УДК 004.052.42

Т. И. Брагина,
Г. В. Табунщик, канд. техн. наук

РИСК–ОРИЕНТИРОВАННЫЙ МЕТОД ТЕСТИРОВАНИЯ ИНТЕГРИРОВАННЫХ БАЗ ДАННЫХ

Аннотация. *Представлен риск-ориентированный метод, позволяющий сформировать тестовое покрытие для базы данных, входящей в состав интегрированной информационной системы, основываясь на априорном анализе рисков и модифицированной модели верификации базы данных. Применение данного метода даст возможность определить наиболее критичные модули и кортежи для тестирования, а также сократить затраты за счет автоматизации трудоемкого процесса верификации данных.*

Ключевые слова: *методы тестирования, базы данных, априорный анализ рисков, интегрированные системы, информационные системы, тестовое покрытие, модель данных, проверочные тесты, продукционные правила*

T. Bragina,
G. Tabunshchyk, PhD.

RISK-BASED TESTING METHOD FOR INTEGRATED DATABASES

Abstract. *Risk-based method to generate test coverage for the integrated information systems database was presented. It based on an a priori analysis of the risks and the modified databases model verification. Application of this method will make it possible to identify the most critical modules and tuples for testing, as well as reduce costs by automating time-consuming process of data verification.*

Keywords: *testing databases methods , a priori risk analysis, integrated information systems, databases, test coverage, data model, validation tests, production rules*

Т. І. Брагіна,
Г. В. Табунщик, канд. техн. наук

РИЗИК-ОРІЄНТОВАНИЙ МЕТОД ТЕСТУВАННЯ ІНТЕГРОВАНИХ БАЗ ДАНИХ

Анотація. *Представлений ризик-орієнтований метод, що дозволяє сформувати тестове покриття для бази даних, що входить до складу інтегрованої інформаційної системи, грунтуючись на апріорному аналізі ризиків і модифікованій моделі верифікації бази даних. Застосування даного методу надає можливість визначити найбільш критичні модулі та кортежі для тестування, а також скоротити витрати за рахунок автоматизації трудомісткого процесу верифікації даних.*

Ключові слова: *методи тестування, бази даних, апріорний аналіз ризиків, інтегровані системи,інформаційні системи, тестове покриття, модель даних, перевірочні тести, продукційні правила*

Введение. Актуальной задачей является интеграция различных информационных систем (ИС), что обусловлено необходимостью организации взаимодействия существующих разрозненных ИС. В процессе интеграции для поддержки согласованности, непротиворечивости, полноты и минимальной избыточности системы необходимо с ранних этапов разработки внедрять методы тестирования для контроля этих характеристик. В частности, одной из важнейших составляющих интегрированной ИС является база данных, которой необходимы средства поддержки и контроля целостности данных.

Проблемы интеграции баз данных. Сложность задачи интеграции данных повышается с увеличением сложности ИС. Источниками проблем при интеграции баз данных (БД) являются [6]:

– неоднородность источников данных (на физическом и семантическом уровнях);

– распределенный характер организации;

– повышение требований к безопасности данных;

– необходимость наличия многоуровневых справочников метаданных.

Анализ возникающих проблем позволил разделить их на следующие классы [4]: ссылочная целостность данных, коллизия изменений; корректность данных; последовательность ввода данных; структурное различие схем данных.

Из рассмотренных классов проблем наиболее актуальными и требующими разработки новых методов решения являются коллизия изменений и корректность данных,

т.е. при разработке интегрированной ИС необходимо обеспечить:

соответствие имеющейся в БД информации внутренней логике, структуре и всем явно заданным правилам ИС;

корректное параллельное изменение одних и тех же данных в двух и более информационных базах.

Решение этих задач состоит из следующих этапов [6]:

1) разработка архитектуры системы интеграции данных;

2) создание интегрирующей модели данных, являющейся основой единого пользовательского интерфейса в системе интеграции;

3) разработка методов отображения моделей данных и построение отображений в интегрирующую модель для конкретных моделей, поддерживаемых отдельными источниками данных;

4) интеграция метаданных, используемых в системе источников данных;

5) преодоление неоднородности источников данных;

6) разработка механизмов семантической интеграции источников данных.

В процессе разработки данных механизмов необходим постоянный контроль над процессом интеграции, содержимым БД и корректности использования, данных методами ИС, т.к. сложнее всего выявить неточности в логике приложения и обмене информацией между модулями и БД.

