*Проблемы бионики*
*1999. Вып. 51. С. 141–145*
ХТУРЭ
УДК 007.681.518.2

*О.С. ПОНОМАРЕНКО*

# ИНТЕЛЛЕКТУАЛЬНЫЕ CASE-СРЕДСТВА НОВОГО ПОКОЛЕНИЯ

При создании программного обеспечения (ПО) влияние человеческого фактора влечет за собой низкую эффективность и производительность разрабатываемого ПО. Так, по мере развития методологий разработки ПО осуществлялась и их автоматизация. В промышленности уже существовали такие системы, как CAD (Computer Aided Design - автоматизация проектирования) и CAM (Computer Aided Manufacturing - автоматизация производства). В сфере разработки ПО аналогичные системы появились значительно позже, и им соответствовала CASA (Computer Aided System Analysis - автоматизация программирования). Их объединение и породило CASE (Computer Aided Software Engineering). В отечественной литературе термин CASE переводится как "автоматизация проектирования и реализации ПО", где под проектированием понимается все, что предшествует реализации. Годом рождения CASE считается 1984, когда на рынке ПО появились первые графические средства анализа.

Внедрение CASE-средств в процесс автоматизации создания ПО уменьшило количество разработчиков и повысило их профессиональный уровень в результате изъятия из их повседневного труда низкоинтеллектуальных операций. Современные CASE-средства позволяют достаточно просто создавать графические интерфейсы различных типов, легко формировать нормализованные базы данных, представляют удобный интерфейс для создания проектной документации, формируют скелет кода и т.п.

В основе любого из подходов к созданию ПО (структурный, объектно-ориентированный и т.п.) лежит определенное представление о жизненном цикле ПО. Жизненный цикл - это тот путь, который проходит ПО от зарождения идеи его создания до изъятия из эксплуатации [1]. Таким образом, жизненный цикл - это модель процесса разработки и сопровождения ПО. Одним из первых таких циклов стала модель, согласно которой процессы кодирования и разработки ПО начинаются почти одновременно. В основе такого подхода лежит представление о том, что написание программы, ее отладку и последующую аппроксимацию к фактическим требованиям пользователя необходимо осуществлять на как можно более ранней стадии. В результате отсутствия предварительных этапов анализа и проектирования ПО как продукт имеет низкие характеристики качества, а последующая модификация его крайне затруднена.

Позднее появилась так называемая каскадная модель жизненного цикла. Она предполагает наличие последовательных этапов: анализа, проектирования и реализации (значительно позже был добавлен еще один этап -

стратегического планирования). Данная модель существенно улучшила своего предшественника, однако еще не были сформированы методологии поддержки отдельных этапов и процесса разработки ПО в целом. Одной из первых методологий была структурная, которая и стала основой для возникновения ныне популярной объектно-ориентированной методологии. Каскадная модель является громоздкой и трудоемкой при создании "больших" программных систем, так как в ней реализован цикл с лавинообразным нарастанием сложности. Также эта модель несовместима с эволюционным подходом и перспективными методологиями проектирования [4].

Учитывая недостатки каскадной модели, были предложены спиральная и объектно-ориентированная модели разработки ПО. Однако их применение в автоматизации процесса создания ПО не улучшило качества создаваемых систем, так как каждое из применяемых CASE-средств ориентировано на определенный этап разработки ПО (анализ, проектирование, генерация кода и т.п.) и, следовательно, для поддержки всех этапов жизненного цикла необходимо последовательное применение нескольких CASE-средств. Взаимодействие между используемыми средствами автоматизации происходит посредством человека, что приводит к дополнительным трудностям и большим затратам на промежуточных этапах.

Следовательно, чтобы значительно повысить эффективность применения CASE-средств и ускорить создание ПО, необходима концепция сквозной поддержки всех этапов жизненного цикла при помощи одного комплекса программных средств. Подобный комплекс должен представлять собой интеллектуализированное CASE-средство, осуществляющее процесс сквозной поддержки и взаимодействия между другими CASE-средствами. Для создания такого комплекса необходимо выбрать жизненный цикл, который соответствовал бы уровню интеллектуализации системы и уровню качества создаваемого ПО.

