УДК 004.94:004.415.53

МОДЕЛИРОВАНИЕ ТЕХНОЛОГИЧЕСКОГО ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПРЕДСТАВЛЕНИЯ И ОБРАБОТКИ ЗНАНИЙ В ОБЛАСТИ ДИАГНОСТИКИ ОШИБОК

Черняховская Лилия Рашитовна
Д.т.н., профессор кафедры технической кибернетики, e-mail: lr_chern@yandex.ru
Никулина Наталья Олеговна
К.т.н., доцент кафедры автоматизированных систем управления,
e-mail: nick_nataly@rambler.ru
Малахова Анна Ивановна
К.т.н., доцент кафедры автоматизированных систем управления,
e-mail: aimalakhova@gmail.com
Гайткулов Ринат Тагирович
Магистрант 2-го курса кафедры автоматизированных систем управления,
e-mail: tag270791@gmail.com
ФГБОУ ВО «Уфимский государственный авиационный технический университет»,
450008, г. Уфа, ул. К. Маркса, 12

Аннотация. В статье обосновывается необходимость онтологического и динамического моделирования технологического процесса тестирования программного обеспечения в проектах по созданию информационных систем. Цель моделирования – разработка базы знаний для поддержки принятия решений в возникающих в ходе разработки программного обеспечения проблемных ситуациях, требующих коллективного обсуждения в условиях ограниченных ресурсов, в том числе, временных. Исследования поддержаны грантом РФФИ № 16-08-00575 «Интеллектуальные методы многокритериальной диагностики состояний сложных технических систем и технологических процессов».

Ключевые слова: тестирование программного обеспечения, онтологическая модель, динамическая модель, база знаний.

Цитирование: Черняховская Л.Р., Никулина Н.О., Малахова А.И., Гайткулов Р.Т. Моделирование технологического процесса тестирования программного обеспечения для представления и обработки знаний в области диагностики ошибок // Информационные и математические технологии в науке и управлении. 2018. №2 (10). С. 52–60. DOI:10.25729/2413-0133-2018-2-05

Введение. Повсеместное использование вычислительной техники в производственных процессах привело к тому, что программное обеспечение (ПО) стало усложняться. В то время как аппаратные средства обеспечивают достаточно высокий уровень надежности, программное обеспечение становится основным источником ошибок и неправильного функционирования систем [10, 15]. Для того, чтобы найти и исправить ошибки в функционировании программного обеспечения, необходимо качественно и всесторонне его протестировать. Несмотря на большое многообразие и постоянное обновление инструментальных средств тестирования, вопрос повышения качества

тестирования не теряет своей актуальности. Проблема обостряется еще и высокой текучестью кадров в организациях, занимающихся разработкой программного обеспечения.

Одним из выходов из сложившейся ситуации является использование систем поддержки принятия решения в процессе тестирования программного обеспечения.

Целью проводимых исследований является создание системы поддержки принятия решений для участников технологических процессов в области управления проектами разработки и внедрения крупномасштабных информационных систем на примере тестирования программного обеспечения.

Практическая ценность исследования состоит в использовании компонентов системы поддержки принятия решений для обнаружения и распознавания ошибок в тестируемом программном обеспечении, а также для обучения начинающих тестировщиков и повышения квалификации персонала ИТ-компаний. Применение онтологической базы знаний позволит принимать решения в технологическом процессе тестирования программного обеспечения на основе достоверной и однозначной информации, что приведет к снижению ошибок при управлении этим процессом.

1. Постановка задачи. Одним из способов повышения эффективности процесса тестирования является учет опыта ведения предыдущих проектов. Процесс тестирования не поддаётся формализации в полном объеме, так как большая часть информации принимает форму фактов и правил, которые не всегда четко определены. Почти никогда нельзя сказать, сколько ошибок присутствует в программе и сколько из них получится зафиксировать и исправить. Однако есть возможность описания наиболее часто встречающихся ошибок, методов их нахождения и устранения [2, 3, 7].

Виды тестирования и техника его проведения полностью зависят от конкретной ИТ-компании. Это связано с тем, что программные средства для поддержки тестирования и виды тестируемого программного обеспечения отличаются большим многообразием. Текучесть кадров и необходимость обучения новых сотрудников также привносят свою долю неопределенности в решение проблемы.