Исходя из того, что полное тестирование всех кортежей БД и модулей интегрированной системы, имеющих доступ к ним, является невозможным в связи с несопоставимыми трудозатратами на тестирование, возникает необходимость в разработке метода определения наиболее критичных кортежей БД и модулей ИС, связанных с ними, а также механизмов выбора методов тестирования для них. Учитывая, что большинство методов разработки тестов [3, 7, 8, 10] ориентируется на модель предметной области, не учитывая важность предварительной оценки рисков и учета их при составлении плана тестирования, вышесказанное приводит к задаче разработки метода тестирования БД для формирования тестового покрытия, основанного на априорном анализе рисков.

Риск-ориентированный метод тестирования интегрированных БД. Основой риск-ориентированного метода тестирования интегрированных БД является создание модели верификации БД, включающей набор правил для тестирования в зависимости от рисков, выделенных для кортежей БД и методов ИС.

Метод состоит из следующих этапов.

Этап 1. Выбор архитектуры системы интеграции.

При анализе предметной области необходимо изначально представить все слои взаимодействия между интегрированными ИС и БД для определения всех информационных процессов, которые происходят в системе, и, соответственно, возможных локализаций ошибок. Для этого необходимо выбрать архитектуру системы интеграции, наиболее распространенными из которых являются консолидация, федерализация, распространение данных и сервисный подход (Service Oriented Architecture) [1, 2, 9].

Этап 2. Создание интегрирующей модели данных.

Задачи, связанные с объединением и верификацией фрагментов БД, могут быть решены путем формирования базы знаний (БЗ) на языке представления знаний продукционного типа, например, модель верификации БЗ может быть представлена в следующем виде [8]:

$$M_{v2} =< K_B, P_2, R_2, F_2 >, \qquad (1)$$

где K_B – БЗ;

$P_2 = \{P_2^i\}$ – план верификации, состоящий из множества выбранных ошибок, где $P_2 \in E_2$, E_2 – множество всех предусмотренных ошибок;

$R_2 = \{R_{21}, R_{22}\}$ – протокол верификации, где R_{21} и R_{22} – обнаруженные и устраненные ошибки;

$F_2 = \{F_{21}, F_{22}\}$ – функции формирования R_2 на основе K_B, P_2, путем обнаружения (F_{21}) и устранения (F_{22}) ошибок.

Недостатком данной модели является то, что план верификации формируется лишь на основании ошибок E, не анализируя причины их возникновения и их критичность, а также нет возможности в явном виде связать

знания, данные БД и атрибуты интегрированных методов для составления плана тестирования, объединяющего БД и слои ИС.

Для устранения данных недостатков в модель верификации БД (1) была добавлена информация о рисках, связанных с данными БД, и их критичности, а также связь полей БД и переменных методов, в результате модель приобрела вид

$$M_{db} =< I, D, V_M, M, P, R, E, P_R, P_E, R_T >, \quad (2)$$

где I – извлеченные знания из предметной области различными методами;

D – поля (атрибуты сущностей) БД;

M – совокупность методов, имеющих доступ к БД. В случае использования N различных приложений $M = \{M_1, M_2, \ldots M_N\}$, каждое из которых содержит множество методов $M_1 = \{m_{11}, m_{21}, \ldots m_{k1}\}$;

V_M – переменные, используемые в методах M, содержащие информацию из атрибутов БД;

P – план работ с d_i, где $d_i \in D$;

R – множество возможных рисков для $d_i (d_i \in D)$, $v^i_{m^k}$, $m_k (m_k \in M)$;

E – множество ошибок, которые могут возникнуть при работе с d_i, $E = \{E_{DB}, E_M, E_N\}$, где E_{DB} – ошибки, возникшие при обращении к БД, E_M – ошибки, возникшие в M, E_N – новые выделенные возможные ошибки, которые еще не отнесли к E_{DB} или E_M;

P_R – множество продукционных правил, отражающих план действий в случае возникновения ошибок при работе с d_i, где $d_i \in D$;

P_E – множество проверочных корректировочных тестов, содержащих план обработки ошибок при работе с d_i, где $d_i \in D$; элементы множества P_E имеют доступ к данным d_i, и могут менять их формат;

R_T – результат тестирования, исходя, из которого определяется необходимость в повторном запуске используемого проверочного теста либо изменении направления тестирования, т.е. применении других тестов.

Этап 3. Построение отображения моделей данных.

