находиться в одном исполняемом файле, выполняться в рамках одного процесса или размещаться на одной аппаратной системе);независимость от платформы (компоненты могут выполняться на различных аппаратных и операционных платформах, взаимодействуя друг с другом в рамках единой системы); независимость от языка программирования (различия в языках, которые используются при создании компонентов, не препятствуют их взаимодействию друг с другом).

Если посмотреть на структуру объектов, то можно заметить, что всем им присущи некоторые одинаковые поля – ID. Если воспользоваться терминами DDD (Domain Driven Design), то можно выделить общий интерфейс для всех объектов.

Выводы и предложения

Произведен анализ процесса рефакторинга в структуре оптимизации программного кода. Затронут аспект применения технологии, поддерживающий концепцию распределенных объектных систем. В рамках прикладных компонентов CORBA и DCOM рассмотрен процесс осуществления рефакторинга. Такой рефакторинг сервисов, предоставляющих работу с данными, открывает возможность для рефакторинга других частей приложения, позволяет выполнить преобразования некоторых частей приложения в соответствии с полезными методиками, однако тут есть и свои минусы – трудоемкость и порой невозможность проведения таких изменений, если приложение больших размеров и большая часть сервисов уже написана.

Список литературы: 1.Операции рефакторинга базы даннях [Електронний ресурс] ИД Вильямс // С.45-49. - Название с титул. экрана. - Режим доступа: http://www.williamspublishing.com/PDF/978-5-8459-1157-5/part.pdf . 2.Методы улучшения качества кода: рефакторинг. [Електронний ресурс] - Название с титул. экрана. - Режим доступа: http://social.msdn.microsoft.com/Forums/ru-ru/fordesktopru/thread/63d9ba19-491b-4e75-9796-6de5132e1a56 3. Сравнение COM и CORBA [Електронний ресурс] - Название с титул. экрана. - Режим доступа: http://kunegin.narod.ru/ref3/corba5/12.htm 4. Robert L., Ira D., Michael Mehlich. Re-engineering C++ Component Models Via Automatic Program Transformation. - Название с титул. экрана. - Режим доступа: http://www.semanticdesigns.com/Company/Publications/WCRE05.pdf

Поступила в редколлегию 19.03.2011

УДК 519.8

А. М. МЕЗЕРНИЙ, студент, НТУ «ХПІ»
Д. В. КУКЛЕНКО, канд. техн. наук, доц., НТУ «ХПІ»

ЗАСТОСУВАННЯ КОНЦЕПЦІЇ АВТОМАТИЗОВАНОЇ ПОБУДОВИ СПЕЦИФІКАЦІЇ ВИМОГ ДО ПРОЦЕСУ ОБЛІКУ ТА ФОРМАЛІЗАЦІЇ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті розглянуто основні проблеми та актуальність управління вимогами. Запропоновано підхід до виконання обліку та формалізації вимог до програмного забезпечення на основі концепції автоматизованого генератора специфікації вимог.

В статье рассмотрены основные проблемы и актуальность управления требованиями. Предложен подход к выполнению учета и формализации требований к программному обеспечению на основе концепции автоматизированного построителя спецификации требований.

In this paper the basic problems and the actuality of requirements management is examined. Also proposed an approach for software requirements collecting and formalization is proposed, which is based on the concept of automated requirements specification builder.

Вступ. Основою успіху при створенні надійного й корисного програмного забезпечення завжди є чітке розуміння потреб його користувачів. У час, коли технології розробки програм увесь час удосконалюються й нарощують свої можливості, неправильне або неповне розуміння потреб користувачів створюваного програмного забезпечення усе ще залишається однією з причин провальних результатів проектів при його розробці.

Щоб забезпечити більш адекватний облік потреб користувачів при створенні програмного забезпечення, у рамках процесу його розробки зазвичай виділяють особливу діяльність, що зветься аналізом і формалізацією вимог, яка включає, як мінімум наступні дії:

Попередній аналіз предметної області.

Збір побажань і виявлення дійсних потреб усіх зацікавлених осіб.

Формулювання на їхній основі вимог до створюваної програми.

Фіксація отриманої інформації у вигляді ряду документів і моделей.