Необходимо внедрение базы знаний, которая будет разработана в результате онтологического анализа прецедентов принятия решения в случае обнаружения ошибок в программном обеспечении во время его тестирования. Такая база знаний будет содержать в себе правила логической поддержки и прецеденты принятия решений по диагностике тестируемого программного обеспечения.

Для создания базы знаний также необходима разработка динамической модели процесса тестирования программного обеспечения, с помощью которой становится возможным проектирование механизма взаимодействия участников процесса. Одним из результатов динамического моделирования является матрица ответственности, отражающая распределение прав и обязанностей лиц, принимающих решения.

Для достижения поставленной цели необходимо решить следующие задачи:

1) разработать динамические и онтологические модели технологического процесса тестирования ПО для представления знаний в области обнаружения и распознавания ошибок;

2) разработать правила принятия решений по диагностике программного обеспечения и правила взаимодействия участников процесса в соответствии с результатами диагностирования.

2. Разработка матрицы ответственности на основе динамической модели процесса тестирования программного обеспечения. Разработка и проектирование программного обеспечения включает в себя несколько ярко выраженных этапов, для которых четко определяют состав и последовательность работ, применяемые методы и средства, список участников процесса, а также результаты и критерии их оценки.

Наиболее развитой и современной моделью гибкой разработки программного обеспечения семейства Agile является методология Scrum. Сутью Scrum является предоставление заказчику в короткие промежутки времени (спринты) версии программного обеспечения с новой приоритетной функциональностью [4]. Благодаря тому, что каждый спринт занимает небольшой период времени и в его начале заданы определенные цели и задачи, процесс разработки становится предсказуемым и гибким.

Процесс разработки представляет собой последовательный контролируемый переход от одного спринта к другому. Каждый спринт занимает не более четырех недель. Так как в конце каждого спринта командой разработки должна быть получена рабочая версия программы, необходимо быстрое и качественное тестирование на протяжении всего спринта. Разработка тест-плана для каждого отдельного спринта приводит к тому, что на каждом этапе тестирования определяются как ожидаемые результаты тестирования, так и критерии входа и выхода из этапа [6]. Параллельно идет формирование стратегии тестирования следующих спринтов. Однако создание тест-планов не всегда удаётся из-за отсутствия хорошо формализованных требований как заказчика, так и команды аналитиков.

Благодаря возможности удаленной работы взаимодействие между участниками процесса можно представить как взаимодействие сотрудника с системой распределения задач, такой, как Jira [9]. На рисунке 1 представлена концептуальная модель взаимодействия участников процесса.

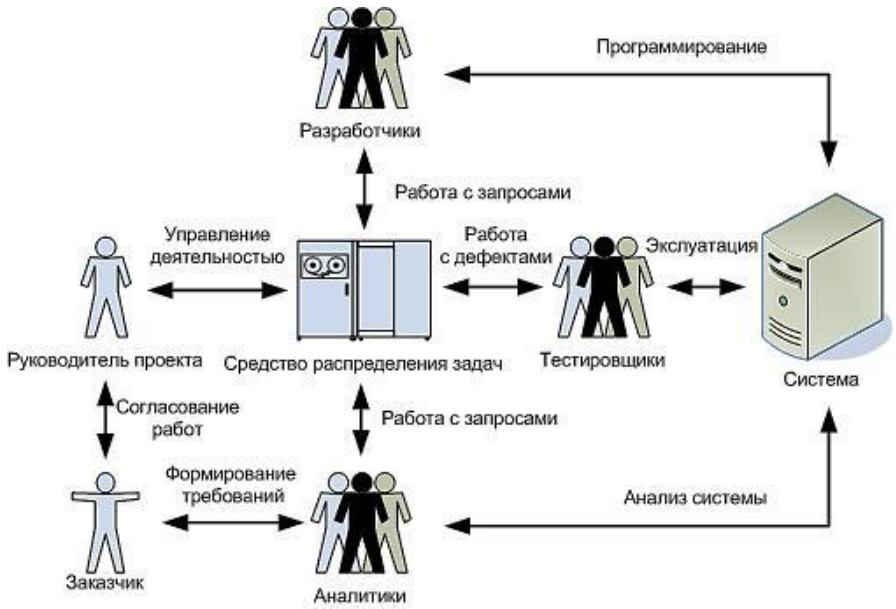

Рис. 1. Концептуальная схема разработки ПО

Контроль над работой сотрудников осуществляется с помощью систем управления проектами, такими как, Microsoft Project Server. Взаимодействие между участниками процесса тестирования программного обеспечения может осуществляться по различным

сценариям в зависимости от вида выявленной ошибки, типа проблемной ситуации, возможных последствий от принятых решений на ход выполнения проекта.