Для определения отображения модели данных необходимо определить объекты информационного пространства, их представление в БД *D* и отражения в слоях ИС V_M следующим образом $\{d_i; v^i_{m^k}\}$:

- d_i – представление *i*–той характеристики объекта из множества *I*, хранящееся в БД;
- v^i_{m} – переменная, используемая в методе m^k, m^k, *M,* и содержащая информацию из d_i; в случае, использования нескольких наборов методов для различных приложений $M = \{M_1, M_2, \ldots M_N\}$ для каждого атрибута d_i устанавливается связь с каждым набором методов.

Этап 4. Интеграция метаданных.

Для интеграции метаданных необходимо:

– провести анализ рисков, которые могут возникнуть при работе с ИС, связь рисков с выделенными объектами, переменными и методами можно представить в виде

$$\{r_j; d_i; v^i_{m^k}; m^k\}; \quad (3)$$

определить e_i, e_i, *E* – множество ошибок, которые могут возникнуть при работе с d_i, e_i, E_{DB}, $E_M\}$ в случае реализации риска.

Этап 5. Преодоление неоднородности источников данных.

Для контроля и преодоления неоднородности источников данных необходимо:

– создать план тестирования p_i для проверки элементов d_i , v^i_m или m^i , для которых на предыдущем этапе были выделены риски r_j, r_j, *R*.

– создать P_E, p^j_E, P_E – множество проверочных корректирующих тестов, содержащих план обработки ошибок e_i, E_{DB}, $E_M\}$ при работе с d_i, где $d_i D$.

Этап 6. Разработка механизмов семантической интеграции источников данных.

В качестве механизма семантической интеграции был выбран механизм логических правил следующего вида.

1. Если во время выполнения теста p_i, возникла ошибка из множества E_{DB} и/или E_M, необходимо выполнить проверочный тест p^j_E:

$$if\,(result(p_i) = failed \cap \cap (e_i \in E_{DB} \cup e_i \in E_M) \Rightarrow p_E^j. \quad (4)$$

2. Если во время выполнения теста p_i, возникла непредвиденная ошибка e_N, т.е. не принадлежащая множеству E_{DB} или E_M, необходимо добавить новый вид ошибки в E_N. Через определенный период времени тестировщик должен рассмотреть данную ошибку, определить с какими элементами d_i, v^i_m или m^i она связана, перенести e_N из множества E_N в E_{DB} или E_M, написать новый проверочный тест p_j, создать новый риск r_j, затем связать новые элементы $\{r_j;\ d_i\ ;\ v^i_{m^k}\ ;\ m^k\}$ и создать логическое правила вида:

$$if\,(result(p_i) = failed \cap (e_N \notin E_{DB} \cap e_N \notin E_M) \Rightarrow$$
$$\Rightarrow e_N \in E_N \Rightarrow$$
$$\Rightarrow (move(e_N, E_{DB}) \cup move(e_N, E_M)) \cap$$
$$\cap create(p_E^j, r_j, \{r_j; d_i; v^i_{m^k}; m^k\}) \Rightarrow p_E^j.$$

3. Если при прохождении теста p_i не возникли ошибки, необходимо занести в r_T^i статус «*checked*», иначе, в r_T^i заносится код полученной ошибки;

4. Если тест p_i проверяет кортеж данных, для которых установлено r_T^i статус «*checked*» и не выделено критичных рисков, выполнение данного теста пропускается:

$$if\,(r_T^i = checked \cap (d_i; r_i) \in 0) \Rightarrow \Rightarrow result(p_i) = passed. \quad (5)$$

5. Если тест p_i или метод m_i обновляет кортеж данных, для него необходимо установить r_T^i статус «*not tested*».

Риски, выделенные на первом и четвертом этапе, влияют на процесс тестирования и позволяют принимать управляющие решения для автоматизации составления плана тестирования.

Представленный метод создания модели верификации БД отличается наличием связей множества проверочных корректировочных тестов, содержащих план обработки ошибок P_E, с рисками для атрибутов сущностей БД.

Данная связь позволяет автоматически определять наиболее критичные кортежи данных и методы тестирования, направленные на контроль наиболее вероятных рисков, с целью исправить некоторое множество известных ошибок E_{DB}, E_M}, а также получить информацию о неисследованных ошибках. Рассмотрим подробнее процесс разработки данных тестов.

Метод разработки проверочных риск-ориентированных корректирующих тестов. Предлагаемый авторами метод разработки проверочных риск-ориентированных корректирующих тестов, ориентируется на опыт компании IBM [5] и отличается ориентацией на априорную оценку рисков. Он предназначен для извлечения, структурирования и формализации знаний из различных источников с целью автоматизировать процесс обработки ошибок при работе с интегрированной БД.

