ОБ ОПЫТЕ КОЛЛЕКТИВНОЙ РАЗРАБОТКИ УЧЕБНО-МЕТОДИЧЕСКИХ ПРОГРАММНЫХ СИСТЕМ

Алексей Волошин, Юрий Газин, Игорь Гридчин, Игорь Литвиненко

***Аннотация**: Рассматриваются особенности создания студенческих коллективных программных продуктов в виде учебно-методических систем. Описывается опыт создания функционального ядра как основы для создания подобных систем, а также опыт применения этого ядра в качестве основы для создания обучающе-тестирующей системы по учебному курсу «Системы и методы принятия решений». Даются рекомендации по управлению процессом разработки коллективных студенческих проектов.*

***Ключевые слова**: учебно-методические системы, коллективная разработка, студенческий проект, теория принятия решения, информационные технологии, электронное образование.*

***ACM Classification Keywords:** D.2.13 Reusable Software - Reuse models.*

***Conference**: The paper is selected from Third International Conference "Modern (e-) Learning" MeL 2008, Varna, Bulgaria, June-July 2008*

Введение

На конференции Mel-2006 были представлены [Волошин, 2006] и опубликованы [Волошин, 2007] первые результаты работы по созданию силами студенческих коллективов под руководством преподавателя учебно-методических программных продуктов на факультете кибернетики Киевского национального университета имени Тараса Шевченко. Соавторами публикаций, студентами-разработчиками, отмечалось, что, во-первых, «хотя современные программные продукты на 90% являются результатом коллективной разработки, этому на факультете нас совсем не учат», во-вторых, «создание коллективного программного продукта, даже учебного, является намного более сложной задачей, чем казалось вначале» [Волошин, 2006].

Интересно отметить, что, по свидетельству студентов-соавторов данного доклада (см. «Информацию об авторах»), за последние два года ситуация в рассматриваемом вопросе на факультете значительно улучшилась (соавтор-преподаватель питает надежду, что благодаря и его стараниям). При выполнении лабораторных работ в некоторых учебных курсах коллективные студенческие проекты уже стали обязательными, приветствуются инициативы студентов по созданию совместных программных продуктов при подготовке курсовых и дипломных работ.

На основе полученного опыта авторами при разработке обучающе-тестирующей программной системы по курсу «Системы и методы принятия решений» (СМПР) [Волошин, 2001; Волошин, Мащенко, 2006] приводятся описания всех аспектов, возникающих при создании студенческих коллективных продуктов: описание цели обучающе-тестирующей программной системы СМПР; описание общей структуры проекта; описание программной оболочки (ядра) системы; описаний программной системы СМПР; рекомендации по созданию учебно-методических программных систем на основе разработанного ядра; рекомендации по управлению процессом разработки коллективных программных продуктов.

Цель создания обучающе-тестирующей системы

Конечным продуктом коллективной разработки стала учебно-методическая система, используемая в курсе «Системы и методы принятия решений», специальность информатика, 3 курс.

Система может быть использована, как: вспомогательное средство выполнения лабораторных работ; средство демонстрации методов решения задач на лекциях курса; средства тестирования знаний; часть курса дистанционного обучения. Эти возможности представлены на Use-Case диаграмме (рис.1):