Возможны следующие ситуации (рис. 2):

1) обнаружение и исправление ошибок выполняется тестировщиком самостоятельно без привлечения других участников процесса;

2) в случае выявления ошибки, обусловленной неясно сформулированными требованиями, тестировщик решает проблему совместно с аналитиками;

3) в случае выявления ошибки, исправление которой требует доработки ПО, тестировщик решает проблему совместно с разработчиками;

4) в случае выявления ошибки, обусловленной неясно сформулированными требованиями, исправление которой требует разработки дополнительной функциональности, тестировщик решает проблему совместно с разработчиками и аналитиками;

5) последний рассмотренный сценарий может потребовать привлечения заказчика, если устранение проблемы потребует дополнительных временных или финансовых затрат или повлияет на качество конечного продукта.

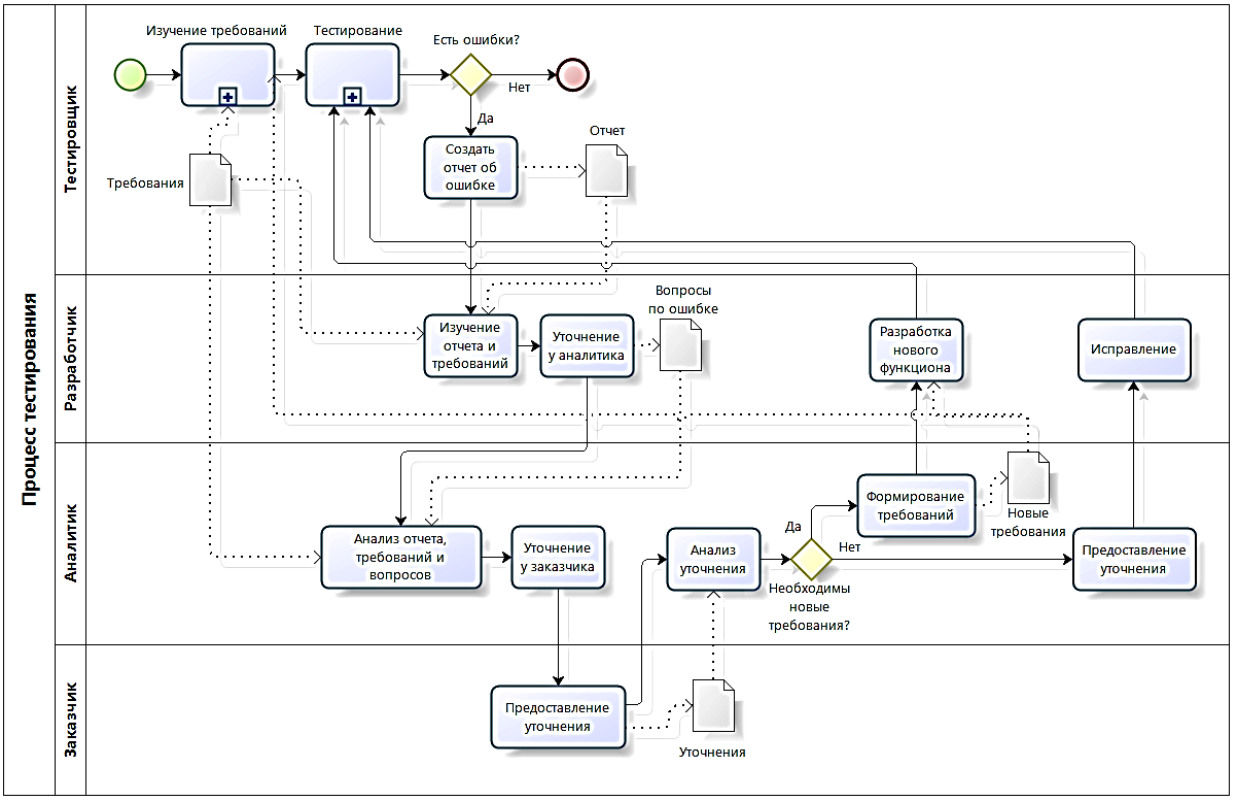

Рис. 2. Динамическая модель тестирования ПО

На основании полученной информации построена матрица ответственности (табл. 1), где О – ответственный, Н – наблюдатель, К – консультант, И – исполнитель.