Метод анализа граничных значений [8] применяется к значениям атрибутов объектов, факторам уверенности и точности для проверки соответствия ограничениям, устанавливаемым по умолчанию и уточняемым в каждом сеансе тестирования. Основой для генерации проверочных корректирующих тестов служат фрагменты БЗ, сформированные из БД при анализе ее экспертами и по результатам проведенных ранее серий тестирования. Для сокращения перебора при генерации тестов применяется группировка правил по использованию атрибутов, переменных и методов. Для структурирования и формализации знаний разработан набор функций обработки единого объекта, элементы которого состоят из представлений вида «атрибут, переменная, функция» $\{d_i,\ v^i_{m^k},\ m^k\}$, соответствуют утверждениям, извлекаемым из эксперта или БД, и связаны друг с другом с помощью правил и рисков, выделенных для них.

Метод состоит из следующих шагов.

1) Определение граничных значений для атрибута d_i, и связанной с ним переменной $v^i_{m^k}$;

2) Определение возможных рисков и ошибок, возникающих в результате реализа-

ции данных рисков, при работе с d_i и $v^i_{m^k}$; каждый риск априорно оценивается двумя характеристиками: вероятность возникновения p^i_r и влияние (критичность) c^i_r, а также определяется, может ли изменяться уровень этого риска в процессе тестирования s^i_r и на какую величину $\{a^i_r; e_j\}$, т.е. создается правило вида

$$if(e_j \cap s^i_r) \Rightarrow p^i_r = p^i_r + a^i_r / k, \qquad (6)$$

где k – количество протестированных записей между возникновениями ошибки e_j.

3) Составление автоматического теста, анализирующего состояние d_i или $v^i_{m^k}$ и корректирующего его формат для соответствия его требования.

4) Выполнение проверочного корректирующего теста.

5) Анализ эффективности разработанного теста.

6) Анализ возможности применения данного теста к другим атрибутам d_j и переменным $v^j_{m^k}$.

Данный метод необходимо применить ко всем данным, для которых определены риски и возможные ошибки, с целью написания проверочных корректирующих тестов. Каждый тест может быть использован для одной и более переменной (атрибута, метода), каждой переменной (атрибуту, методу) может быть написано один и более проверочных тестов. База тестов должна регулярно пополняться за счет анализа новых, ранее не проанализированных ошибок (этап 6 риск-ориентированного метода тестирования, формула (4)).

Применение риск-ориентированного метода тестирования интегрированных БД. Предложенный метод был использован для тестирования корректности взаимодействия БД приемной комиссии с БД «Єдиної державної електронної бази з питань освіти» («ЄДЕБО»). Задача состояла в контроле процесса интеграции БД, созданной с использованием системы управления БД MS Access, и БД, созданной на базе технологии Oracle.

При создании тестовых сценариев для проверки корректности интеграции для локальной БД и «ЄДЕБО» были выделены основные и вспомогательные сущности и оценена мощность множества тестируемых атрибутов D из (2) – $|\mathrm{D}|=450$. Для каждого атрибута d_i были определены граничные значения и условия на значение, а также переменные v^i_m, используемые в методах интегрированной ИС в следующем виде, например:

– $d_{10}['name'] = "pasnum"$;

– 2 варианта граничных условий:

1) для граждан Украины: $d_{10}['\min'] = 000000$, $d_{10}['\max'] = 999999$;

2) для нерезидентов $d_{10}['\min'] = 0$, $d_{10}['\max'] = 9999999999$;

- имя переменной, используемой в интеграционных методах, соответствует имени данного атрибута в «ЄДЕБО» $v^{10}_m = "\mathrm{PasportNumber}"$.