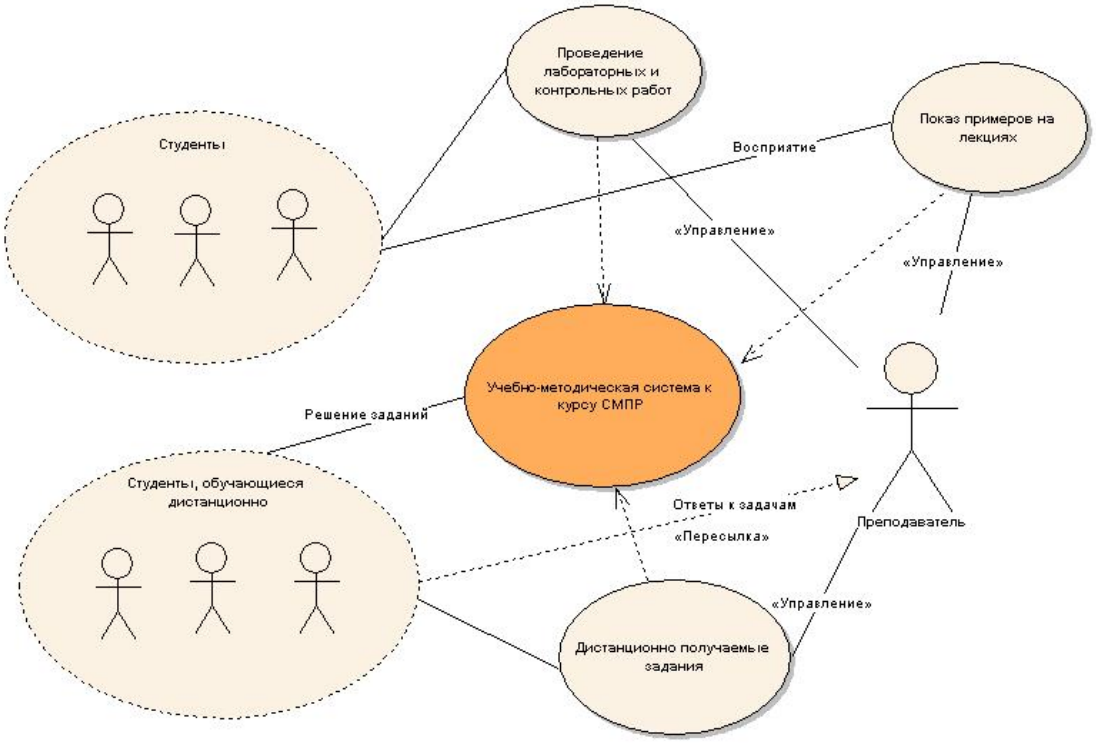

Рис. 1. Use-Case диаграмма учебно-методической системы на базе созданного ядра.

Техническое задание было сформулировано одним из авторов, читающим данный курс.

До создания этой системы предпринималось несколько попыток создания подобных систем [Волошин, 2006]. Именно «попыток», поскольку их результат не удовлетворял некоторым важным требованиям, таким как: возможность модификации системы в будущем; возможность добавления новых методов решения задач.

Общая структура проекта была определена, исходя из технического задания и структуры курса СМПР. Курс имеет множество подразделов, заданием каждого из которых является решение определенного класса задач принятия решений. Такая структура предмета явно требует от сопутствующего ей методического программного продукта свойств модульности и расширяемости.

То есть, новая система изначально проектировалась с учетом возможности будущей модификации, в плане возможности добавления новых методов и модулей. Эти требования и легли в основу создаваемого нами коллективного программного продукта.

В начале семестра из студентов, слушающих курс, были избранны три архитектора (соавторы доклада, имеющие опыт участия в разработках коллективных проектов), отвечающие за конечный результат.

Исходя из целей создания системы и технического задания, архитекторами для разработки была выбрана платформа .NET, как наиболее соответствующая этим целям, в частности, как наиболее перспективная для будущей поддержки. По предпочтениям студентов, языком разработки был выбран язык C# (хотя, следует отметить, платформа .NET предоставляет возможность разработки на нескольких языках). Выбор платформы определил требование к компьютеру, на котором учебно-методическая система: на компьютере должен быть установлен .NET Framework 2.0.

Общая структура проекта

Начальной стадией проектирования системы стало построение Use-case диаграмм, разработка ядра системы и абстрактных структур понятий – модуля и метода. Было оценено необходимое количество участников проекта, после чего в течение нескольких дней были распределены роли и обязанности участников проекта.

Ядро образует среду функционирования модулей с возможностью параллельного решения задач из разных разделов курса и обмена данными между модулями. Ядро предоставляет общие интерфейсы, стандарты обмена данными, системы помощи и информации о модулях. Также ядро включает буфер – средство обмена данными между модулями. Буфер может манипулировать исходными данными различного рода, а также результатами их обработки. Он представляет собой «среду обитания» данных уровня системы, то есть данных, к которым может получить доступ любой модуль. На архитектурном уровне буфер представляет собой класс, созданный в соответствии с паттерном Singleton [Гамма, 2007], который предоставляет модулям интерфейс загрузки, сохранения и валидации данных. Итак, буфер позволяет производить с данными уровня приложения следующие операции: сохранение данных из модулей в буфер; автоматическое сохранение результатов работы модуля в буфер после завершения его работы; загрузка данных в модуль из буфера (загрузка любого из доступных данных; загрузка данных, удовлетворяющих определенным условиям); создание пользователем новых экземпляров данных; редактирование данных; введение данных пользователем вручную (случайная генерация данных; сохранение данных буфера в файл).