Применение методологии Scrum для разработки ПО ограничивает состав участников команды, в результате границы ответственности ролей за выполняемые функции получаются размытыми, что, в свою очередь, ведет к возникновению проблемных ситуаций. Так, например, наиболее часто встречающейся ситуацией является совмещение ролей исполнителя и ответственного за результаты.

Таблица 1. Фрагмент матрицы ответственности

Операция	Участники			
	Разработчик	Тестировщик	Аналитик	Заказчик
Исправление кода	И, О	К	Н	
Исправление требований	Н	К	И, О	Н
Разработка требований	Н	Н	О, И	К
Разработка кода	И, О		Н	
Тестирование кода	О	И	К	
Тестирование требований	Н	И	О	К

Динамическое моделирование позволяет более точно определить границы и степень ответственности участников проекта за отдельные виды работ, а также выявить ситуации, где необходимо коллективное принятие решений.

3. Разработка правил взаимодействия участников процесса на основе результатов моделирования. Помимо динамического моделирования, для дальнейшей формализации процесса тестирования программного обеспечения необходима разработка онтологической модели. Методология Scrum ограничивает не только состав участников, но и время, необходимое для разработки заданной функциональности, что может привести к ухудшению качества программного обеспечения либо за счет сокращения первоначально согласованного списка требований, либо за счет поверхностного тестирования. Ускорить принятие решений в случае возникновения проблемной ситуации, а также определить персональную ответственность участников процесса разработки ПО поможет база знаний, разработанная с учетом накопленного в организации опыта ведения проектов [1, 11-13].

В ходе анализа предметной области были выделены 20 классов и 12 свойств объектов и данных. Фрагмент разработанной OWL-онтологии представлен на рисунке 3.

Для разработки онтологии были использованы наиболее часто используемые атрибуты идентификации и описания ошибок.

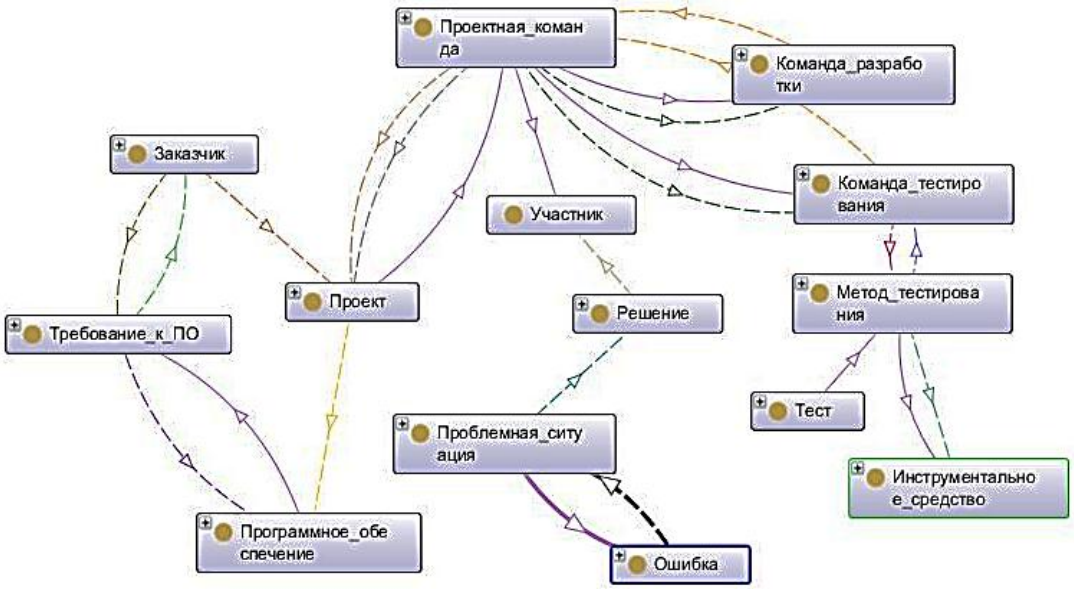

Рис. 3. Фрагмент онтологии процесса тестирования ПО

Для разработанной онтологии с учетом проведенного ранее динамического моделирования были сформированы правила, примеры которых представлены в таблице 2.

