Teorija i praktika. SPb.: SPbGUP, 352.**7.** *Cysar', A.* (2005). Internet–tehnologii v «Mystery Shopping». Zhurn. upravlenie personalom. Otdel kadrov, 5, 50–53. **8.** *Kapusta, I.* (2003). Podderzhivat' zakonnye ozhidanija potrebitelej. Zhurn. Marketing v Rossii i za rubezhom, 9, 71–77. **9.** Jenciklopedija marketinga. Internet-proekt «Jenciklopedija marketinga» (2014). Available at: http:// www.marketing.spb. ru/libmm/sales/serv_quality.htm. **10.** Ocinjuvannja jakosti turistichnih poslug (2013). Jalta-Kiev: ATM Ukrainy, 208. **11.** *Akao, Y.* (1996). Quality Function Deployment on Total Quality Management and Future Subject. (QFD and TQM Series No. 1 (Japanese). Quality Control, Vol. 47, № 8, 55–64.**12.** *Lapidus, V. A.* (2000). Vseobshhee kachestvo (TQM) v rossijskih kompanijah. Tipografija Novosti, 432.

Надійшла (received) 28.05.2014

УДК 665.9

Р. А. ГАМЗАЕВ, канд. техн. наук, ассистент, НТУ «ХПИ»;
Н. В. ТКАЧУК, д-р техн. наук, проф., НТУ «ХПИ»;
И. О. МАРТИНКУС, аспирантка, НТУ «ХПИ»;

МЕТА-МОДЕЛЬ ПРОЦЕССА ТРАССИРОВКИ ТРЕБОВАНИЙ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Рассмотрены особенности управления требованиями к программному обеспечению (ПО) и обоснована актуальность проблемы разработки методологических основ для построения мета-моделей процессов трассировки требований. Проанализированы некоторые существующие концептуальные модели трассировки требований и на основе их обобщения предложена унифицированная мета-модель процессу трассировки требований с использованием фреймовых спецификаций. Приведен пример возможности технологической реализации этой модели в гибком процессе разработки ПО по методологии Scrum.

Ключевые слова: программное обеспечение, трассировка требований, мета-модель, фрейм, гибкая методология Scrum.

Введение. Решение проблемы повышения эффективности процессов инженерии требований является одним из основных факторов успеха при использовании любой методологии разработки ПО: начиная с классических моделей жизненного цикла (ЖЦ), таких как каскадная и спиралевидная и заканчивая получившими распространение в настоящее время новыми подходами к разработке, такими как, например, RUP (Rational Unified Process) или XP (eXtreme Programming). Это связано с тем, что как показывает опыт выполнения реальных проектов по созданию сложных программных систем, именно ошибки на этапе анализа и моделирования требований становятся наиболее критичными с точки зрения их влияния на сроки и затраты всего проекта в целом и, с другой стороны, именно сбор и анализ требований является наиболее трудоемким и слабо формализуемым этапом в ЖЦ разработки ПО [1]. Существующие методы инженерии требований, такие как проведение интервью с различными участниками проекта, составление вербальных описаний требований и т.д., позволяют либо весьма приблизительно и субъективно вести обработку соответствующей информации, либо предполагают достаточно специфические и абстрактные подходы, сложные для понимания конечных пользователей и

заказчиков ПО, которые должны быть активными участниками процессов сбора и анализа требований. Поэтому весьма актуальной является проблема поиска рационального методологического компромисса в вопросах моделирования и управления требованиями, который должен дать возможность представить эти механизмы в достаточно структурированной и в тоже время наглядной, визуальной форме, позволяющей получать количественные оценки состояния определенного требования как в процессе разработки ПО.

Важным компонентом такого механизма управления требованиями является их трассировка (requirements traceability), которая представляет собой, по существу, процесс взаимного отображения множества требований $R = (r_1, r_2, ..., r_n)$ в множество проектных файлов $F = (f_1, f_2, ..., f_m)$, под которыми следует понимать любые артефакты, которые создаются разработчиками в ходе реализации целевой ПС: это могут быть фрагменты исходного кода, структуры хранения данных, дизайн интерфейса пользователя, файлы проектной документации и т.п. (рис. 1).