На рис. 2 представлен пользовательский интерфейс работы с буфером:

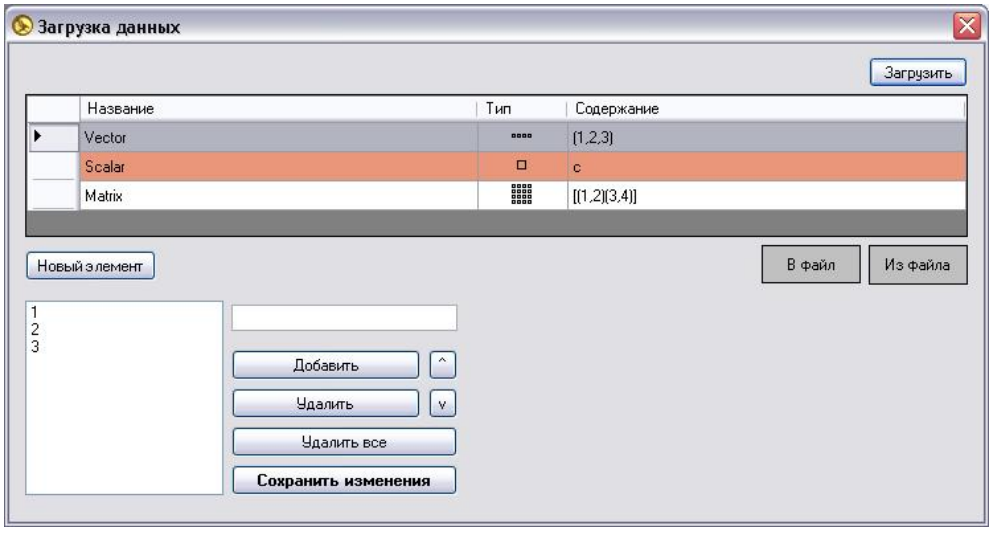

Рис. 2. Пользовательский интерфейс работы с буфером.

Определенная гибкость работы с буфером заставила установить некоторые ограничения на типы данных, подлежащих помещению в буфер. Был создан абстрактный класс – BufferData, представляющий собой единицу информации в буфере. Каждый наследник этого класса поддерживает, в частности, возможность редактирования в динамически генерируемом элементе UserControl. Как видно из рис. 2, он имеет иконку для отображения типа данных (поле «Тип» в таблице), а также может отображать свое содержимое в виде строки (поле «Содержимое» в таблице). Для учебно-методической системы по курсу СМПР было принято ограничиться тремя классами- наследниками BufferData: скаляр, вектор, матрица.

Однако важно то, что эти классы являются generic-классами. Благодаря этому, разработчик модуля имеет возможность очень просто создавать достаточно большой класс типов, работающих с буфером. Например: векторы целых чисел, матрицы вещественных, скаляры-точки, а также множество других.

Конечно же, приведенные три класса не являются ограничениями для ядра, а лишь для созданной на его базе системы. Добавление новых классов не составит больших трудностей после знакомства с архитектурой ядра.

Описанная архитектура также позволяет легко создавать методы тестирования знаний по схеме «получение исходных данных – решение задачи «вручную» – ввод решения – сравнение введенных результатов с программным решением задачи», что является достаточно удобным способом проверки навыков студентов. Разработку конкретных методов тестирования было решено предоставить разработчикам модулей, описав лишь рекомендации в техническом задании.

Результаты проектирования ядра и абстрактных структур легли в основу технического задания для руководителей групп разработки модулей. Задания содержали диаграммы абстрактных классов, описание модели модуля (словесное и в виде диаграмм).

Абстрактная схема работы отдельного модуля изображена на рис. 3.