Таблица 2. Правила принятия решений по результатам тестирования

Описание правила	Правило на языке SWRL
Если ошибка имеет вид «Ошибка программы» и зафиксирована в модуле «Администрирование», то компетенция «Тестировщик»	Модуль(?z), Ошибка(?y), Проблемная_ситуация(?x), Участник(?p), имеетРешение(?x, ?d), проявляетсяВ(?y, ?z), вид(?y, «Ошибка программы»), компетенция(?p, «Тестировщик»), наименование(?z, «Администрирование») -> исполняет(?p, ?d)
Если ошибка имеет вид «Ошибка требований», компетенция «Заказчик» и статус «Реализован», то решение «Новые функции»	Заказчик(?c), Ошибка(?y), Проблемная_ситуация(?x), имеетРешение(?x, ?d), вид(?y, «Ошибка требований») -> исполняет(?c, ?d), значение(?d, «Новый функционал»)
Если ошибка имеет вид «Ошибка требований» и зафиксирована в «Техническом задании», то компетенция «Заказчик»	Заказчик(?c), Ошибка(?y), Проблемная_ситуация(?x), Техническое_задание(?w), имеетРешение(?x, ?d), компетенция(?c, true) -> исполняет(?c, ?d), значение(?d, «Исправление»)
Если ошибка имеет вид «Ошибка требований», компетенция «Аналитик», статус «Не реализован», то решение «Исправить требование»	Аналитик(?a), Ошибка(?y), Проблемная_ситуация(?x), имеетРешение(?x,?d), вид(?y, «Ошибка требований») -> исполняет(?a, ?d), значение(?d, «Исправить требование»)

Использование онтологии в процессе тестирования ПО позволит [5, 8, 14]:

1) описать тестируемую программную систему: её функции, требования к ним, необходимые тесты и связи между ними;

2) описать наиболее часто встречающиеся ошибки, привести их классификацию, а также методы их устранения;

3) определить наиболее квалифицированных тестировщиков для каждого класса тестируемых программных систем.

Применение онтологической базы знаний позволит принимать решения в технологическом процессе тестирования ПО на основе достоверной и однозначно понимаемой информации, что приведет к снижению ошибок при управлении этим процессом.

Заключение. В ходе исследования был выполнен анализ процесса тестирования программного обеспечения, разработана динамическая модель, сочетающая различные сценарии взаимодействия участников процесса, разработаны OWL-онтология и правила к ней, а также матрица ответственности.

Использование базы знаний на основе онтологии позволит:

– использовать общую терминологию всеми участниками процесса для обеспечения однозначности восприятия информации;

– использовать прецеденты, накопленные в системе Jira, для выработки правил принятия решений в типовых проблемных ситуациях;