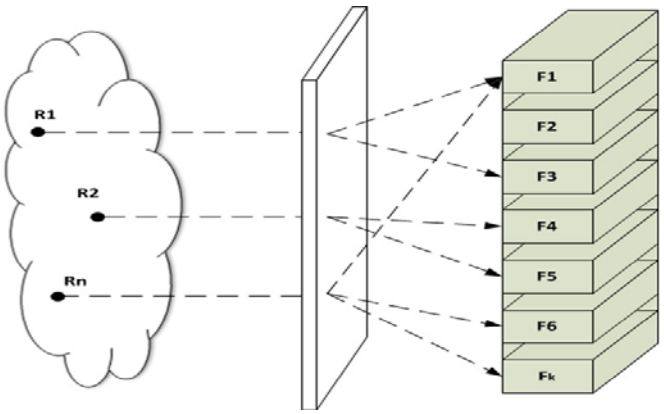

Рис. 1 – Концептуальная схема механизма трассировки требований

Существует несколько более специальных определений понятия трассировки требований, например, в стандарте IEEE [1] говорится о том, что ”....это способность программного обеспечения устанавливать связь между требованиями и их реализацией, с учетом особенностей разработки и области выполнения”, а в работе известного украинского специалиста в области программной инженерии проф. Е.М. Лаврищевой [2] указано на то, что“… трасування вимог, тобто розвиток і обробка вимог із простежуванням ідентифікованих зв'язків, що повинні бути зафіксовані за двома напрямками – від потреб до робочих продуктів і навпаки…”. В [3] предложена классификация основных существующих способов моделирования процессов трассировки требований, а именно: матрицы трассировки (traceability matrix), ассоциативные правила (association rule), паттерны трассировки (traceability pattern), и приведены некоторые соображения относительно их преимуществ и недостатков. Они представляют собой, по сути, уже технологическую реализацию различных процедур трассировки требований, и для анализа и поиска путей повышения их эффективностинеобходимо исследование их механизмов на более высоком, концептуальном уровне рассмотрения этих процессов.

Поэтому целью данной статьи является разработка методологического подхода и построение унифицированной мета-модели процессов трассировки требований, позволяющей создавать на ее основе новые, более эффективные модельно-технологические инструменты для управления этими процессами, которые могли бы адаптироваться с учетом особенностей конкретного проекта по разработке ПО, а именно: уровня семантической сложностьи предметной

области, особенностей выбранной методологии управления проектом, используемых программных средств и т. п.

Анализ существующих подходов к моделированию процессов трассировки требований. Одна из первых моделей процессов трассировки требований (МПТТ) была предложена в [4], ее схема показана на рис. 2.

Анализ этой МПТТ позволяет сделать вывод о том, что основными компонентами в концептуальной схеме трассировки требований выступают такие сущности как: “Лицо-владелец требований”, “Объект” и “Источник”, которые связаны такими базовыми отношениями как: “выполняет роль”, “трассирует”, “управляет” и “документирует” (documents). Авторы этой МПТТ называют ее эталонной (reference) моделью трассировки, что предполагает возможность использования ее концептуальных компонентов (сущностей и отношения между ними) могут быть использованы в других МПТТ, построеннных на ее основе.

К категории концептуальных МПТТ следует отнести и итерационную схему процесса трассировки требований в гибкой (agile-) разработке ПО из [5], которая приведена на рис. 3.