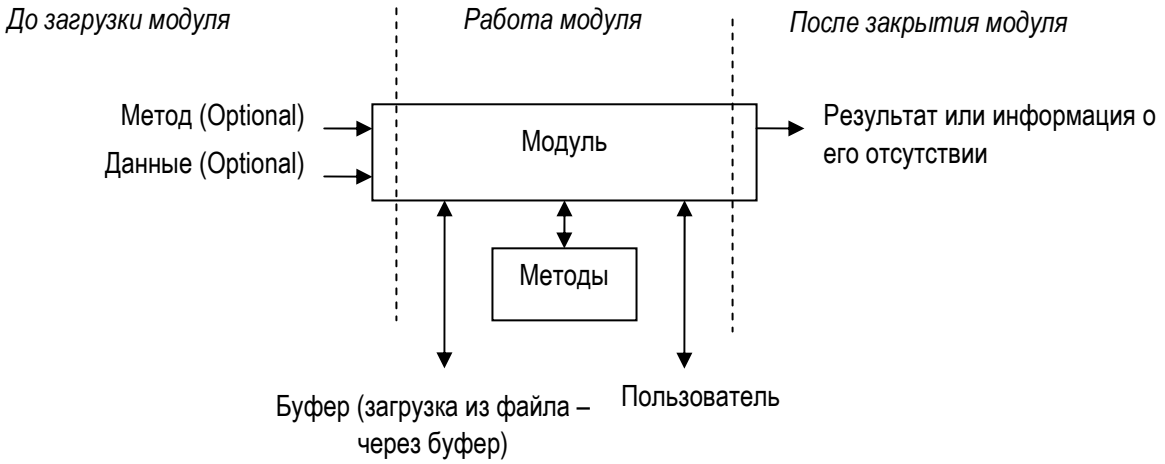

Рис. 3. Абстрактная структура модуля.

Ядро как базис для разработки обучающих систем

На начальной стадии проектирования было принято решение не специализировать никоим образом ядро системы под конкретные алгоритмы курса СМПР. Таким образом, архитектура ядра системы ориентирована лишь на структуру курса. Конкретику же решаемых задач задает лишь тот набор внешних модулей, который доступен ядру для использования.

Именно это свойство ядра системы позволяет использовать его как основу для разработки учебно-методических систем для прикладных курсов, имеющих подходящую структуру, представленную на рис. 4. Для удобства рассмотрено 2 класса задач. На самом деле их количество ограничено лишь структурой курса.

То есть, прикладной курс должен рассматривать несколько классов задач, для каждого из которых существует определенный набор методов (или алгоритмов) их решения. При этом результаты решения задач одних классов могут служить входными данными для решения задач из другого класса.

Структура модуля, отвечающего за конкретный класс задач, построена таким образом, что ядро может автоматически определить не только наличие модуля, а и некоторые иные его характеристики. Например, список реализованных методов, список авторов, разделы помощи и многое другое.

Для нахождения модулей, ядро сканирует соответствующую папку на наличие подходящих динамически подгружаемых библиотек. В них ядро ищет реализации модулей и их методов. Извлекая метаданные из

этих библиотек, можно определить множество характеристик модулей. Яркой демонстрацией преимуществ этого подхода есть динамически генерируемое меню (рис. 5).

Таким образом, для приспособления данного ядра к решению задач конкретного курса, необходима лишь реализация требуемого числа проектов, созданных по предоставляемому шаблону (о шаблоне будет сказано позже) и представляющим отдельные классы задач.

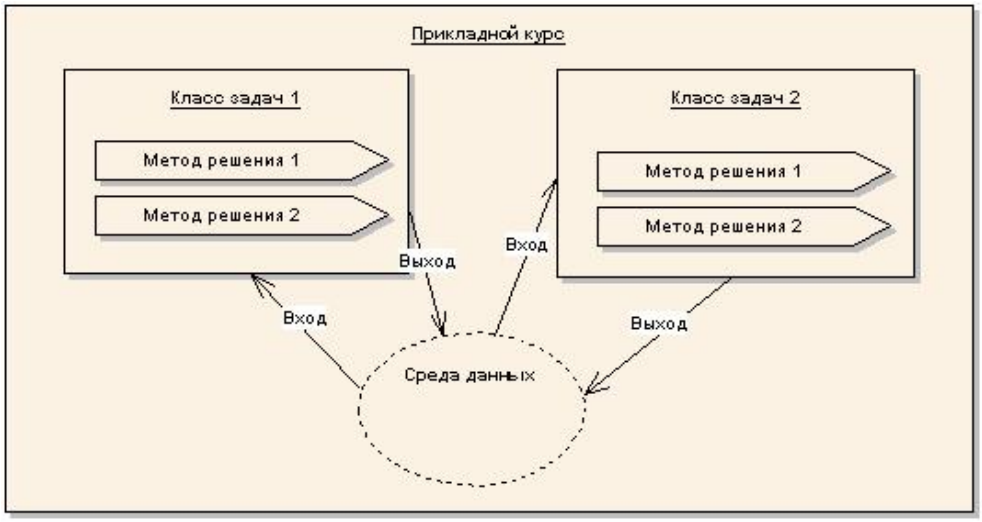

Рис. 4. Структура курса.

