УДК 681.3
Брагинский А.А., Концевой М.М.
Научный руководитель: профессор Муравьев Г.Л.

ПОДХОДЫ, КЛАССЫ, АЛГОРИТМЫ ПРОТОТИПИРОВАНИЯ ПРИЛОЖЕНИЙ

Обучение конструированию программ предполагает выработку специальных навыков, включая спецификацию предметной области, прецедентов, проектирование архитектуры, алгоритмов, спецификацию алгоритмов, кодирование модулей на языках высокого уровня (ЯВУ), тестирование спецификаций, документирование проектных решений т.п. [1]. Существует много систем, автоматизирующих обучение. Однако конструирование программ в части работы со спецификациями, архитектурой, модульного проектирования, прототипирования, генерации каркасов слабо поддержаны обучающими средами [2-4]. Как правило, отсутствуют отладчики спецификаций, нет средств генерации текстов на ЯВУ. Оценка корректности проектных решений, спецификаций, локализация ошибок, тесно связанные с обучением конструированию программ, производится преимущественно вручную.

Качество обучения можно повысить использованием специальных компьютерных сред, подобных системам программирования, системам автоматизации проектирования, базирующимся на принципе прототипирования программ. Это обеспечивает по аналогии с указанными средствами системность обучения, исполнимость и проверяемость спецификаций проектных решений [4].

В практике разработки ПО прототипирование является этапом получения пробных версий программ и предусматривает: спецификацию требований, сценариев работы; разработку модульного каркаса, прототипа интерфейса с акцентом на отработку общего управления; – изучение прототипа, тестирование спецификаций. Это хорошо согласуется с перечисленными проблемами обучения.

Ядро средств прототипирования должно обеспечивать разработку и редактирование спецификаций модулей, схем иерархии модулей, сценариев, прототипирование интерфейсов, генерацию исполнимых шаблонов модулей, каркасов (консольных и оконных windows-приложений с упрощенным и с полноценным графическим интерфейсом), тестирование прототипов, хранение, документирование и визуализацию результатов проектирования [4, 5].

Соответственно целью работы является разработка средств построения систем прототипирования для создания прототипов-прообразов проектов приложений на модульном уровне в рамках процедурной парадигмы с ориентацией на предполагаемый тип каркаса реализации.

В работе использованы: методы прототипирования приложений объектно-оринтированных технологий, методы структурной разработки программ, модульного проектирования, нисходящей разработки (для выявления модулей, связей модулей, построения схемы иерархии модулей), методы пошаговой детализации и псевдокод (для описания тел заглушек и в качестве языка описания сценариев использования модулей). Принципы каркасного, шаблонного программирования (для автоматической верифицированной генерации прототипов приложений и обеспечения их исполнимости), сквозной структурный контроль, UML-диаграммы для описания проектных решений и моделирования алгоритмов.

Полученные результаты могут быть использованы для разработки программных средств, обеспечивающих пользователя (обучаемого, разработчика) поддержкой: проектирования модульной архитектуры ПО с использованием принципов структурной разра-

ботки; прототипирования отдельных модулей; прототипирования сценариев использования модулей и интерфейсов; тестирования отдельных действий разработчика по редактированию проектных решений и самих результатов проектирования; генерации прототипов и обеспечения их исполнимости; документирования результатов разработки, генерации отчетов и т.д.

Построена и приведена модель предметной области, спроектированная в виде иерархии классов (типа ПРОЕКТ, СЛОВАРЬ_ДАННЫХ, СТРУКТУРНЫЕ_ДАННЫЕ, МАССИВЫ, ИНТЕРФЕЙС, МОДУЛЬ, СЦЕНАРИЙ, СЦЕНАРИЙ_МОДУЛЯ, СХЕМА_ИЕРАРХИЙ и др.), обеспечивающих функциональность системы.

Предложены способы описания проектной информации, включая спецификацию задач, требований, сценариев, интерфейсов, данных, модулей, модульной архитектуры проекта-приложения, вариантов использования и т.д.

Определены виды и структура модулей-заглушек, способы описания тел модулей и их интерфейсов. Определены ситуации и способы сквозного структурного контроля принимаемых проектных решений, выбраны форматы протоколирования результатов тестирования прототипов, способы генерации консольных и оконных прототипов, определен состав отчетов, их форматы и способ генерации. Разработаны пошаговые ситуационные примеры проектирования, учитывающие варианты проектирования, перепроектирования приложений, составляющие базу тестовых примеров.

Список цитированных источников

1. Орлов, С.А. Технологии разработки программного обеспечения. – СПб.: Питер, 2004. – 527 с.

2. Касьянов, В.Н. Проблемы обучения информатике и программированию / В.Н. Касьянов // Информационно-коммуникационные технологии в образовании (IST/IMS-2001). [Электронный ресурс]. – 2001. Режим доступа: http://www.ict.edu.ru. – Дата доступа 1.02.2010.

3. Липаев, В.В. Программная инженерия. Методологические основы: учеб. / В.В. Липаев; гос. ун-т. – Высшая школа экономики. – М.: ТЕИС, 2006. – 608 с.

4. Муравьев, Г.Л. О построении систем обучения конструированию программ / Г.Л. Муравьев, В.И. Хвещук // Вестник БГТУ. – 2011. – № 5 (71). – С. 64-65.

5. Вышинская, Н.В. Прототипирование приложений на базе процедурной парадигмы // Новые математические и компьютерные технологии в проектировании, производстве и научных исследованиях: материалы 15 РНК студентов и аспирантов. – Гомель, ГГУ им. Ф.Скорины, 2012. – Ч. 2. – С. 53.

УДК 519.713

Канашук А.А.

Научный руководитель: ст. преподаватель Тузик И.В.

ПРОГРАММА ДЛЯ МИНИМИЗАЦИИ ЧАСТИЧНЫХ АВТОМАТОВ

В данной работе описаны возможности разработанной автором программы, главным назначением которой является минимизация частичных автоматов с конечным числом состояний. Программа написана на языке C++, в среде разработки C++ Builder 6 с использованием Стандартной Библиотеки Шаблонов.

На рисунке 1 представлено основное окно программы, в котором показана таблица исходного автомата, а также результат работы программы – найденные классы совместимых состояний и таблица покрывающего автомата, т.е. результат минимизации исходного автомата.

Автомат, покрывающий заданный частичный, функционирует так же, как исходный автомат, там, где функции переходов и выходов исходного автомата определены. При этом покрывающий автомат часто имеет меньшее число состояний, чем исходный авто-