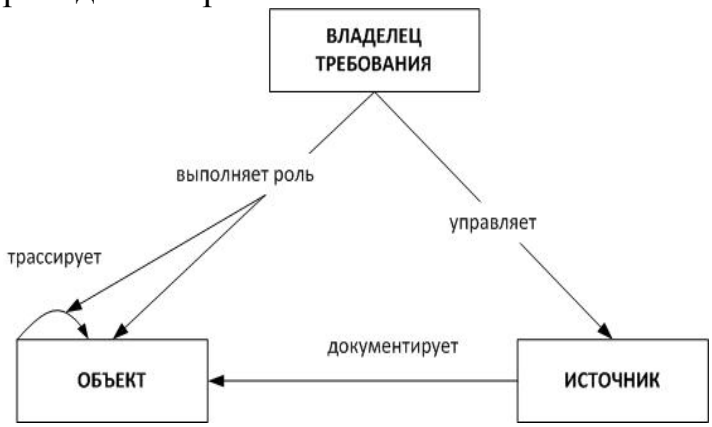

Рис. 2 – Эталонная МПТТ из работы [4]

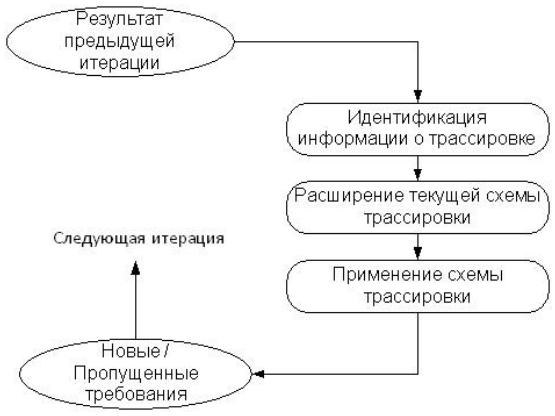

Рис. 3 – Итерационная МПТТ из [5]

Из нее следует, что трассировка требований выполняется в некотором циклическом процессе (блок «Следущая итерация») с контуром обратной связи (блок «Результат предыдущей итерации»), причем соответствующие действия по идентификации данных о трассировке (блок «Идентификация информации о трассировке») приводят к расширению текущей схемы трассировки (блок «Расширение текущей схемы трассировки»), и затем, в результате ее применения (блок «Применение схемы трассировки», в процесс трассировки вовлекаются новые / либо ранее не рассмотренные требования (блок «Новые/Пропущенные требования»).

Непосредственное взаимовлияние процессов трассировки и технологий обеспечения качества программного обеспечения, в частности, подхода SPICE (Software Process Improvement Capability dEtermination), отражает МПТТ, предложенная в работе [6], концептуальная схема которой показана на рис. 4.

Основное преимущество этой модели, по мнению ее авторов, заключается в том, что она интегрирована в многоуровневую модель всего процесса разработки ПО по методологии SPICE, в соответствии с которой, на первом этапе на основе спецификаций пользователей (*ENG.1:Требования пользователей*) вначале

выполняется этап разработки общесистемных спецификаций (*ENG.2: Спецификации системных требований*), затем – этап высокоуровневого (архитектурного) проектирования всей целевой системы (*ENG.3: Системный дизайн*), далее, с учетом его результатов, происходит переход к этапу разработки спецификаций к ПО (*ENG.: Требования к ПО*), и только после этого выполняется этап непосредственного проектирования ПО (*ENG.3: Дизайн ПО*).

