УДК 681.518:658.512

Н. В. ТКАЧУК, д-р техн. наук, проф. каф. АСУ НТУ «ХПИ»;
К. А. НАГОРНЫЙ, асп. каф. АСУ НТУ «ХПИ»;
И. О. МАРТИНКУС, студентка НТУ «ХПИ»

МЕТОДИКА ОЦЕНКИ ДИНАМИЧЕСКОЙ СЛОЖНОСТИ ТРЕБОВАНИЙ В ПРОЦЕССАХ СОПРОВОЖДЕНИЯ УНАСЛЕДОВАНЫХ ПРОГРАММНЫХ СИСТЕМ

Предложен поход к постановке задачи анализа сложности унаследованных программных систем (УПС) в процессе их сопровождения, который комплексно рассматривает такие многомерные факторы как структурная сложность УПС и динамика изменения состояния требований (Т). Предложен алгоритм ранжирования Т в пространстве критериев «Приоритет» - «Функциональная сложность», что позволяет в дальнейшем определить отдельные типы УПС в пространстве их признаков «Ранг Т» – «Структурная сложность УПС» и эффективно проводить выбор инструментальных средств для их структурной адаптации.

Запропоновано підхід до постановки задачі складності успадкованих програмних систем (УПС) у процесі їх супроводу, який комплексно розглядає такі багатовимірні фактори як структурна складність УПС та динаміка зміни стану вимог (В). Запропоновано алгоритм ранжування В у просторі критеріїв «Пріоритет» – «Функціональна складність», що дозволяє у подальшому визначити окремі типи УПС у просторі таких ознак як «Ранг В» – «Структурна складність ПС» та адекватно проводити вибір інструментальних засобів для їх структурної адаптації.

An approach to the complex estimation of complication legacy software systems (LSS) complexity in the process of maintenance is proposed, which provides complex consideration of such multidimensional factors as structural complexity of PS and changing's dynamics of the requirement's (R) state. The ranking algorithm for R in the criteria space «Priority» – «Functional complexity» is proposed, which allows in future to define the different types of PS in the attribute space «Rank of R» – «Structural complexity of LSS» and to provide adequately choice of tools for structural adaptation of PS.

1. Введение. Актуальность проблемы и постановка задачи исследования. Согласно стандарту ISO 12207:2007 процесс сопровождения программной системы (ПС) – это процесс модификации существующей системы с целью сохранения её целостности [1]. Он инициируется после передачи разработанной ПС ее пользователям и включает любые действия, направленные на устранения программных ошибок, неточностей реализации бизнес-логики, повышения производительности и т.д., с учетом к постоянно изменяющихся требований. Эта стадия жизненного цикла ПС наиболее трудоемкая и затратная и, по некоторым оценкам [2], занимает около 55% всех затрат жизненного цикла ПС. Если в ходе эксплуатации ПС происходит моральное старение ее проектных решений и технологий, но она продолжает представлять ценность для компании, поскольку реализует важную функциональность и аккумулирует накопленные за годы работы ресурсы данных и знаний, то такую ПС называют унаследованной программной системой (УПС) [3].

Большинство современных УПС разработаны и продолжают разрабатываться на основе объектно-ориентированной парадигмы (ООП), поэтому при их сопровождении возникают проблемы, связанные как с необходимостью адаптации к воздействиям внешней среды, так и в целом характерные для парадигмы ООП (см., напр., в [4]):

- колебания уровня вычислительной нагрузки на ПС;
- постоянно изменяющиеся требования;
- сильная связность (high coupling) программных компонентов;
- наличие компонентов, выполняющих большое количество разнородных функций (низкая степень сцепления – low cohesion);
- сложность повторного использования кода;

и некоторые другие.

Таким образом, задача сопровождения УПС представляет собой актуальную и достаточно сложную научно-техническую проблему. Для ее решения существует несколько типов адаптивных технологий разработки архитектуры ПС и управления ее компонентами, классификация которых приводится в [4]. В соответствии с предложенным в этой работе кибернетическим подходом к решению задачи структурной адаптации УПС, для выбора эффективного управляющего воздействия в процессе целенаправленного изменения структуры системы под влиянием внешней среды (т.е. с учетом изменяющихся требований), необходимо корректно определять такие 2 комплексные характеристики сопровождаемой УПС как:

1. Структурная сложность программного обеспечения системы,
2. Динамика изменения требований в процессе ее функционирования.