Следующим этапом было определение рисков и вероятных ошибок для атрибутов, например, для номера паспорта:

– выделен риск «некорректный ввод данных», вероятность возникновения $p^{10}_r = 0{,}2$, критичность $c^{10}_r = 1$ (критичность была оценена по шкале [0; 1]);

– определены возможные ошибки e_{10} = {«значение не соответствует граничным условиям», «паспорт не найден», «персона с указанным паспортом уже существует»};

– установлено $s^{10}_r = true$, что означает, что вероятность возникновения риска может изменяться в процессе тестирования на величину $a^{10}_r = 1$ в случае, если получена ошибка «значение не соответствует граничным условиям», т.е. было создано правило вида:

$$if(e_{10}[1] \cap s^{10}_r) \Rightarrow p^{10}_r = p^{10}_r + a^{10}_r / k.$$

Автоматические проверочные тесты были написаны с помощью языка *Visual Basic for Application* и запросов *SQL*. Например, для исключения вероятности

некорректного переноса из текстового формата в числовой, было написано несколько тестов, среди них следующий запрос «UpdatePassNum1»:

UPDATE MAIN
SET MAIN.pasnum = "0" & [pasnum]
WHERE (((Len([pasnum]))=5)).

После выполнения проверочных корректирующих тестов для верификации 700 записей в локальной БД были получены результаты R_T из (2), представленный тест «UpdatePassNum1» обнаружил и исправил 40 записей номеров паспортов. При передаче данных в «ЄДЕБО» были обнаружены две ошибки e_{10}[2] =«паспорт не найден». Следовательно, эффективность приведенного для примера теста составила 95%, т.е. в 95 % случаев применение корректировочного теста привело к исчезновению ошибок e_{10}= {«значение не соответствует граничным условиям», «паспорт не найден», «персона с указанным паспортом уже существует»}. В остальных случаях, ошибкой являлась механическая опечатка при вводе цифр, что невозможно исправить автоматически.

Вероятность возникновения риска «некорректный ввод данных» увеличился до $p_r^{10} = 0{,}25$ для d_{10}, но уменьшен для v_m^{10} $p_r^{10}(v_m^{10}) = 0$, так как данные, связанные с этой переменной были проверены и по возможности исправлены, следовательно, нет необходимости их повторно тестировать.

Количество реализованных тестов для проверки информации в локальной БД равно 197, для проверки корректности данных полученных из «ЄДЕБО» – 253 и для проверки данных, используемых в методах для обмена информацией – 69. При тестировании 700 кортежей в основных сущностях локальной БД с использованием стандартных методов тестирования [8] тестовое покрытие составило 519 тестов и было затрачено 2, 3 часа. При использовании риск-ориентированного метода тестирования интегрированных количество тестовый сценариев в тестовом покрытии составило 482 теста при временных затратах 1,94 часа. Количество выявленных ошибок в обоих случаях составило 120 некорректных данных в значениях атрибутов локальной БД.

Из вышеописанного можно сделать вывод, что применение предложенного риск-ориентированного метода тестирования интегрированных БД позволило снизить временные затраты на 17 % за счет сокращения тестового покрытия, сохранив уровень выявленных ошибок на допустимом уровне (0,1 % ошибочных данных по отношению к общему числу данных в БД).

Выводы. Научная новизна представленного риск-ориентированного метода тестирования интегрированных баз данных заключается в формировании тестового покрытия и плана тестирования на основании априорного анализа рисков для кортежей БД и методов ИС, а также в возможности корректировать начальную оценку рисков, используя результаты тестирования. За счет возможности автоматически формировать тестовое покрытие на основе анализа рисков сокращается количество используемых тестовых сценариев и, как следствие, снижаются временные затраты на тестирование.

Таким образом, разработанный метод и модель верификации БД можно использовать в нескольких направлениях:

– использование информации о выделенных рисках позволит определить наиболее критичные модули и кортежи для тестирования;

– использование проверочных корректирующих тестов позволит автоматизировать трудоемкий процесс приведения данных к необходимому формату и непротиворечивости;

– постоянный контроль новых неизученных ошибок позволит расширить базу тестов и, как следствие, снизить одну из самых распространенных причин ошибок в приложении – появление непредвиденных и, следовательно,

неисследованных комбинаций входных данных.

Практическое применение предложенного риск-ориентированного метода тестирования интегрированных БД позволило снизить данные временные затраты на 17 %, сохранив уровень не выявленных ошибок на допустимом уровне (0,1 % ошибочных данных по отношению к общему числу данных в БД).

Список использованной литературы

1. IBM: Patterns: Implementing an SOA Using an Enterprise Service Bus [Электронный ресурс] / Режим доступа: http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf (01.07.2004)

2. Microsoft Integration Patterns [Электронный ресурс] / Режим доступа: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=12474 (08.05.2004)

3. Владимиров М. А. Критерии полноты тестового покрытия в генетических алгоритмах генерации тестов / М. А. Владимиров. – М. : Труды Института системного программирования РАН. – 2006. – № 3. – Т. 9. – С. 57 – 66.