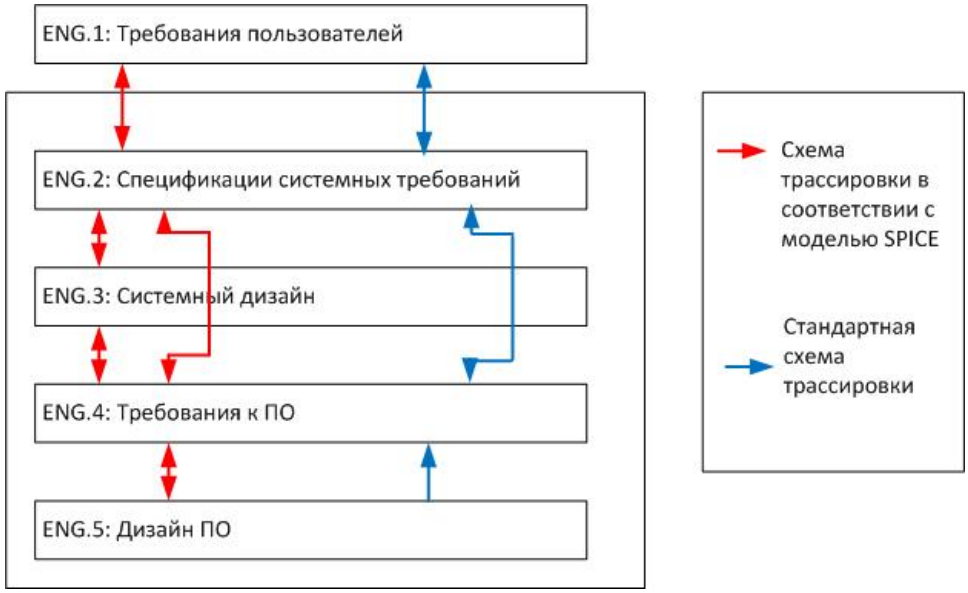

Рис. 4 – Интегрированная МПТТ для процесса SPICE [6]

Анализируя методологические принципы разработки концептуальных МПТТ, следует также упомянуть о том, что во многих работах по этой проблематике подчеркивается, что для повышения эффективности процессов трассировки необходимо применять различные интеллектуальные методы обработки первичной текстовой информации о требованиях. Так, например, в работе [7] предлагается многоуровневая схема трассировки первичной текстовой информации о требованиях (multi-level textual traceability) с применением их модельно-управляемой обработки (model-driven engineering) и методов информационного поиска (information retrieval). Важность использования в МПТТ именно методов обработки слабоструктурированной текстовой информации подчекивается также в работах [8, 9], в которых рассмотрены вопросы трассировки текстовых спецификаций и проектных артефактов с применением, соответственно, специального аннотированого словаря событий (annotated phenomena vocabulary) и метода латентно-семантического индексирования (latent semantic indexing).

Мета-модель процесса трассировки требований на основе фреймовых спецификаций. Таким образом, в результате проведенного выше краткого анализа некоторых методологических аспектов построения различных МПТТ, можно сделать следующие выводы:

целью процесса трассировки, и соответственно, целью разработки и применения МПТТ, является установление как можно более точной (полной) семантической связи между множеством требований к ПО и множеством тех проектных артефактов (файлов), которые создаются в процессе разработки ПО с целью реализации этих требований;

процесс трассировки требований, на концептуальном уровне его моделирования, представляет собой некоторую траекторию T в многомерном

информационном пространстве, координаты которого определяются как подмножество декартова произведения $T \subset R \times F \times S$, где R - это множество собственно требований к ПО, которые должны быть реализованы в конечной программном продукте; F – множество проектных артефактов (файлов), которые создаются для реализации этих требований; S – множество проектных итераций, которые выполняются в ходе разработки данной ПС, что позволяет рассматривать процесс трассировки динамически, с учетом фактора времени);

процесс трассировки должен выполняться итерационно, в некотором цикле, с учетом наличия обратной связи, которая позволяет на каждой последующей итерации из множества S уточнять семантические связи между элементами множеств R и F;

для непосредственной реализации механизма отображения множества R в множество F могут применяться такие средства как матрицы трассировки, ассоциативные правила и паттерны трассировки (см. выше);

поскольку изначально требования из множества R, как правило, представлены в виде текстовых описаний на естественном языке, то для повышения эффективности всего процесса трассировки в целом, необходимо предварительно применять определенные логико-лингвистические методы обработки этих текстов с целью устранения в них избыточности информации, разрешения возможных логических противоречий в описании требуемых свойств конечного программного продукта и т.п.

На основе этого качественного анализа методологических аспектов построения различных МПТТ можно предложить формализованное представление для целого класса таких моделей, для чего целесообразно использовать понятие фрейма (frame) [10], поскольку именно фреймовые спецификации могут быть эффективно использованы для представления сложных структурированных данных (знаний), соответствующих определенным стереотипным ситуациям, в том числе и при моделировании различных информационных процессов и систем.