В работе [5] была разработана и исследована концепция многомерного информационного метапространства для адаптивной разработки ПС, которая, в частности, предполагает построение нескольких взаимосвязанных локальных подпространств, где происходит накопление и анализ данных о требованиях, моделях и методах проектирования систем. Предлагаемый ниже подход также использует эту концепцию.

2. Многомерный подход к оценке типов унаследованных программных систем. Очевидно, что совокупность определенных значений характеристик (1) – (2), приведенных в предыдущем разделе, позволяет определить несколько различных типов УПС, что в свою очередь, необходимо для дальнейшего эффективного выбора соответствующей технологии для модификации структуры УПС [6]. Для дальнейшего корректного рассмотрения предлагаемого подхода введем ряд определений.

Определение 1. *Тип системы (TS)* – это качественная характеристика УПС, задаваемая парой значений:

$$TS = \langle StructuralComplexity, RequirementRank \rangle, \quad (1)$$

где *StructuralComplexity* – структурная сложность ее программного обеспечения (ПО);

RequirementRank – ранг требований, которые должны быть учтены в процессе сопровождения УПС.

Определение 2. Структура программной системы (S) – это формальный объект, который представляется кортежем двух множеств:

$$S = \langle \underline{C}, \underline{R} \rangle, \quad (2)$$

где $\underline{C} = \{c_i\}, i = \overline{1,n}$ – множество компонент (классов) УПС;

$\underline{R} = \{r_i\}, j = \overline{1,m}$ – множество связей между компонентами.

Следует отметить, что для оценки структурной сложности ПО в программной инженерии существует множество подходов и метрик программного кода, начиная от более простых характеристик – например, количество строк исходного кода и инфодинамические метрики Холстеда [7], и заканчивая комплексными показателями сложности взаимодействия компонентов ПО на уровне классов и пакетов (см. напр., в [8]). Эта задача имеет самостоятельное значение в разрабатываемом подходе и будет более подробно исследована в других публикациях.

3. Определение понятия «Ранг требования» и подход к его оценке. Для того, чтобы предложить адекватный подход к определению понятия «Ранг требования» – см. выражение (1), – прежде всего необходимо уточнить, что понимается по требованием к ПО в контексте данной работы.

Определение 3. *Требование (Т)* – это документированные условия или возможности, которыми должна обладать программная система в целом или ее отдельные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам [9].

Требования разделяются на два вида: *функциональные* (functional requirements) и *не функциональные* (non-functional requirements), и четыре уровня детализации: бизнес-требования (business requirements), пользовательские (user requirements), системные (system requirements), проектная спецификация (software requirements specification, SRS) [3,9]. Под требованиями будем понимать функциональные требования (ФТ), определяющие функциональность ПС, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований [9].

Опыт работы авторов в реальных проектах показывает, что для корректной работы с требованиями необходимо учитывать два важных параметра: *приоритет выполнения ФТ* и *сложность ФТ*. *Ранг требования* – это кортеж, состоящий из пары значений:

$$RequirementRank = \langle Priority, Complexity \rangle, \quad (3)$$

где *Priority* – приоритет выполнения ФТ;

Complexity – сложность ФТ.

Определение 4. *Приоритет выполнения ФТ (*далее *приоритет ФТ)* – выражение относительной важности требования по отношению к системе либо другим требованиям [10]. В системах управления требованиями (СУТ): IBM Rational Requisite Pro, CalibreRM, Accompa, Gatherspace, *приоритет выполнения ФТ* и *сложность ФТ* часто характеризуются лингвистическими значениями. Например для приоритета значения будут: *«низкий», «средний», «высокий»*, однако в некоторых СУТ существует и более подробная градация приоритета, например: *«без приоритета», «низкий», «нормальный», «высокий», «срочный», «немедленный»*. Приоритет ФТ определяется на основе экспертных оценок заинтересованных лиц (экспертов предметной области).

В формализованном виде *приоритет ФТ* можно представить как множество значений, лингвистической переменной [11]. Лингвистическая переменная L – это набор:

$$L = \langle X, T(X), U, G, M \rangle, \quad (4)$$

где X – название переменной;

$T(X)$ – терм-множество переменной X, т.е. совокупность её лингвистических значений;

U – универсальное множество числовых значений;

G – синтаксическое правило, порождающее термы множества $T(X)$, бесконтекстной грамматики;

M – семантическое правило, которое каждому лингвистическому значению X, ставит в соответствие его смысл $M(X)$, причем $M(X)$ обозначает нечеткое подмножество множества U.