Фіксація всієї цієї інформації в документах і моделях повинна допомогти перейти від неясних, часто неусвідомлених, суперечливих і постійно мінливих проблем і потреб користувачів до чітко сформульованих, несуперечливих вимог, що однозначно розуміються, які вже можна використовувати для розробки програмних систем. Причому, незалежно від конкретних розроблювачів і організацій, що брали участь у їхньому створенні на основі виділених вимог, підсумкові системи повинні виходити «однаковими», тобто взаємозамінними при розв'язку будь-якого завдання. Крім того, часто потрібно, щоб різні частини або модулі таких систем могли правильно взаємодіяти один з одним, не помічаючи відмінностей між елементами, розробленими в рамках одного проекту й у незалежних проектах.

Облік та формалізація вимог. При роботі з вимогами в цей час загальноприйнято поділяти вимоги на функціональні й нефункціональні [1, 2, 3].

Функціональні вимоги визначають дії, які повинна виконувати ПС, без урахування фізичних обмежень. Тим самим вони визначають поведінку системи. Процес виявлення функціональних вимог досить складний і трудомісткий. Це пояснюється наступними причинами:

таких вимог до системи звичайно багато,

замовник не завжди здатний чітко сформулювати, чого він прагне від системи,

вимоги в підсумковому документі повинні бути викладені так, щоб вони однаково розумілися замовником і виконавцем і не допускали неоднозначності,

між функціональними вимогами можуть бути різні залежності, що ускладнюють управління ними якщо буде потреба внесення змін.

Для подолання цих труднощів застосовується моделювання вимог. Модель вимог дозволяє, по-перше, установити ієрархію вимог, що сприяє кращому розумінню людиною, по-друге, дає наочне графічне подання вимог і залежностей між ними, по-третє дозволяє зв'язати графічну форму подання з текстовою, забезпечуючи людину повною інформацією.

Нефункціональні вимоги не описують поведінку програмної системи, але описують її атрибути або атрибути оточення. Нефункціональні вимоги не потрібно включати в модель вимог, але вони повинні бути точно сформульовані. Зазвичай нефункціональних вимог не буває багато, однак вони кардинальним образом впливають на вибір архітектури системи.

Управління вимогами – це досить складний і розтягнутий у часі процес. Він триває протягом більшої частини життєвого циклу, оскільки зміни можуть вноситися як під час розробки, так і після здачі системи на етапі дослідної експлуатації й при супроводі. Причини цього полягають у тому, що вимоги:

неочевидні;

виходять із багатьох джерел;

важко формулюються (мова неоднозначна);

складаються із множини різних деталей;

нерівнозначні;

зв'язані один з одним;

лежать не тільки у функціональній області;

можуть змінюватися протягом розробки й при супроводі.

Цілями аналізу й моделювання вимог є:

досягнення угоди між розроблювачами, замовниками й користувачами про те, що повинна робити ПС;

досягнення кращого розуміння розроблювачами того, що повинна робити система;

обмеження системної функціональності;

створення базису для планування розробки проекту;

визначення користувацького інтерфейсу;

складання специфікації вимог.

Ролі людей, що беруть участь у процесі роботи з вимогами [4, 5, 6]:

Системний аналітик

Розроблювач варіантів використання

Зацікавлені особи

Експерт

Розроблювач користувацького інтерфейсу

Для досягнення поставлених цілей передбачається створення наступних документів:

Попередня угода – текстовий документ, який описує, що буде включено в ПС і що вирішено виключити, тобто, він обмежує системну функціональність.[7]

Модель вимог служить для досягнення угоди між замовником і розроблювачами, даючи можливість замовникові переконатися в тому, що

система буде робити те, що вони очікують, а розроблювачам створити те, що потрібно. Модель вимог дозволяє, по-перше, установити ієрархію вимог, що сприяє кращому розумінню людиною, по-друге, дає наочне графічне подання вимог і залежностей між ними, по-третє дозволяє зв'язати графічну форму вистави з текстової.

Специфікація вимог (Software Requirements Specification) – основний документ, використовуваний при проектуванні й розробці ПС. Вона включає модель вимог і додаткові специфікації, які являють собою текстовий опис вимог до кінцевого продукту, але не до процесу його розробки й не містять деталей реалізації вимог.

Прототип користувацького інтерфейсу забезпечує візуальне подання інтерфейсу користувача із ПС.