В соответствии с известным представлением фрейма как структурированной совокупности слотов, каждый из которых имеет свой допустимый набор значений, а именно:

Frame: "*Name*" =
{
Slot_1: "Name"= (List of Values…),
Slot_2: "Name"= (List of Values…),
…….…
…….……
Slot_N: "Name"= (List of Values…)
}

представим фрейм "Модель процесса трассировки требований (МПТТ)" в следующем виде (рис. 5)

Очевидно, что используя общее определение фрейма и его интерпретацию в терминах возможных значений для каждого из соответствующих слотов, показанных на рис. 5, можно вполне корректно определить, как каждую из ранее

рассмотренных частных моделей трассировки (рис. 2–4), так и сформировать некоторый новый тип такой модели, задавая определенную конфигурацию конкретных параметров отдельных слотов Slot _1 - Slot _4.

Frame: ***МПТТ***

{

Slot_1: "*Структура информационного пространства трассировки P*" = ("множество требований ***R***"x "множество файлов ***F***"x "множество проектных сессий ***S***" x ….)

Slot_2: "*Способ построения траектории трассировки в пространстве P*" = («прямая трассировка», «обратная трассировка», «итерационный процесс»)…

Slot_3: *"Механизм отображения связей между компонентами"* = («матрица трассировки», «ассоциативные правила», «паттерны трассировки»…)

Slot _4: *"Метод подготовки исходных данных о требованиях"*= *(*«текстовые спецификации», «графические схемы», «логико-лингвистические методы» ……*)*

}

Рис. 5 – Фреймовая спецификация для МПТТ

Пример технологической реализации мета-модели трассировки требований в гибком процессе разработки ПО по методологии Scrum. В работе [11], на основе рассмотренной выше макро-модели МПТТ, разработана алгоритмическая модель процесса построения адаптивной матрицы трассировки и соответствующая информационная технология, обеспечивающая возможность реализации сфокусированного интерфейса разработчика, которые могут быть применены в схеме автоматизированного управления гибкой разработкой ПО с применением методологии Scrum, которая представлена на рис. 6. В ней, дополнительно и в контексте двух уже существующих и хорошо апробированных на практике организационных циклов выполнения типового Scrum -проекта:

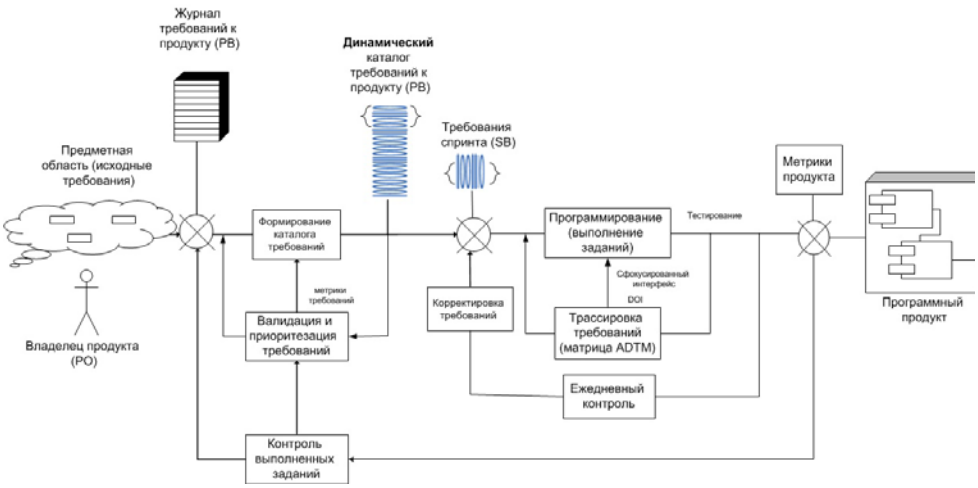

Рис. 6 – Схема автоматизированного управления Scrum-процессом разработки ПО

– цикл построения общего каталога требований программного продукта (Product Backlog - PB) и его дальнейшей обработки путем формирования каталога требований для одной проектной итерации или одного спринта (sprint) в терминологии Scrum (Sprint Backlog - SB),

– цикл выполнения последовательности ежедневных задач, которые необходимы в рамках одной проектной итерации.