Смысл лингвистического значения X характеризуется функцией совместимости $c: U \to [0,1]$, которая каждому элементу $u \in U$ ставит в соответствие значение совместимости этого элемента с X [11].

В качестве функции совместимости предлагается использовать функцию принадлежности Харрингтона [12]. Она определяется следующей формулой:

$$d = e^{-e^{-R}}, \quad (5)$$

где d – значение шкалы функции совместимости;

R – значение лингвистической шкалы L.

Точками перегиба являются: $d(0) = 1/e \approx 0.37$ и $d(0.78) = 1 - 1/e \approx 0.63$.

Применение функции Харрингтона является не единственной возможностью для решения данной задачи, однако оно мотивировано результатами наблюдений за некоторыми характерными признаками реальных УПС, а так же обладает такими свойствами как непрерывность, монотонность и гладкость. Кроме того, эта кривая хорошо передает тот факт, что в областях желательностей, близких к 0 и 1, «чувствительность» ее существенно ниже, чем в средней зоне [12].

Для *приоритета ФТ* лингвистическая переменная L, имеет вид:

$X : Priority$;

$T(Priority) = \{нейтральный, актуальный, немедленный\}$;

U – универсальное множество (пустое, т.к. оценка изначально качественная, однако для определения функции совместимости используется кодированная шкала, для выражения мнения экспертов [-6;-4;-2;0;2;4;6]) [12].

G – синтаксическое правило, порождающее термы множества $T(X)$, контекстно-свободной грамматики;

M – функция совместимости Харрингтона: $d = e^{-e^{-R}}$.

Определение 5. *Сложность ФТ* – это показатель уровня затрат, необходимых для реализации данного требования [10]. В СУТ *сложность ФТ* обычно принимает значения: высокая, средняя, низкая (high, medium, low). В стандарте IEEE 830-1998 [13] приведены восемь основных характеристик качества требований, в список этих характеристик дополнен, т.о. требование должно, быть *корректным* (correct), *недвусмысленным* (unambiguous), *полным* (complete) и т.д. [14].

Количественные метрики *сложности ФТ* на основе приведенных характеристик приведены в [15]. Метрики спецификации NASA [16] позволяют рассчитывать качество ФТ на основе наличия в спецификации тех или иных лингвистических конструкций, например: «должен», «будет», «предполагается», «возможно» и т.д. Следует отметить ряд походов направленных на оценку функциональности требований: *измерение объема функциональности* (Functional Size Measurement, FSM) концепция которого представлена в стандарте ISO/IEC 14143.1:1998 [17], *метод приближенных баллов Вариантов Использования* (Unadjusted Use Case Points – UNC) основан анализе сценариев Use Case [18], *метод ранних функциональных баллов* (Early Function Points Analysis – EFPA). Для расчета сложности требования предлагается использовать метод ранних функциональных баллов, это модификация метода функциональных баллов (Function Points Analysis – FPA), он апробирован на достаточно большом количестве проектов разных масштабов и применяется на ранних стадиях разработки (этап сбора и анализа требований), для прогнозирования сложности будущей ПС [19, 20]. Детально, этот метод будет рассмотрен в следующих публикациях. Результат, полученный с помощью этого метода – это количество функциональных баллов приходящихся на оцениваемое ФТ.

Для дальнейших расчетов на основе полученных количественных оценок, для *сложности ФТ* строится лингвистическая переменная (см. выражение 5):

$X : Complexity$;

$T(Complexity) = \{малая, средняя, большая\}$;

U – универсальное множество функциональных баллов, полученных по методу EFPA.

G – синтаксическое правило, порождающее термы множества $T(X)$, бесконтекстной грамматики;

M – функция совместимости Харрингтона: $d = e^{-e^{-R}}$.

Рассмотренные выше ранговые признаки [21] образуют пространство признаков, в котором координатные оси ФТ*: приоритет выполнения ФТ, сложность ФТ,* а их значения представляются лингвистическими переменными.