На многих этапах разработки ПО возможно использование средств интеллектуальной поддержки, однако стройности и однообразия в этих функциях не существует. При анализе процесса создания проекта, т.е. как на самом деле происходит переход от одного этапа к другому, получена схема унифицированного жизненного цикла разработки ПО (рис. 1). Унифицируемость данной модели была достигнута путем добавления нового этапа, который контролирует этапы разработки ПО, очередность их исполнения и взаимосвязь между ними.

Особенностью цикла является выделение этапа интеллектуального анализа и эволюции. Этап же тестирования разделен на два по своей сути различных этапа: формального тестирования и интеллектуального тестирования. Формальное тестирование требует четких спецификаций и определений. На этом этапе проводится тестирование путем полного перебора возможных классов входных данных, событий, ситуаций, направлений

ветвления и т.п. К ним относятся методы «черного» и «белого ящика». Тестирование при помощи перечисленных методов может быть проведено для определенных типов языков программирования с использованием стандартных процедур анализа классов входных данных и исходного кода проекта.


Рис.1

Однако подобное тестирование не гарантирует выявления всех ошибок, касающихся логики программы, а также соответствия функционирующего программного продукта спецификациям, целям, задачам проекта, определенным на начальных этапах разработки. Ошибки подобного типа являются трудно определяемыми и наиболее существенными, поэтому для их выявления можно использовать лишь эвристические методы и сложные методы анализа. Этап тестирования при разработке ПО является наиболее важным, однако методы его формализации отсутствуют, так как в настоящее время он реализуется исключительно человеком. При сотрудничестве человека и ЭВМ на этапе интеллектуального тестирования повысится качество и эффективность работы системы в целом. Данный этап является логически обособленным по отношению к остальным этапам. Он требует для своей реализации специальных подходов и интеллектуальных средств поддержки.

В предложенном жизненном цикле разработки ПО этап эволюции, как таковой, не выделяется. В некоторых подходах к разработке ПО этап эволюции определен как постоянное совершенствование и улучшение ПО на основе результатов эксплуатации его пользователем и в соответствии с изменяющейся конъюнктурой рынка. В результате эволюционного развития ПО происходит изменение начальных требований, спецификаций, целей и задач, поэтому необходим процесс реинжиниринга (доразработки) програм-

много продукта [4]. Интеллектуальный анализ проекта устанавливает соответствие проекта текущим задачам и целям, поэтому доразработка программного продукта может быть осуществлена путем непосредственного возвращения на этап анализа и переходом на последующие этапы. Таким образом, этап интеллектуального анализа и этапы жизненного цикла ПО полностью включают в себя цели и задачи этапа эволюции.

Также выделяется этап анализа текущего состояния проекта, который наступает в каждом случае, когда необходимо принять решение о дальнейшем развитии проекта. В процессе разработки ПО переход между этапами может происходить в любом порядке, причем для каждой логически обособленной части проекта, независимо от проекта в целом. Единственным ограничением является невозможность перехода к следующему этапу без завершения соответствующих и достаточных частей работ на предыдущем этапе. Этап анализа текущего состояния проекта всегда инициирует и завершает любую транзакцию и обязательно является промежуточным при переходе от одного этапа к другому. Ранее он считался естественной составляющей при разработке ПО, так как не занимал большого количества времени. Однако при автоматизации процесса разработки ПО и широкой интеллектуализации CASE-средств этот этап является незаменимым и предполагает наибольшее число интеллектуальных усилий. Введение этого этапа устраняет также монополию человеческого звена при управлении процессом разработки.