Введено 2 новых технологических контура управления с обратной связью, а именно [11]:

– контур управления процедурами приоритезации и оценки качества требований (requirements priority and quality estimation), что делает формирование так называемого динамического каталога требований РВ;

– контур управления трассировкой требований в процессе непосредственного программирования (выполнение задач проекта), с целью обеспечения эффекта сфокусированного интерфейса разработчика ПО с применением предложенной адаптивной матрицы трассировки ADTM.

Результатом функционирования этой схемы является программный продукт, который должен соответствовать определенным значение метрик качества ПО (рис. 6). Предложенная схема управления выполнением Scrum - проекта позволяет применять знания - ориентированные методы разработки ПО, то есть такие, что накапливают и используют в дальнейшем для принятия проектных решений количественные оценки о состоянии выполнения отдельных этапов проекта и качества полученных при этом результатов. Тестирование этого подхода, полное описание методики проведения которого и полученные при этом результаты приведены в [11], подтвердило его эффективность

Выводы и направления дальнейших исследований. В данной научной статье предложена знание-ориентированная мета-модель процесса трассировки требований, для построения которой использован механизм составления соответствующих фреймовых спецификаций и приведен пример возможности технологической реализации этой модели в гибком процессе разработки ПО по методологии Scrum. В дальнейшем планируется развить данный подход путем дополнения этой мета-модели процесса трассировки требований такими логическими компонентами и технологическими механизмами их реализации, которые в настоящее время предлагаются в новой концепции разработки сложных программных систем, получившей название предметно-ориентированного проектирования или проектирования, управляемого доменными моделями (domain-driven development - DDD) [12].

Список литературы: **1**. *Соммервил, И.* Инженерия программного обеспечения [Текст] / *И. Соммервил* ; пер. с англ. – М.: Вильямс, 2002. – 624 с. **2.** *Лаврищева, Е.М.* Методы и средства инженерии программного обеспечения [Текст] / *Е. М. Лаврищева, В. А. Петрухин* – Москва: МФТИ, 2007. – 415 с. **3.** *Tkachuk M. V.* Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development [Текст] / *TkachukM.V., R.O.Gamzayev, H.C.Mayr, V.O. Bolshutkin* – Проблемы программирования. – К.: НАН України. – 2012. – № 2–3 (спец. выпуск). – С.252 – 260. **4**. *Ramesh B.* Toward Reference Model for Requirements Traceability [Текст] / *B. Ramesh, M. Jarke.* – IEEE Software Engineering – 2001. –#27(1) – P.58–93. **5**. *Taromirad M.* Agile Requirements Traceability Using Domain-Specific Modelling Languages [Текст] / *M. Taromirad, R. Paige* – Extreme Modeling Workshop – 2012. – P. 45-50. **6**. *Turban B.* An Integrated Decision Model For Efficient Requirement Traceability In SPICE Compliant Development [Текст] / *B. Turban* – Intelligent Solutions in Embedded Systems – 2007. – P. 273 – 286. **7**. *Sannier N.* Towards Multi-level Textual Requirements Traceability Using Model-Driven Engineering and Information Retrieval [Текст] / *N. Sannier, B Baudry* – Proceeding of 2nd International Workshop on Model-Driven Requirements Engineering – 2012. – P. 29 – 38. **8**. *Ladenberger L.* Requirements Traceability between Textual Requirements and Formal Models Using ProR [Электронный ресурс] / *L. Ladenberger, M. Jastram.* – Режим доступа к статье: http://www.stups.uni-duesseldorf.de/mediawiki/ images /f /f7/ Pub-LadenbergerJastram_iFMABZ2012.pdf. **9**. *Lormans M.* Can LSI help Reconstructing Requirements Traceability in Design and Test? [Текст] / *M. Lormans,*