4. Алгоритм перехода из пространства «Ранг требований» к пространству «Тип ПС». Для анализа эмпирических многомерных данных существует ряд подходов: логический, геометрический и т.д. [22, 23]. Геометрический подход использует пространство признаков и дает более наглядную трактовку рассматриваемым критериям[22], образующим пространство «Тип ПС» (см. выражение (1)), которое включает в себя обобщенную характеристику требований – *ранг требований* (Requirement Rank), и её начальную объектно–ориентированную *структурную сложность* (Structural Complexity). В свою очередь *ранг требований* – сложный критерий, который тоже представляет пространство. Очевидно, что в такой интерпретации данной проблемы возникает задача корректного перехода из одного пространства в другое. Для ее решения предлагается алгоритм перехода, который проиллюстрированный с численным примером и графической интерпретацией представленной на рис. 1, он использует особенности ранжирования разнородных признаков которые содержатся в источниках [24, 25]:

Шаг 1. Определить зоны сложности (ЗС) в пространстве «Ранг требований».

Имеем n уровней сложности ФТ и m уровней приоритета выполнения ФТ. Количество ЗС – $\#ZoneComplexity$ вычисляется по формуле:

$$\#ZoneComplexity = m \cdot n . \qquad (6)$$

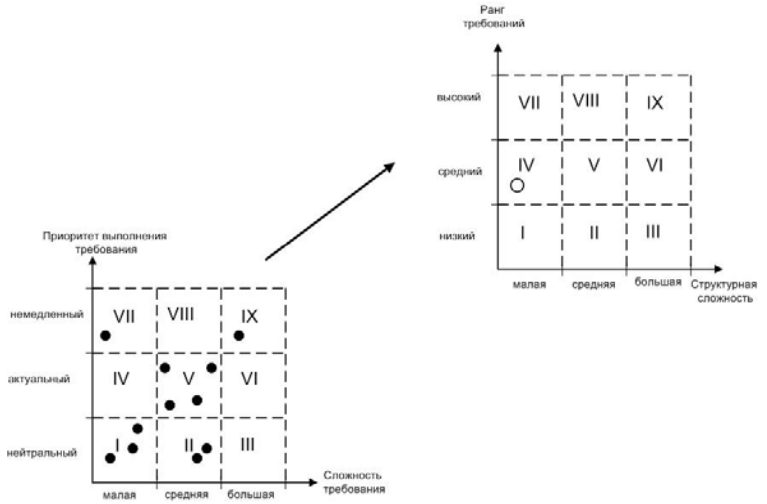

Рис. 1 – Отображение пространства «Ранг требований» в пространство «Тип ПС»

Шаг 2. Экспертным путем определить вес требования в каждой зоне.

Вес для каждого уровня сложности ФТ $w_n = n$, для каждого приоритета выполнения ФТ $w_m = m$. Вес W_{nm} для каждой ЗС вычисляется по формуле:

$$W_{nm} = m + n. \quad (7)$$

Шаг 3. Определить количество обобщенных ЗС.

ЗС относятся к одной обобщенной ЗС, если их вес W_{nm} одинаковый. Максимальное количество обобщенных ЗС – R вычисляется по формуле:

$$R = n + m - 1. \quad (8)$$

Шаг 4. Посчитать общее количество ФТ $\#Requirement^{total}$.

Шаг 5. Вычислить общий вес требований $GeneralRequirementW$

$$GeneralRequirementW = \sum_{i=1}^{R} W_i \cdot (\#RequirementInZone_i). \quad (9)$$

где $W_i; i = [\overline{1;R}]$ – вес $i-ой$ зоны, вычисленный на шаге 2,

$\#RequirementInZone_i, i = [\overline{1;R}]$ – количество ФТ, принадлежащих одной обобщенной ЗС.

Шаг 6. Вычислить средний вес ФТ $AvgRequirementWeight$ для всего пространства «Ранг требований»:

$$AvgcRequirementWeight = \frac{GeneralRequirementW}{\#Requirement^{total}}. \quad (10)$$

Шаг 7. Определить шкалу ранжирования требований, по которой находится числовое значение ранга требований.

Для данного алгоритма выбрана интервальная шкала т.к. граничные значения для расчитываются по принципу концентрации всех объектов в двух соседних зонах при этом соотношение между границами интервалов остается неизменным, каждое меняется в одно и то же число раз [17]. Для вычисления граничных значений шкалы используется выражение:

$$W_i = \frac{W_q + W_p}{2}, i = [\overline{1;R}], \quad (11)$$

где W_q и W_p – веса соседних зон.

Таким образом формируются интервалы, начиная с первой зоны, При этом нижняя граница каждого интервала закрыта, а верхняя – открыта.