Глосарій – текстовий документ, що містить визначення основних понять і термінів, які повинні однаково розумітися замовником і розроблювачем.[8]

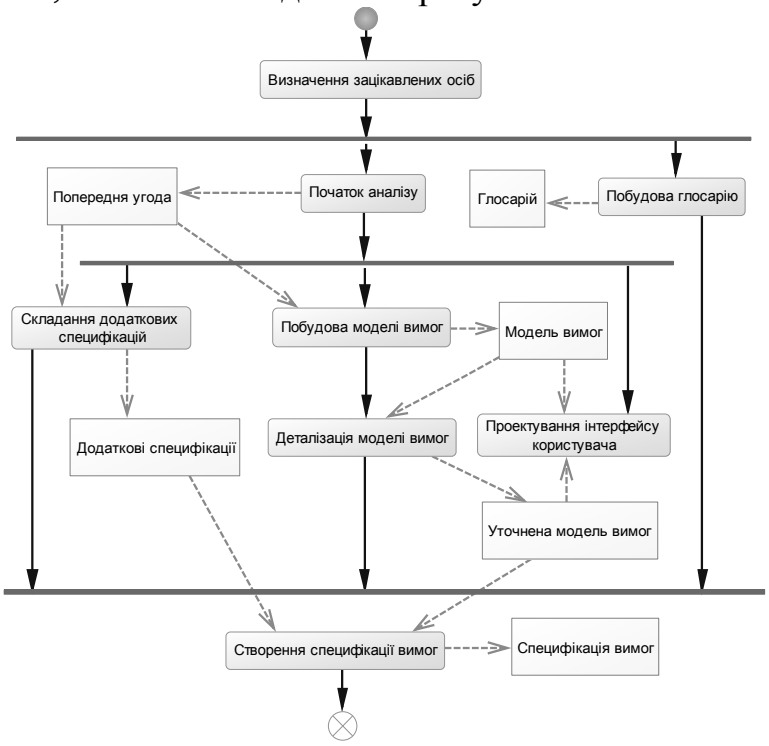

Рис. 1 – Процес обліку, аналізу та формалізації вимог

У процесі аналізу й моделювання вимог можна виділити кілька основних етапів, які представлено на рисунку 1.

Початок аналізу. Збираються побажання зацікавлених осіб до майбутньої ПС. Ці побажання аналізуються, визначаються основні властивості й границі ПС, досягаються угоди про те, які проблеми повинні бути вирішені.

Результати аналізу. Повинен бути складений документ «Попередня угода», який буде відправною крапкою для виконання всіх наступних робіт. На цьому етапі починається створення глосарія.

Побудова моделі вимог. Ця робота припускає виявлення акторів, варіантів використання й взаємодій між ними. [9]

Деталізація моделі вимог. Цілі даної діяльності:

вибір з варіантів використання характерних фрагментів, які можуть розглядатися як окремі абстрактні варіанти використання.

виявлення нових абстрактних акторів, які відіграють ролі, поділені декількома акторами;

реструктуризація моделі вимог;

детальний опис потоків подій для варіантів використання;

завдання пріоритетів варіантів використання.

Складання додаткових специфікацій. Додаткові специфікації являють собою текстові описи вимог. Вони доповнюють модель вимог і поряд з нею включаються в підсумковий документ – специфікацію вимог до ПС. [10]

Проектування користувацького інтерфейсу. Цей процес виконується для того щоб замовник міг більш точно уявити собі роботу й можливості майбутньої ПС і видати свої зауваження й уточнення вимог. Залежно від складності проекту й рівня підготовленості замовника результати цих робіт можуть бути представлені в різних формах:

програмна реалізація, що відтворює точний вид екранних вікон;

альбом екранних форм;

модель навігації екранів у вигляді діаграм класів із вказівкою атрибутів – полів і операцій – кнопок.

Створення специфікації вимог. Специфікація вимог створюється на основі моделі вимог і додаткових специфікацій. Вона затверджується керівництвом замовника й розроблювача й служить основним відправним документом для проектування й розробки. Зокрема, модель вимог, що входить у неї, надалі буде розвинена в модель аналізу й дизайну.[11]

Концепція та завдання запропонованого підходу.

Основним недоліком наведеного вище процесу є його велика тривалість, а це означає також дуже великі затрати на початковий етап, ще до початку розробки системи. Для деяких систем, що розроблюються, такий підхід є прийнятним, але існує цілий ряд систем, у яких довготривалий процес збору та формалізації вимог є неприпустимим, як з точки зору необхідного для цього часу, так й з точки зору неприпустимості використання значної кількості ресурсів для початкового етапу.