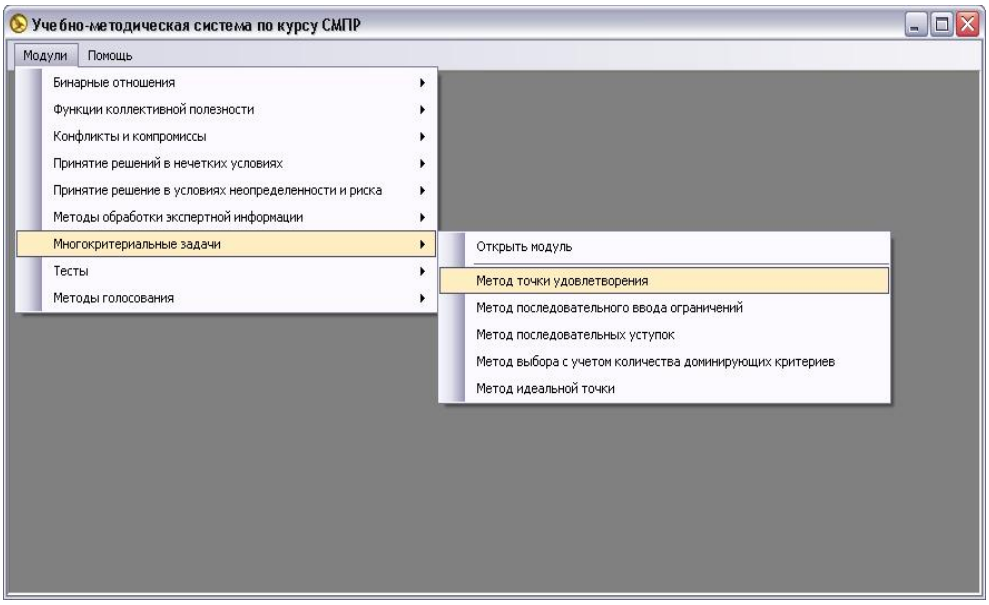

Рис. 5. Внешний вид главного меню системы.

Масштабируемость системы

Как уже было сказано, система была спроектирована с учетом расширения ее функциональности в плане добавления возможностей решения новых классов задач (масштабируемость по модулям/темам), а также методов решения существующих задач (масштабируемость по методам).

Для расширения класса решаемых задач необходимо: создать новый проект модуля, в котором реализуются заложенные на этапе проектирования абстрактные классы; скомпилировать данный проект в библиотеку; добавить готовую библиотеку в папку плагинов (дополнений) системы. После запуска программного комплекса, новая функциональность будет автоматически добавлена в главное меню

проекта. Создание проекта нового модуля возможно сразу после ознакомления с техническим заданием для руководителей проектов, однако для еще большего упрощения и ускорения процесса разработки был создан шаблон проекта модуля. Таким образом, создателю нового модуля необходимо лишь создать новый проект с использованием этого шаблона и реализовать описанные в нем классы. Добавление новых методов решения задач (расширяемость по методам) реализуется несколько иначе. Оно не требует ознакомления с общей структурой проекта, а лишь знакомства с абстрактным классом метода конкретного модуля. Однако добавление новых методов в существующий модуль требует наличия исходных кодов проекта модуля и предполагает работу в этом проекте. В каждом проекте модуля (а также и в шаблоне проекта) находится папка Methods, в которой находятся файлы классов методов решения задач данного модуля – по одному файлу для каждого метода. Собственно, добавление нового метода заключается в добавлении в эту папку нового файла, в котором будет реализован класс метода. После этого требуется перекомпиляция проекта.

Управление процессом создания системы

Организация процесса создания системы имеет трехуровневую иерархическую структуру. Во главе создаваемой системы стоят 3 системных архитектора, в обязанности которых входит организация процесса разработки, создание ядра системы, полное проектирование системы, тестирование, отладка, консультирование нижестоящих разработчиков.

На уровень ниже находятся разработчики, каждый из которых отвечает за создание отдельного модуля и организацию работы внутри своей группы.

Поскольку система состоит из ядра и множества модулей, представленных в форме плагинов, наличие или отсутствие отдельно взятого модуля не влияет на работоспособность других модулей и системы в целом.

Важным моментом является мотивация студентов к участию в проекте и созданию работоспособного модуля. К участию в проекте привлекались не все студенты потока, а лишь 20 человек, которые были, по крайней мере, знакомы с технологиями, используемыми при разработке, что составило около 40% от числа студентов потока. Так как участие в проекте являлось добровольным, то мотивацией в данной ситуации выступила возможность получить более интересное задание, чем лабораторные работы, которые делались много лет подряд, а также получить опыт участия в масштабной коллективной разработке. Немаловажную роль сыграла первоначальная ориентация на использование программного продукта в стенах учебного заведения.

