УДК 629.78.018

**Е.В. СОКОЛОВА**

***Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина***

# ПРОЕКТИРОВАНИЕ СЛОЖНЫХ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ КОМПЛЕКСОВ НА ОСНОВЕ МОДЕЛИ ТРАЕКТОРИЙ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Выполнен анализ известных современных технологий разработки проблемно-ориентированных программных комплексов. Обоснована необходимость создания новых методов и технологий разработки для целого класса задач, обеспечение гарантоспособности которых – трудоемкая задача из-за сложности внутренней организации процесса поиска решений. Рассмотрена теоретическая возможность построения таких систем на основе модели вычислительных процессов, которая базируется на идее переноса описания процесса поиска решения во внешний источник информации – базу данных. Показано, что в таком случае разработка проблемно-ориентированного профессионального языка позволит привлечь специалиста в данной прикладной области к процессу разработки программного обеспечения

**технология проектирования, вычислительный процесс, траектория вычислительного процесса**

## Введение

Проектирование программного обеспечения представляет собой сложный итеративный процесс, включающий реконструкцию и исправление на каждой стадии разработки. Время и силы, затраченные на этапе проектирования, вознаграждаются тем, что при удачной декомпозиции каждый компонент можно программировать и модифицировать независимо или почти независимо от остальной системы. От разработчика требуется высокая степень терпимости к неопределенности. По завершению этапа проектирования разработчик получает структурированное представление о задаче, возможных путях ее решения, может бегло ориентироваться в проблеме. Задача разработчика на этапе проектирования заключается в том, чтобы выделить и компактно выразить основную логику, рассматривая программу на нескольких уровнях абстракции, определяя первые уровни весьма приблизительно и неформально.

Тем не менее, несмотря на исключительную важность, проектирование является неформализуемым этапом создания программы. Значительная часть усилий уходит здесь на выявление постановки задачи, что предполагает активное взаимодействие разработчика с пользователем непрограммистом. Более того, пользователь зачастую и сам не имеет четкого представления о своих нуждах. Попытка выбрать для общения с ним формальный язык или специализированную компьютерную терминологию обречена на неудачу в силу языковой нестыковки.

## Формулировка задачи

В настоящее время широко распространены методы автоматизированного конструирования информационных систем, основанные на концептуальных (понятийных) моделях представления потоков данных [1]. В этом случае основной акцент ставится на формальное вычисление по концептуальной модели структуры базы данных и запросов к этой базе данных. Широкое применение также находят функциональные модели, которые представляют схемы обработки данных и являются скорее моделями для управления разработками. Охарактеризуем наиболее известные из них.

Модель **SADT** представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из

основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы.

Язык **SDL** [2] (Specification and Description Language) предназначен для разработки событийно-ориентированных распределенных систем. Как средство анализа систем язык SDL широко используется в европейских телекоммуникационных стандартах. Его основными составляющими являются структурная модель и расширенный конечный автомат.

Язык **UML** [3] (Unified Modeling Language) представляет собой попытку обобщить графические нотации различных объектно-ориентированных методологий, ввести единый стандарт на объектно-ориентированные средства визуальной спецификации. Недостатком является то, что при этом стандартизован язык, а не средства его применения.

Основополагающий принцип метода **OOSE** (Object-Oriented Software Engineering) – объектная ориентированность, как для анализа, проектирования, программирования, так и для описания процесса разработки ПО в целом. Метод предназначен, в первую очередь, для разработки больших систем и в 1995 году, этот метод стал основой **RUP** (Rational Unified Process). Метод RUP концентрирует внимание на технологии применения языка моделирования и систематизирует процесс создания ПО на основе UML.

Методология **ROOM** (Real-time Object-Oriented Modeling) – это объектно-ориентированная методология разработки систем реального времени. ROOM содержит два уровня представления разрабатываемой системы:

- уровень схем;
- уровень детализации.