– формировать аналитические отчеты о ходе выполнения проекта разработки программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. Бармина О.В., Никулина Н.О. Интеллектуальная система управления взаимодействием бизнес-процессов в проектно-ориентированных организациях // Онтология проектирования. 2017. Т.7. №1(23). С. 48–65. DOI: 10.18287/2223-9537-2017-7-1-48-65.
2. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. М.: Питер. 2004. 320 с.
3. Бек К. Экстремальное программирование: разработка через тестирование. М.: Питер. 2003. 224 с.
4. Джефф Сазерленд. Scrum. Революционный метод управления проектами. М.: Манн, Иванов и Фербер. 2016. 288 с.
5. Добров Б.В., Иванов В.В., Лукашевич Н.В., Соловьев В.Д. Онтологии и тезаурусы: модели, инструменты, приложения. М.: Бином. Лаборатория знаний. 2009. 173 с.
6. Кеннет Рубин. Основы Scrum: Практическое руководство по гибкой разработке ПО. М.: «Вильямс». 2016. 544 с.
7. Лаврищева Е.М., Петрухин В.А. Методы и средства инженерии программного обеспечения: учебное пособие. М.: Изд-во МФТИ. 2006. 304 с.
8. Никулина Н.О., Гайткулов Р.Т., Старцева Е.Б. Информационная поддержка процесса тестирования программного обеспечения на основе онтологической базы знаний // Новые технологии в научных исследованиях, проектировании, управлении, производстве: труды XV Междунар. науч.-техн. конф. (Воронеж, 9-10 ноября 2017 г.). Воронеж: ФГБОУ ВО «Воронежский государственный технический университет». 2017. Т. 1. С. 334–339.
9. Официальный сайт компании «Atlassian». Режим доступа: https://ru.atlassian.com/software/jira/ (дата обращения: 13.02.2018).
10. Поддержка принятия решений при стратегическом управлении предприятием на основе инженерии знаний / Под ред. Л.Р. Черняховской. Уфа: АН РБ. Гилем. 2010. 128 с.
11. Черняховская Л.Р., Малахова А.И. Разработка моделей и методов интеллектуальной поддержки принятия решений на основе онтологии организационного управления программными проектами // Онтология проектирования. 2013. №4 (10). С. 42–52.
12. Черняховская Л.Р. Старцева Е.Б., Владимирова И.П., Малахова А.И. Управление принятием решений в организационном управлении с применением правил // Вестник УГАТУ. 2012. Т. 16, № 3 (48). С. 53–55.
13. Черняховская Л.Р., Никулина Н.О., Малахова А.И., Федорова Н.И. Разработка системы диагностических знаний с использованием интеллектуального анализа данных // Проблемы управления и моделирования в сложных системах: труды XIX Междунар. конф., (Самара, 12-15 сентября 2017 г.). Самара: ООО "Офорт". 2017. С. 519–525.
14. Шпак М.А. Онтология как посредник между пользователем и информационной системой // Молодежный научный вестник МГТУ им. Н.Э. Баумана. 2014. №6. С. 13–18.

15. Gartner Says Worldwide IT Spending Forecast to Grow 2.7 Percent in 2017. Режим доступа: http://www.gartner.com/newsroom/id/3568917 (дата обращения: 13.03.2018).

UDK 004.94:004.415.53

MODELLING OF THE SOFTWARE TESTING TECHNOLOGICAL PROCESS FOR KNOWLEDGE REPRESENTATION AND PROCESSING IN ERROR DIAGNOSTICS AREA

Liliya R. Chernyakhovskaya
Doctor of technical sciences, professor of technical cybernetics department,
e-mail: lr_chern@yandex.ru
Nataliya O. Nikulina
Candidate of technical sciences, docent of automated management systems department,
e-mail: nick_nataly@rambler.ru
Anna I. Malakhova
Candidate of technical sciences, docent of automated management systems department,
e-mail: aimalakhova@gmail.com
Rinat T. Gaitkulov
2nd course master of automated management systems department, e-mail: tag270791@gmail.com
Ufa State Aviation Technical University, 12, K. Marks Str., 450008, Ufa, Russia

Abstract. The article proves a necessity of ontological and dynamic modeling of the software testing technological process in information systems creation projects. A purpose of modeling is to develop a knowledge base for decision-making support in problem situations, which arise in the course of software development and require collective discussion in conditions of limited resources, including time resources. The research is supported by the RFBR grant № 16-08-00575 "Intelligent methods for multi-criteria diagnosis of complex technical systems and technological processes".
Keywords: software testing, ontological model, dynamic model, knowledge base.

References

1. Barmina O.V., Nikulina N.O. Intellektual'naya sistema upravleniya vzaimodeystviyem biznes-protsessov v proyektno-oriyentirovannykh organizatsiyakh [Intelligent system for interactive business processes management in project-oriented organizations] // Ontologiya proyektirovaniya = Ontology of designing. 2017. 7(1). Pp. 48–65. DOI: 10.18287/2223-9537-2017-7-1-48-65. (in Russian)
2. Beizer B. Testirovaniye chernogo yashchika. Tekhnologii funktsional'nogo testirovaniya programmnogo obespecheniya i sistem [The black box testing. Technologies of the software and systems functional testing]. Moscow. Piter. 2004. 320 p. (in Russian)
3. Bek K. Ekstremal'noye programmirovaniye: razrabotka cherez testirovaniye [Extreme programming: test-driven development] Moscow. Piter. 2003. 224 p. (in Russian)
4. Jeff Sutherland. Scrum. Revolyutsionnyy metod upravleniya proyektami [Scrum. The art of doing twice the work in half the time]. Moscow. Mann, Ivanov i Ferber = Mann, Ivanov and Ferber. 2016. 288 p. (in Russian)