A. van Deursen – Proceedings of the 10th European Conference Software Maintenance and Reengineering –2006 –P. 46– 56. **10**. *Минский М.* Фреймы для представления знаний [Текст] / Минский М.; пер. с англ. – М.: Энергия, 1978. – 151 с. **11**. *Гамзаев Р.О.* Модель та інформаційна технологія побудови адаптивної матриці трасування вимог в гнучких процесах розробки програмного забезпечення [Текст] / *Р.О.Гамзаев, М.В.Ткачук* – В*ІСНИК НТУ «ХПІ».* – Харків, 2013 – № 2 (976). – С. 49 – 60. **12** *Эванс, Э.* Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем. [Текст] / *Э. Эванс* – Пер. с англ. – М. «ООО И.Д. Вильямс», 2011.

Bibliography (transliterated): **1**. *Sommerville, I.* (2002). Software Engineering; 6–th edition. Moscow, Williams, 624. **2**. *Lavrishcheva, E. M., Petruhin, V. A.* (2007). Methods and tools for software engineering. Moscow: MIPT, 415. **3**. *Tkachuk M. V., Gamzayev R.O., Mayr H.C., Bolshutkin V.O.* (2012). Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development. Programming problems. K.: National Academy of Sciences of Ukraine, № 2-3 (spec. issue), P.252 – 260. **4**. *Ramesh B., Jarke M.* (2001). Toward Reference Model for Requirements Traceability. IEEE Software Engineering , 27(1) , P.58–93. **5**. *Taromirad M., Paige R.* (2012). Agile Requirements Traceability Using Domain-Specific Modelling Languages. Extreme Modeling Workshop,P. 45-50. **6**. *Turban B.* (2007). An Integrated Decision Model For Efficient Requirement Traceability In SPICE Compliant Development. Intelligent Solutions in Embedded Systems, P. 273 – 286. **7**. *Sannier N., Baudry B.* (2012). Towards Multi-level Textual Requirements Traceability Using Model-Driven Engineering and Information Retrieval. Proceeding of 2nd International Workshop on Model-Driven Requirements Engineering , P. 29 – 38. **8**. *Ladenberger L. , Jastram M.* Requirements Traceability between Textual Requirements and Formal Models Using ProR [Электронный ресурс] /–http://www.stups.uni-duesseldorf.de/mediawiki/ images /f /f7/ Pub-LadenbergerJastram_iFMABZ2012.pdf. **9**. *Lormans M., A. van Deursen* (2006). Can LSI help Reconstructing Requirements Traceability in Design and Test?.Proceedings of the 10th European Conference Software Maintenance and Reengineering, P. 46– 56. **10**. *Minsky M.* (1978). Framework for Representing Knowledge .151. **11**. *Gamzaev R.O., Tkachuk M.V.* (2013). Models and information technology of adaptive matrix trace requirements in flexible software development processes. Journal of NTU "KPI", Kharkiv, № 2 (976), P.49 - 60. **12.** *Evans E.* (2004) Domain-Driven Design - Tackling Complexity in the Heart of Software, Addison-Wesley,P. 529.

Поступила (received) 28.05.2014

УДК 658.512

Л. И. НЕФЁДОВ, д-р техн. наук, проф., зав. каф., ХНАДУ, Харьков;
Ю. А. ПЕТРЕНКО, д-р техн. наук, доц., зав. каф., ХНАДУ, Харьков;
А. С. КОНОНЫХИН, аспирант, ХНАДУ, Харьков

МОДЕЛИ ВЫБОРА КАДРОВОГО ОБЕСПЕЧЕНИЯ ОФИСА В УСЛОВИЯХ НЕЧЕТКОЙ ИНФОРМАЦИИ

В статье разработана модель выбора кадрового обеспечения, которая позволяет сформировать персонал офиса с учетом затратных критериев и квалификационных, образовательных и психологических качеств претендентов заданных размыто.

Ключевые слова: кадровое обеспечение, квалификационные показатели, психологические характеристики, бизнес-процесс, нечеткая информация.

Введение. На сегодняшний день важным аспектом организации работы в офисе является формирование его персонала и его взаимодействие. Очень важно,