Шаг 8. Осуществление перехода от пространства «Ранг требований» в пространство «Тип ПС». Для этого необходимо построить лингвистическую переменную «Ранг требований» (см. выражение 5), графическая интерпретация которой приведена на рис. 2:

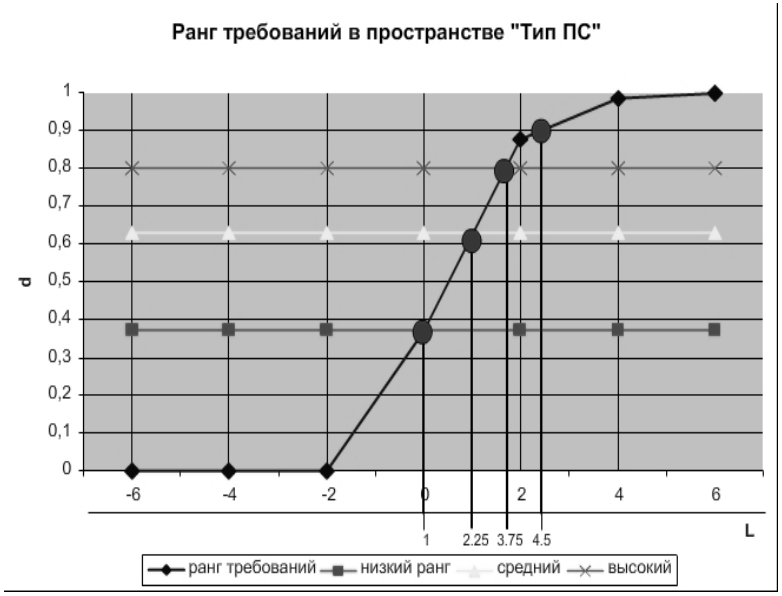

Рис. 2 – Значение функции принадлежности Харрингтона для ранга требований

X : *RequirementRank* ;

$T(RequirementRank) = \{низкий, средний, высокий\}$;

U – универсальное множество числовых значений, полученных в результате работы алгоритма перехода»:

$$шкала\_ранжирования \begin{cases} [1;2.25) \\ [2.25;3.75); \\ [3.75;4.5] \end{cases}$$

G – синтаксическое правило, порождающее термы множества $T(X)$, бесконтекстной грамматики;

M – функция совместимости Харрингтона: $d = e^{-e^{-R}}$.

Реализация данного подхода в настоящее время разрабатывается в виде перспективного CASE-средства, пример графического интерфейса которого приведен на рис. 3.

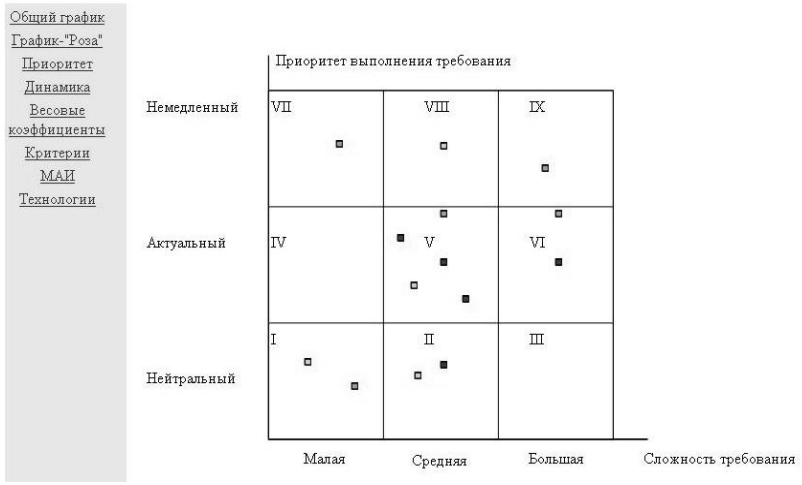

Рис. 3 – Пример графического интерфейса перспективного CASE-средства для пространства «Ранг требований»

5. Выводы и направления дальнейших исследований. В данной научной статье: 1) показана актуальность решения проблемы сопровождения УПС, разработанных на основе ООП, 2) разработана методика оценки сложности требований при сопровождении УПС на основе многомерного подхода для оценки их типов УПС. 3) предложен алгоритм перехода от пространства определения соответствующих типов ПС, 4) представлен пример графического интерфейса перспективного CASE-средства для программной реализации разработанной методики.

Полученные результаты будут в дальнейшем использованы для разработки комплексной оценки эффективности применения некоторых пост объектно-ориентированных технологий в процессах сопровождения унаследованный программных систем.