Выделение этих уровней нацелено на автоматическую генерацию кода. Для уровня схем ROOM предлагает набор графических нотаций. Уровень детализации предполагает использование языка реализации, поскольку сложную систему нельзя специфицировать в виде картинок, достаточных для автоматической генерации работающей программы.

Вместе с тем, существующие методологии являются скорее философией, а не технологией компьютерной инженерии. Эти подходы применимы к сверхбольшим проектам с сотнями разработчиков и сроком исполнения – несколько лет, в которых доли анализа и проектирования достаточно велики по сравнению со всей разработкой. Такие проекты отличаются огромным числом сложно-структурированных данных и небольшим числом вычислительных модулей, например информационных систем в сфере бизнеса, банковского дела и т.д.

Альтернативный обширный класс программ – системы, в которых много вычислительных модулей, преобразующих исходную информацию в результирующую, а обрабатываемые данные могут иметь достаточно простую структуру. К такому классу программ можно отнести системы автоматизации технического и конструкторского проектирования, системы управления, геоинформационные системы, системы автоматизации испытаний и научных исследований; трансляторы как языков программирования, так и профессиональных языков и др. Это достаточно сложные проблемно-ориентированные комплексы программных модулей, для которых характерна задача организации управления совокупностью функциональных модулей, преобразующих исходные данные в результирующие. Эффективное решение такой задачи требует конструирования профессионального языка для записи правил этого преобразования. В результате достигается повышение надежности программных комплексов.

К представлению алгоритмов в подобных комплексах предъявляются строгие требования: надежность, устойчивость, верифицируемость, сопровождаемость, которые, в свою очередь, определяют необходимость обеспечения отсутствия ошибок, ди-

намический контроль корректности условий выполнения, переносимость, модифицируемость, понимаемость и т.д.

Поэтому в настоящее время весьма актуальны исследования, направленные на создание эффективной технологии конструирования проблемно-ориентированных комплексов.

**Цель исследований** — теоретическое обоснование технологии проектирования сложных проблемно-ориентированных комплексов на основе модели траекторий вычислительных процессов.

## Результаты исследований

Выбранный для исследования класс сложных проблемно-ориентированных программных комплексов позволяет предложить архитектурное решение, обеспечивающее декомпозицию общей сложной задачи на последовательность более простых подзадач.

Предлагается трехуровневое решение, когда программные компоненты каждого уровня обеспечивают последовательное преобразование исходных данных $d$ в результат $r$ (рис. 1).

Алгоритмы преобразования данных необходимо вынести из программы во внешний источник данных, представленный, например, в формате реляционной базы данных (БД). Правила такого преобразования определяет эксперт в данной предметной области, задача которого их сформулировать на профессионально-ориентированном языке. Так как эксперт, в общем случае, не владеет навыками записи алгоритма и его преобразования в программу ему необходим инструмент для этого. Таким инструментом является лингвистический процессор $L$ [4], который осуществляет взаимнооднозначное, двунаправленное преобразование профессионально-ориентированного языка в язык исполнительных команд.

Выбор подходящего варианта алгоритма из БД возлагается на операционный процессор $OP$, а непосредственное исполнение алгоритма осуществляется функциональным процессором $F_{op}$. При таком подходе лингвистический и операционный процессоры могут быть достаточно универсальными, а функциональный процессор должен быть ориентировании на предметную область.

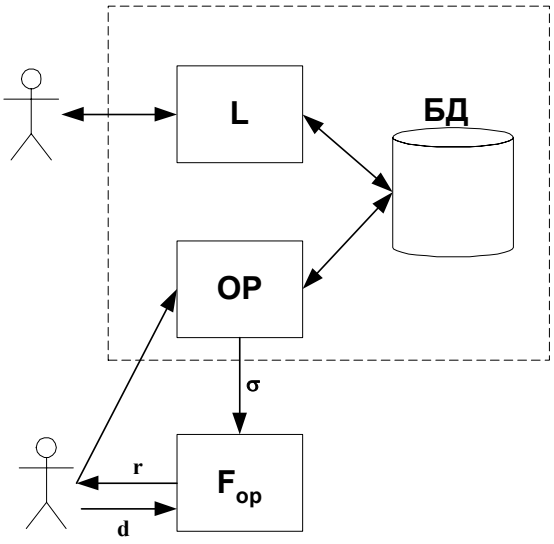


Рис. 1. Архитектура сложного проблемно-ориентированного комплекса

Рассмотрим данный процесс с формальной точки зрения. Пусть задан некоторый класс прикладных задач $T$. Каждая прикладная задача $T_i \in T$ из этого класса представляется как функциональное преобразование множества $D$ допустимых исходных наборов данных в множество результатов $R$:

$$T_i : D \longrightarrow R . \qquad (1)$$

Обозначим через $\Im_i$ процесс поиска решения $r \in R$ прикладной задачи $T_i$ для заданного варианта $d \in D$.

Понятие «траектория вычислительного процесса» подразумевает реализацию $T_i(d)$ вычислений, отождествляемых с определением значения преобразования прикладной задачи в заданной точке:

$$T_i(d) = r, d \in D, r \in R . \qquad (2)$$

В каждом конкретном случае, процесс вычисления значения преобразования прикладной задачи $T_i$ при заданном аргументе – варианте $d$ образует некоторую последовательность $\{\sigma_{T_i}(d)\}$ базисных команд (операций) преобразования данных. При этом предполагается, что имеется некоторый операционный процессор $OP$, способный выполнять такие базисные команды (операции). Множество $\Sigma$ таких базисных команд преобразования данных, каждую из которых может исполнять операционный процессор $OP$, образует систему команд операционного процессора.

Операционный процессор можно представить как совокупность некоторой проблемно-ориентированной библиотеки подпрограмм и управляющей программы – дешифратора.

Дешифратор связывает идентификатор команды с подпрограммой, исполняющей эту команду, и запускает конкретную подпрограмму при поступлении на вход соответствующего ей идентификатора команды.

Для каждой команды операционного процессора $\sigma \in \Sigma$ определены исходные и результирующие операнды. Определим информационно – совместимую последовательность команд $\{\sigma\}$ как такую последовательность, в которой исходные операнды любой команды вычисляются в процессе исполнения предыдущих команд. Обозначим множество таких последовательностей команд операционного процессора $OP$ через $\Sigma$.

Таким образом, операционный процессор определяется как функциональное преобразование:

$$F_{op} : \Sigma^* \times D \to R . \qquad (3)$$

В целом, предложенная схема реализует идею трехуровневого преобразования:

$$\begin{aligned} &L : T_i \to f_{бд}; \\ &OP : F_{бд} \to \sigma; \\ &F_{op} : \Sigma^* \times D \to R. \end{aligned} \qquad (4)$$

Особенностью данного решения является универсальность подхода к проектированию сложных проблемно-ориентированных комплексов, который позволит автоматизировать получение результата и минимизировать влияние человека на состояние системы, тем самым повысив надежность комплекса в целом.

## Выводы

В результате исследований теоретически обоснована технология проектирования сложных проблемно-ориентированных комплексов на основе модели траекторий вычислительных процессов, которая позволит полностью перенести описания выполняемых действий, необходимых для реализации сложного проблемно-ориентированного комплекса из программной реализации во внешний источник информации, представленный в формате реляционной базы данных.

## Литература

1. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М. Финансы и статистика, 2000. – 340 с.

2. Мансуров Н.Н., Майлингова О.Л. Методы формальной спецификации программ: языки MSC и SDL. – Издательство АО “Диалог-МГУ ”, 1998. – 125 с.

3. UML и Rational Rose У.Боггс, М.Боггс: – М: Издательство «Лори», 2000. – 580 с.

4. Кузин С.Г. Ролевый граф в качестве модели понятия // Вестник Нижегородского университета: Математическое моделирование и оптимальное управление. – Н. Новгород: Изд-во Нижегородского университета. – 1998. – 2 (19). – С. 224-235.

*Поступила в редакцию 26.02.2007*

**Рецензент:** д-р техн. наук, проф. А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.