Крім того в результаті застосування наведеного вище підходу, кожен етап породжує певний набір документів, які досить складно відстежувати та обробляти як єдине ціле. Також такий підхід майже не дає можливості отримати на початковому етапі певну «первинну» специфікацію вимог, яку можна використовувати в подальшому одночасно як для поглиблення та уточнення вимог, так й для початку моделювання, розробки та виконання вимог.

Основною ідеєю розроблюваного підходу є спроба полегшити первинний збір вимог до програмного забезпечення та побудову специфікації вимог до проекту з використанням автоматизованої системи. Мається на увазі, що даний підхід повинен бути універсальним за застосовуванням для будь-яких стандартних підходів до розробки інформаційних систем, як то RUP, XP, або інші. Ідеєю даного підходу є надання розробнику ПЗ інструмента, який дозволяв би з одного боку ефективно задіяти замовників у процесі збору вимог, а з іншого, також прискорити процес початкового збору вимог, зменшити ризики, що зв’язані з недоглядом деяких вимог, автоматизувати побудову документів, зокрема специфікації вимог.

У рамках даної концепції було проведено дослідження процесу обліку та формалізації вимог до програмного забезпечення, що розроблюється. У рамках

роботи були визначити основні характеристики автоматизованої системи, що може бути впроваджена.

Сутність запропонованого підходу полягає у наданні користувачу можливості у покроковому режимі обирати можливості та особливості майбутньої системи або вказувати свої унікальні вимоги у відповідності до певного заданого стандартного шаблону. При цьому припускається також можливість виконання уточнень щодо певної можливості або особливості майбутньої системи на додаткових кроках.

Розроблена діаграма діяльності, представлена на рисунку 2, описує процес заповнення даних про функціональні та не функціональні вимоги до розроблюваної системи.