4. Давыдов А. Проблемы интеграции данных различных информационных баз / А. Давыдов [Электронный ресурс] // Режим доступа: http://infostart.ru/public/192353/ (27.06.2013)

5. Документация решения Rational для коллективного управления жизненным циклом. Тестирование [Электронный ресурс] / Режим доступа: http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/index.jsp?nav=%2F24 (01.01.2007)

6. Когаловский М. Р. Методы интеграции данных в информационных системах [Электронный ресурс] / М. Р. Когаловский // Режим доступа: http://www.cemi.rssi.ru/mei//articles/kogalov10-05.pdf (01.05.2010)

7. Липаев В. В. Тестирование компонентов и комплексов программ: Учебник В. В. Липаев – М. : СИНТЕГ, 2010. – 400 с.

8. Рыбина Г. В. Планирование процедур верификации баз знаний в интегрированных экспертных системах / Г. В. Рыбина, В. В. Смирнов // Инженерная физика. – М. : – 2006. – № 3. – С. 53 – 65.

9. Хоп Г. Шаблоны интеграции корпоративных приложений / Г. Хоп, Б. Вульф – М.: Издательский дом «Вильямс». – 2006. – 672 с.

10. Хамбл Д. Непрерывное развертывание ПО: автоматизация процессов сборки, тестирования и внедрения новых версий программ / Д. Хамбл, Д. Фарли – М. : Издательский дом «Вильямс». – 2011. – 432 с.

Получено 01.03.2014

References

1. IBM: Patterns: Implementing an SOA Using an Enterprise Service Bus url:http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf (01.07.2004)

2. Microsoft Integration Patterns. url:http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=12474 (08.05.2004)

3. Vladimirov M.A. Kriterii polnoty testovogo pokrytiya v geneticheskikh algoritmakh generatsii testov [Completeness of Test Coverage Criteria in Test Generation Genetic Algorithms], (2006), *Trudy Instituta sistemnogo programmirovaniya, RAN*, Moskow, Russian Federation, No. 3, Vol. 9, pp. 57 – 66 (In Russian).

4. Davydov A. Problemy integratsii dannykh razlichnykh informatsionnykh baz [Data Integration's Problems from Different data Bases]. (27.06.2013) (In Russian), url: http://infostart.ru/public/192353/

5. Dokumentatsiya resheniya Rational dlya kollektivnogo upravleniya zhiznennym tsiklom. Testirovanie [Documentation Rational Solutions for Collective Management Lifecycle. Testing]. (01.01.2007) (In Russian) url: http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/index.jsp?nav=%2F24.

6. Kogalovskii M.R. Metody integratsii dannykh v informatsionnykh sistemakh [Integrating data Methods in Information Systems] (01.05.2010), (In Russian), url: http://www.cemi.rssi.ru/mei//articles/kogalov10-05.pdf.

7. Lipaev V.V. Testirovanie komponentov i kompleksov programm [Software Components and Systems Testing], (2010), *Uchebnik, SINTEG*, Moskow, Russian Federation, 400 p. (In Russian).

8. Rybina G.V., Smirnov B.B. Planirovanie protsedur verifikatsii baz znanii v integrirovannykh ekspertnykh sistemakh [Planning Procedures of Verification of Knowledge Bases in Integrated Expert Systems], (2006), *Inzhenernaya fizika*, Moskow, Russian Federation, No. 3, pp. 53 – 65 (In Russian).

9. Khop G., Vul'f B. Shablony integratsii korporativnykh prilozhenii [Enterprise Integration Patterns], (2006), *Williams*, Moskow, Russian Federation, 672 p. (In Russian).

10. Khambl D., Farli D. Nepreryvnoe razvertyvanie PO: avtomatizatsiya protsessov sborki, testirovaniya i vnedreniya novykh versii program [Reliable Software Releases through Build, Test and Deployment Automation], (2011), *Williams*, Moskow, Russian Federation, 432 p. (In Russian).

Брагина
Татьяна Игоревна,
аспирант каф. программных средств Запорожского нац. технического ун-та,
ул. Жуковского 64, Запорожье, Украина, 69063.
Телефон: 7698226
E-mail:
bragina.zntu@gmail.com

Табунщик
Галина Владимировна,
канд. техн. наук, доц. каф. программных средств Запорожского нац. технического ун-та,
ул. Жуковского 64, Запорожье, Украина, 69063.
Телефон: 7698267
E-mail:
galina.tabunshchik@gmail.com