Список литературы: **1.** Software Life Cycle Processes : ISO/IEC 12207:2007. // http://www.ieee.org/publications_standards, 12.2010. **2.** Затраты жизненного цикла ПС // http://www.12207.com, 11.2010. **3**. *И. Соммервил.* Инженерия программного обеспечения. [6–е изд. / пер. с англ.] – М. : Вильямс. – 2002. – 624 с. **4**. *Ткачук Н. В., Нагорный К. А*. Структурная адаптация программных систем: анализ состояния проблемы и некоторые подходы к ее решению // Східно-Європейський журнал передових технологій. – 2009. – № 6 (42). – С. 33–36. **5**. *Ткачук М. В*. Моделі, методи та інформаційні технології адаптивної розробки і реінжинірингу інформаційно-управляючих систем : автореф. дис. на здоб. вчен. ступеня д-ра техн. наук : спец. 05.13.06 «Інформаційні технології» / Ткачук Микола В'ячеславович ; НТУ «ХПІ». – Харків, 2006. – 36 с. **6**. *Ткачук Н. В., Нагорный К.* А. Об одном подходе к оценке эффективности применения пост объектно-ориентированных технологий при сопровождении программных систем // Проблемы программирования (Problems in Programmimg. Scientific Journal). – К. : НАН України. – 2010. – № 2–3 (спец. выпуск). – С. 252 – 260. – ISSN 1727.–4907. **7**. K. K. Aggarwal. Empirical Study of Object-Oriented Metrics // JOURNAL OF OBJECT TECHNOLOGY. – 2006. – Vol. 5. – No. 8. – November–December. – P. 149–173. **8**. K. K. Aggarwal. Software Design Metrics for Object-Oriented Software // JOURNAL OF OBJECT TECHNOLOGY. – 2006. – Vol. 6. – No. 1. – January–February. – P. 121–138. **9.** *К. Вигерс.* Разработка требований к программному обеспечению / [пер. с англ.] – М. : Русская Редакция. – 2004. – 576 с. **10**. IBM, Get it right the first time: writing better requirements. – 2009. – 60 p. **11**. *Заде Л.* Нечеткая логика : Понятие лингвистической переменной и его применение к принятию приближенных решений. – М. : Москва. – 1976. – 167 с. **12.** *Ю. П. Адлер, Е. В. Маркова, Ю. В. Грановский.* Планирование эксперимента при поиске оптимальных условий. – М. : Наука. – 1976. – 280 с. **13.** Recommended Practice for Software Requirements Specification : ISO/IEC 830–1998 // http://www.ieee.org/publications_standards, 11.2010. **14**. *Д. Леффингуэлл, Д. Уидриг*. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. – М. : Вильямс. – 2002. – 448 с. **15**. *M. M. Mora, C. Denger*. Requirements Metrics // IESE-Report No. 096.03. – 2003. – Version 1.0. – October 1. – 45 p. **16**. NASA, Software Assurance Technology Center (SATC) // http://satc.gsfc.nasa.gov/tools/arm/, 11.2010. **17.** Information technology – software measurement – functional size measurement. Part 1 : definition of concepts : ISO/IEC 14143.1:1998 // http://www.ieee.org/publications_standards, 11.2010. **18.** *B. Henderson-Sellers, D. Zowghi [et al]*. Sizing Use Cases : How to create a standard metrical approach. Object Oriented Information Systems. – 2002. – 409–421 p. **19**. *Q*/P Management Group // http://www.qpmg.com/info.htm, 10.2010. **20**. *Luca Santillo, Roberto Meli*. Early Function Points : some practical experiences of use // ESCOM – ENCRESS 98, Project Control for 2000 and Beyond. – Rome, Italy. – 1998. – May 27–29. – 13 p. **21**. *Б. Г. Миркин*. Анализ качественных признаков и структур. – М. : Статистика. – 1980. – 319 с. **22.** *В. Дюк.* Обработка данных на ПК в примерах. – СПб. : Питер. – 1997. – 240 с. **23.** *М. Л. Дейвисо.* Многомерное шкалирование. – Миннеаполис. – 1992. – 261 с **24**. *Я. Гаек, З. Шидак*. Теория ранговых критериев. – М. : Москва. – 1971 – 376 с. **25**. *Д. М. Чибисов*. Лекции по асимптотической теории ранговых критериев. – М. : Москва. – 2009. – 174 с.

Надійшла до редколегії 10.11.2010