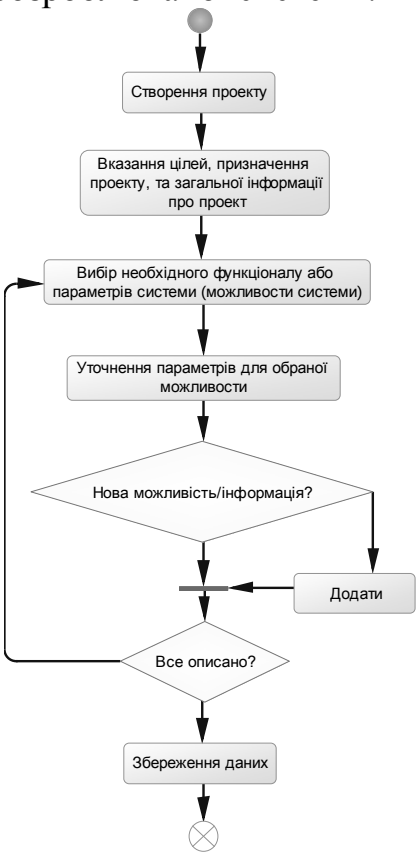

Рис. 2 – Діаграма діяльності процесу опису проекту замовником

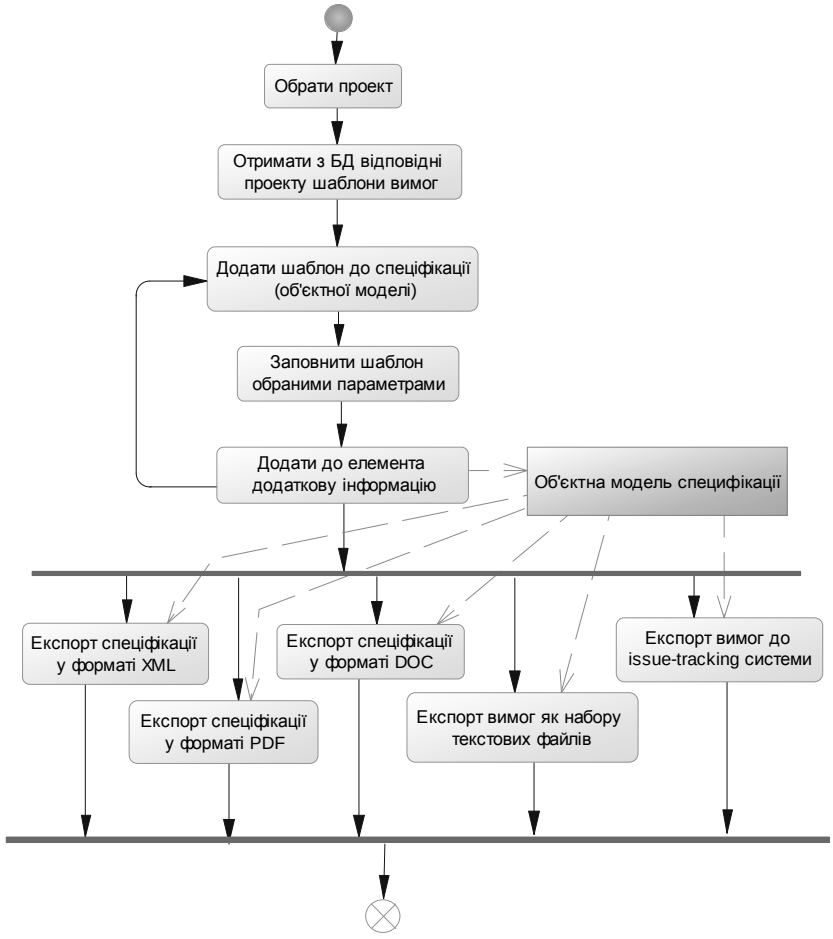

Рис.3 - Процес побудови специфікації

Дана діаграма описує взаємодію замовника з системою обліку вимог в результаті якої замовник може вказати необхідну функціональність, що повинна бути в системі, а також зазначити додаткові нефункціональні вимоги або обмеження. На основі цих даних, що зберігаються у базі даних, у подальшому можна побудувати специфікацію вимог.

В процесі розробки концепції була побудована діаграма діяльності процесу побудови специфікації, яка показує основні етапи побудови специфікації вимог до проекту. Припускається, що система, яка реалізовує описаний підхід повинна мати можливість експорту специфікацій, як у вигляді

текстового документу у форматі PDF або DOC, так і у вигляді набору документів або набору завдань для спеціалізованих систем ведення програмних

проектів Для впровадження концепції, що розглядається, необхідно створити так званий wizard-процес, який забезпечить зручну взаємодію користувача з системою у покроковому режимі з можливістю уточнення параметрів, та навіть створенням індивідуальних вимог у разі потреби.

Оскільки розроблюваний підхід повинен бути гнучким та універсальним, то не тільки дані,а й уся інша інформація щодо покрокового вибору та специфікації вимог повинна зберігатися окремо й бути легко змінюваною та настроюваною, зокрема, наприклад, додаткові параметри для кожної функціональної можливості або не функціональної вимоги. Крім того, сама послідовність та зміст кроків по створенню специфікації також мають бути настроюваними. Так наприклад можна гнучко варіювати, яким буде наступний крок – наприклад можна задати, що наступний крок – це вибір архітектури системи, або зазначення нефункціональних вимог. При цьому інтерфейс системи повинен змінюватися відповідно до тих даних, що їй відповідають.

Така гнучкість досягається завдяки двом підходам – спеціально розробленій моделі даних, яка дозволяє зберігати та змінювати більшість параметрів та налаштунків, а також завдяки компонентному підходу у побудові користувацького інтерфейсу, який дозволяє відображати ті шаблони або компоненти, які у даний момент необхідні для відображення існуючих даних та збереження вибору користувача.

Особливості та переваги запропонованого підходу. Основною особливістю запропонованого підходу є надання системи, яка забезпечує:

допомогу у формалізації мети майбутньої системи/розв'язуваних проблем;

допомогу у визначенні типу системи, що замовляється, її концепції та структури;

допомогу в оформленні (формалізації) побажань клієнтів;

можливість зазначення нестандартних вимог або системи в цілому;

наочне прототипне подання обраної клієнтом конфігурації;

надання початкової орієнтовної ціни проекту;

надання первинно формалізованих вимог для можливості швидкого запуску процесу розробки (прототипу ТЗ);

одержання уявлень про дизайн майбутньої системи;

прискорення одержання результатів від клієнта шляхом обліку потреб клієнтів, що найчастіше зустрічаються.