5. Dobrov B.V., Ivanov V.V., Lukashevich N.V., Soloviev V.D. Ontologii i tezaurusy: modeli, instrumenty, prilozheniya [Ontologies and thesauruses: models, tools, applications]. M.: Binom. Laboratoriia znanii = Binom. Knowledge laboratory. 2009. 173 p. (in Russian)
6. Kenneth S. Rubin. Essential Scrum: Osnovy Scrum: Prakticheskoye rukovodstvo po gibkoy razrabotke PO [A Practical Guide to the Most Popular Agile Process]. Moscow. «Williams». 2016. 544 p. (in Russian)
7. Lavrisheva E.M., Petrukhin V.A. Metody i sredstva inzhenerii programmnogo obespecheniya: uchebnoye posobiye [Methods and means of software engineering: tutorial]. Moscow. MFTI = MPTU. 2006. 304 p. (in Russian)
8. Nikulina N.O., Gaitkulov R.T., Startseva E.B. Informatsionnaya podderzhka protsessa testirovaniya programmnogo obespecheniya na osnove ontologicheskoy bazy znaniy [Information support of software testing process based on ontological knowledge base] // Novyye tekhnologii v nauchnykh issledovaniyakh, proyektirovanii, upravlenii, proizvodstve: trudy XV Mezhdunar. nauch.-tekhn. konf. (Voronezh, 9-10 noyabrya 2017 g.) = New technologies in scientific research, design, management, production: proceedings of the XV International scientific and technological conference (November, 9-10, 2017, Voronezh). Voronezh. Voronezh State Technical University (VSTU). 2017. V. 1. Pp. 334–339. (in Russian)
9. Official website of "Atlassian". Available at: https://ru.atlassian.com/software/jira/ (accessed 13.02.2018). (in Russian)
10. Podderzhka prinyatiya resheniy pri strategicheskom upravlenii predpriyatiyem na osnove inzhenerii znaniy [Decision making support in enterprise strategic management based on the knowledge engineering] / Under edition of L.R. Chernyakhovskaya. Ufa. Akademiya nauk Respubliki Bashkortostan, izdatel'stvo "Gilem" = Academy of Sciences of the Republic of Bashkortostan, publishing house "Gilem". 2010. 128 p. (in Russian)
11. Chernyakhovskaya L.R., Malakhova A.I. Razrabotka modeley i metodov intellektual'noy podderzhki prinyatiya resheniy na osnove ontologii organizatsionnogo upravleniya programmnymi proyektami [Development of intelligent decision support models and methods based on the software projects organization management ontology] // Ontologiya proyektirovaniya = Ontology of designing. 2013. № 4(10). Pp. 42–52. (in Russian)
12. Chernyakhovskaya L.R., Startseva E.B., Vladimirova I.P., Malakhova A.I. Upravleniye prinyatiyem resheniy v organizatsionnom upravlenii s primeneniyem pravil [Decision-making control in organization management using rules] // Vestnik USATU = Bulletin of the USATU. 2012. V. 16. № 3(48). Pp. 53–55. (in Russian)
13. Chernyakhovskaya L.R., Nikulina N.O., Malakhova A.I., Fedorova N.I. Razrabotka sistemy diagnosticheskikh znaniy s ispol'zovaniyem intellektual'nogo analiza dannykh [Development of diagnostic knowledge system using data mining] // Problemy upravleniya i modelirovaniya v slozhnykh sistemakh: trudy XIX Mezhdunar. konf., (Samara, 12-15 sentyabrya 2017 g.) = Management and modeling problems in complex systems: proceedings of the XIX International conference (September, 12-15, 2017, Samara). Samara. OOO “Ofort” =Ltd. "Ofort". 2017. Pp. 519–525. (in Russian)
14. Shpak M.A. Ontologiya kak posrednik mezhdu pol'zovatelem i informatsionnoy sistemoy [Ontology as a mediator between the user and the information system] // Molodezhnyi nauchnyi vestnik MGTU im. N.E. Baumana = Youth Scientific Bulletin of the Bauman. MSTU. 2014. №6. Pp. 13–18. (in Russian)
15. Gartner Says Worldwide IT Spending Forecast to Grow 2.7 Percent in 2017. Available at: http://www.gartner.com/newsroom/id/3568917 (accessed 13.03.2018)