На этапе анализа текущего состояния проекта выполняются следующие задачи:

- управление ресурсами: анализ и прогноз стоимости проекта, объем необходимых материальных и финансовых ресурсов и т.п.;
- управление штатом: статистика результативности и качества работы разработчиков, объем работ, распределение работы между разработчиками;
- управление сроками и задачами: инициализация и завершение проекта, слежение за соблюдением сроков, прогнозирование темпов развития;
- поддержка принятия решения менеджера проекта, в том числе решения о переходе от одной стадии к другой, предварительная оценка качества решений менеджера проекта и последующая оценка результатов решений;
- сбор и обработка статистики, касающейся ведения проекта.

На основе унифицированного цикла представим архитектуру интеллектуальной автоматизированной системы создания ПО, основанной на архитектуре клиент - сервер (рис.2). Основными составными частями архитектуры интеллектуального CASE-средства нового поколения являются: репозиторий, интеллектуальное ядро и набор CASE-средств. CASE-средства обращаются за данными к репозиторию и ведут себя в соответствии с правилами, определенными интеллектуальным ядром, получая от него постоянную интеллектуальную поддержку.

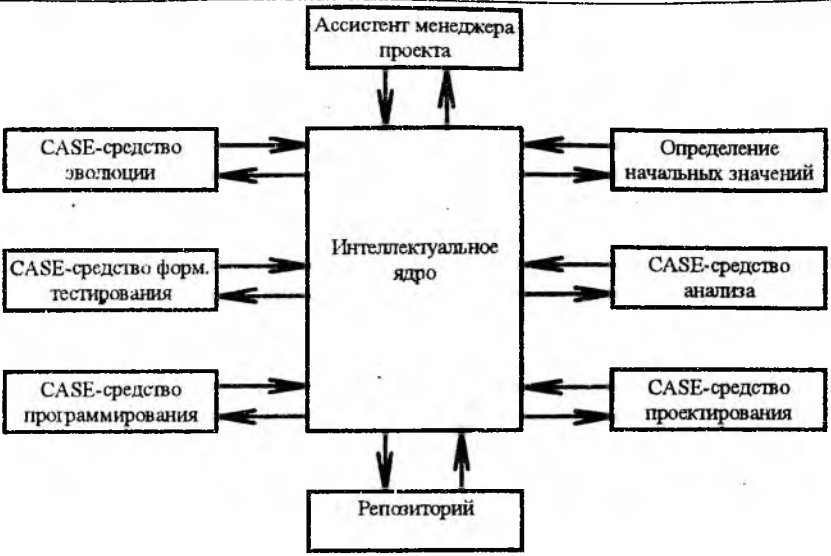


Рис.2

Для полноценной интеллектуальной поддержки управления проектом выделяется специальное CASE-средство - Ассистент менеджера проекта, которое является основным приемником результатов работы интеллектуального ядра. Остальные CASE-средства являются специализированными для каждого из этапов разработки ПО. Взаимодействие между CASE-средствами происходит через репозиторий при поддержке интеллектуального ядра. Состав CASE-средств для каждого конкретного проекта может изменяться в зависимости от объема, целей и задач проекта. Каждое CASE-средсво может иметь свои локальные данные, но учитываемыми и актуальными они становятся только после перенесения их в репозиторий.

**Список литературы:** 1. *Буч Г.* Объектно-ориентированное проектирование с примерами применения / Пер. с англ. М.: Конкорд, 1992. 519. 2. *Shlaer S., Mellor S.J.* Object-oriented systems analysis // Modeling the word in data. Prentice Hall, Englewood Cliffs, New York, 1988. 3. *Shlaer S., Mellor S.J.* An object-oriented approach to domain analysis // Software Engineering Notes. A.C.M. Press, New York, July, 1989. 4. *Bachman Ch.*, A CASE for reverse engineering, Datamation 34, 13 (July, 1988). 49-56 p. 5. *Moore J. and Bailin S.* Position Paper on Domain Analysis. // Laurel, MD: CTA Incorporated, 1988. P. 2-22.

*Поступила в редколлегию 01.10.98*