Висновки. У даній статті проаналізовані основні проблеми та задачі процесу обліку та формалізації системних вимог. Запропоновано підхід до виконання обліку та формалізації вимог до програмного забезпечення на основі концепції автоматизованого генератора специфікації вимог, який дозволяє значно полегшити та прискорити розробку специфікацій вимог.

Результати даної роботи можуть бути використані в стадії розробки автоматизованої системи обліку та формалізації вимог – на етапах програмної реалізації, тестування системи та дослідження характеристик розробленого програмного забезпечення, такі як дослідження ефективності, надійності, продуктивності та поліпшення підтримуваності інформаційної системи обліку та формалізації вимог.

Список літератури: 1. Стандарт IEEE 830-98. Recommended Practice for Software Requirements Specifications 2. Стандарт IEEE 1233-98. Guide for Developing System Requirements Specifications 3. Кулямін В. В. Формалізація вимог на практиці / В. В. Кулямін, Н. В. Пакулін, О. Л. Петренко // – 2006. 4. Вігерс К. Розробка вимог до програмного забезпечення / К. Вігерс // Пер, з англ. – М.:"Російська Редакція", 2004. 5. Мацяшек Л. А. Аналіз вимог і проектування систем. Розробка інформаційних систем з використанням UML. / Л. А. Мацяшек // Пер. с англ. – М.: "Вільямс", 2002 6. Коберн А. Сучасні методи опису функціональних вимог до систем. / А. Коберн // – М.: "Лорі", 2002 7. Стандарт SWEBOK- Guide to the Software Engineering Body of Knowledge, 2004 8. Халл Є. Розробка й керування вимогами. / Є. Халл,К. Джексон, Д. Дік // "Springer", 2005 9. Леффиигуэлл Д. Принципи роботи з вимогами до програмного забезпечення. Уніфікований підхід. / Д. Леффиигуэлл, Д. Уидриг Д. // Пер. с англ. – М.: "Вільямс", 2002 10. Ройс У. Управління проектами по створенню програмного забезпечення. / У. Ройс // – М.: Лори, 2002. 11. Якобсон А. Уніфікований процес розробки програмного забезпечення. / А. Якобсон, Г. Буч, Дж. Рамбо. // СПб.: Питер, 2002.

Поступила в редколлегию 16.03.2011

УДК 378.14: 004.94:62

А.Г. ЧУХРАЙ, канд. техн. наук, доцент, ХАИ**,**
С.И. ПЕДАН, аспир., ХАИ

ОБ ОДНОМ ПОДХОДЕ К РАЗРАБОТКЕ ИНТЕЛЛЕКТУАЛЬНЫХ КОМПЬЮТЕРНЫХ СРЕДСТВ ОБУЧЕНИЯ

Описан подход к графическому построению интеллектуальных компьютерных обучающих программ с помощью универсальной среды. Создаваемые программы обеспечивают выработку индивидуальной траектории обучения пользователей, адаптированной под уровень компетенции каждого из них. Проведен анализ трех контуров адаптации, таких как внешний цикл обучающих программ, педагогический сценарий выполнения их заданий и внутренний цикл каждого из них. Проведено тестирование работы среды.

Описано підхід до графічної побудови інтелектуальних комп'ютерних навчальних програм за допомогою універсальної середи. Створювані програми забезпечують формування індивідуальної траєкторії навчання користувачів, адаптованої під рівень компетенції кожного з них. Проведено аналіз трьох контурів адаптації, таких як зовнішній цикл навчальних програм, педагогічний сценарій виконання їх завдань і внутрішній цикл кожного з них. Проведено тестування роботи середи.

The approach to intelligent tutoring programs graphic construction by means of the universal environment is described. Created programs provide formation of an individual trajectory of users tutoring, which adapted under the level of their competence. The analysis of three loops of adaptation, such as outer loop of tutoring programs, pedagogical scenario of their tasks execution and inner loop of each of them is carried out. Testing of environment work is held.

Введение. К настоящему времени в Мире созданы и широко используются системы автоматизированной разработки (САР) компьютерных средств обучения (КСО). Среди таких САР одними из наиболее известных являются универсальная оболочка Moodle [1] и математическая оболочка ActiveMath [2]. Обе эти системы позволяют создавать образовательные курсы с широким спектром типов заданий и различными вариантами их наполнения. Кроме того, система ActiveMath реализует технологии интерактивного принятия решений с целью