы

Сформулирована модель ЖЦ автоформализации знаний о ПрО, в основу которой положена методология ПИК.

Предложены концепции структурирования ПрО, ориентированные на выявление родовых знаний в виде ПИК с механизмами их адаптации к конкретным условиям функционирования, тем самым заложены предпосылки создания долгоживущих ПС.

Определены принципы декомпозиции задач ПрО, ориентированные на выделение и использование ПИК. Намечены пути их автоматизации.

Список литературы

1. Objekt Frameworks / Ed. D. Tsichritzis. - Geneve: Centre Univ. d'informatigue, 1992. - 321 p.
2. Forte G., Norman R. I., A Self-assessment by the Software Engineering Community // CACM. - 1992. - N4. - P. 43-49.
3. Maiden N., Sutcliffe A. Exploiting Reusable Specifications Through analogy // Ibid. - P. 55-64.
4. Baxter I. Desing Maintenance Systems // Ibid. - P. 73-88.
5. Caldiera G., Basili V. Identifying and Qualifying Reusable Software Components // Computer. - 1991. 24, N2. - P. 61-70.
6. Barnes B.H., Bollinger T.B. Making Reuse cost Effective // IEEE Software. - 1991. - N2. - P. 13-24.
7. Norman R., Chen M. Working Together to Integrate Case // Ibid - P. 12-17.
8. Norman R., Chen M. A Framework for integrated CASE // Ibid. - P. 18-22.
9. Thomas I., Nejmeh B. Definition of tools integration for Environments // Ibid. - P. 29-35.
10. Fernstrom C. Software Factory, Principles, Architecture and Environment // Ibid. - P. 36-44.
11. Thomas I. Case database design, the key to framework PCTE: building a foundation for case // Computer Desing. - 1991. - N7 - P. 104-105.
12. Williams T. New tools bring case power to embedded systems // Ibid. - P. 92-93.
13. Corbi T.A. Program understanding: challenge for the 1990 // IBM Systems J. - 1989. - 28, N2. - P. 294-306.
14. Tsichritzis D., Nierstaz O. Beyond objects: objects//Intern. J. of intelligent and cooperative inform: Systems. - 1992. - 1, N1. - P. 43-60.
15. Лаврищева Е.М. Технологическая подготовка и программная инженерия //УСиМ. - 1988. - N1. - С. 48-52.
16. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. - Киев: Наук. думка, 1991. - 209 с.
17. *Topics in* Expert Systems Design. Methodologics and Tools / Ed. G. Guida, C. Tasso. — Amsterdam etc.: North Holland, 1989. — 441 p.
18. *Aylett R.* Knowledge acquisition tools // Expert Systems User. — 1990. — 6, N 6. — P. 17—23.
19. **Бабенко Л. П.** Компоненты многократного использования и сборочное проектирование систем обработки данных // Кибернетика. — 1989. — № 6. — С. 33—35
20. *Packages* Software Reports. — 1991. — N 1—12.
21. *Knowledge* — Based Systems / Ed. J. Boose, B. Gaines. — N-Y; London: Academic Press, 1988. — 2. — 512 p.
22. *Steggles P.* Reuse implies Eiffel //EXE. — 1991. — N 6. — P. 39—46.
23. *Martin J., Mc Clure C.* Structured techniques the basis for Case. — N.-V: Prentice Hall; Englewood Cliiffs, 1988. — 413 p.
24. *Booch G.* The Booch Method // Computer Language. — 1992. — Part 1. — P. 47—70.
25. *Booch G.* The Booch Method // Ibid. — Part 2. — P. 37—51.
26. *Meyer B.* Lessons from the design of the Eiffer // CACM. — 1990. — P. 69—74.
27. *Myers B. A.* User interface Tools: Introduction and Survey // IEEE Software. — 1991. — 6, N 1. — P. 121—128.
28. *Hix D.* Generations of User Interface Management Systems // Ibid. — P. 23—29.
29. *Libutti L. R.* Systems Application Architecture. — S. I. : TAB BOOKS, 1990. — 204 p.
30. *Harbert A., Lively W., Sheppard S.* A graphical Specification Systems for User — interface design // IEEE Software. — 1990. — N 4. — P. 12—20.
31. **Адамович И. М., Козырев В. В.** «Пиктограф» — пакет для создания интерфейсов пользователя // Мир ПК. — 1991. — № 5. — С. 103—106.
32. *Adhami E., Browne D. P.* Generic user interfaces for multiple applications // Proc. 3rd Intern. Expert Systems Conf. — London, 1987. — P. 233—244.

Научное издание

Бабенко Людмила Петровна
Лаврищева Екатерина Михайловна

ПРИОБРЕТЕНИЕ ЗНАНИЙ В ИНЖЕНЕРИИ ПРИЛОЖЕНИЙ

Редакторы: М. И. Сахарова
Г. Ф. Трохимец

Подп. в печ. 24.12.92. Формат 60×84/16. Бум. кн.-журн. Офс. печ. Усл. печ. л. 1,16. Усл. кр.-отт. 1,28. Уч.-изд. л. 1,5. Тираж 100 экз. Заказ 1954. 60 к.

Редакционно-издательский отдел с полиграфическим участком
Института кибернетики имени В. М. Глушкова АН Украины
252207 Киев 207, проспект Академика Глушкова, 40