Из-за большого количества модулей, а, соответственно, и участников проекта, необходимым было использование неких средств организации работы. Рассмотрев несколько вариантов, мы остановились на интернет-ресурсе http://www.assembla.com, необходимым условием работы с которым является лишь наличие доступа к Интернету. С его помощью стало возможным: распространение актуальной документации; распространение последних версий ядра системы; назначение заданий рабочим группам; получение отчетностей от руководителей групп; консультации архитекторов и преподавателя курса.

Рекомендации по созданию учебно-методических систем на базе «ядра»

Поскольку ядро может быть использовано для разработки учебно-методических систем для произвольных (соответственно структурированных) курсов. Как создатели ядра и примера такой системы, авторы считают целесообразным дать рекомендации для разработки подобных систем на базе предоставляемого ядра:

1. Использовать иерархическую систему организации работ, описанную выше.
2. Разработчикам модулей необходимо начинать знакомство с теорией своих задач с начала процесса разработки, для чего необходимо обеспечить разработчиков методической литературой (в данном

случае [Волошин, Мащенко, 2006], [Волошин, 2001]) и возможностью консультирования у преподавателя.

3. Использовать предоставленный стандартный шаблон модуля как основу для собственных модулей.
4. Использовать средства контроля версий разработки, периодически тщательно тестировать модули.
5. Строго определить сроки выполнения различных частей работы (разработка визуального интерфейса модуля, определение абстрактного метода, разработка методов).
6. Поддерживать постоянный контакт между разработчиками модулей и руководителями проекта.
7. Не пытаться следовать всем пунктам одновременно.

Заключение

На конференции Mel-2006 презентовалось учебное пособие [Волошин, Мащенко, 2006], проводилась демонстрация «нулевой» версии учебно-методической программной системы. Пособие за прошедшее время прошло апробацию в учебном процессе многих вузов Украины, было положительно оценено научно-педагогической общественностью Украины как первое достаточно полное изложение теории принятия решений для направления «Прикладная математика», специальность «Информатика». Авторы пособия планируют к концу 2008 года переиздать исправленную и дополненную версию, к которой будет приложен диск с описанными выше прикладным программным продуктом «Системы и методы принятия решений» и программной оболочкой (ядром) для создания подобных учебно-методических систем. В 2009 году планируется выпустить версию учебного пособия на русском языке.

Благодарности

Авторы выражают благодарность всем разработчикам отдельных модулей программной системы – студентам третьего курса факультета кибернетики Киевского национального университета имени Тараса Шевченко (специальность информатика), особенно руководителям групп - Лозицкому Дмитрию, Уманскому Виталию, Березану Алексею , Криволапу Андрею, Михайлову Павлу.

Список литературы

[Волошин, 2006] А. Волошин, К. Березовский., И. Дроздов. О создании коллективных учебно-методических программных продуктов по курсу «Теория принятия решений»//Труды конференции «Mel-2006»,София, 2006. - С.67-70.

[Voloshyn, 2007] O. Voloshyn, K. Berezovskiy, I. Drozdov. Developing Collective Teaching Computer Software for the Course “Decision Theory”//”Information Technologies & Knowledge”, 2007, Vol. 1, N 1.- P.33-36.

[Волошин, Мащенко, 2006] А.Ф. Волошин, С.О. Мащенко. Теория принятия решений. Учебное пособие.- Киев: Издательско-полиграфический центр «Киевский университет», 2006. - 304 с. (укр.яз.).

[Волошин, 2001] А.Ф. Волошин, С.О. Мащенко. Методические рекомендации к выполнению практических и лабораторных работ по теории принятия решений.- Киев: Издательско-полиграфический центр «Киевский университет», 2001. - 46 с. (укр.яз.).

[Гамма, 2007] Э.Гамма, Р. Хелм, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – Киев: Издательство «Питер», 2006.

Информация об авторах

Алексей Федорович Волошин *– Профессор, Киевский национальный университет имени Тараса Шевченко, факультет кибернетики; e-mail: ovoloshin@unicyb.kiev.ua*

Юрий Александрович Газин *– студент, КНУ им. Т. Шевченко; e-mail: arrrrch@mail.ru*

Игорь Юрьевич Гридчин *– студент, КНУ им. Т. Шевченко; e-mail: igorcyb@mail.ru*

Игорь Александрович Литвиненко *– студент, КНУ им. Т. Шевченко; e-mail: dev@ert.org.ua*