РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

УДК 004.052

О.В. ПОМОРОВА, А.В. ИВАНОВ
Хмельницкий национальный университет
М.Ю. ЛЯШКЕВИЧ
Черновицкий национальный университет им. Ю. Федьковича

СИСТЕМА НЕЧЕТКОГО ВЫВОДА ДЛЯ АНАЛИЗА, МОНИТОРИНГА И ОЦЕНКИ РИСКОВ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье представлены результаты исследования процесса управления рисками в контексте неопределенности информации. На примере пяти базовых рисков процесса разработки программного обеспечения построено множество нечетких экспертных правил. Представлена система нечеткого вывода для анализа, мониторинга и оценки рисков при разработке программного обеспечения с использованием пакета Fuzzy Logic Toolbox системы Matlab. Приведены результаты функционирования системы.

Ключевые слова: программное обеспечение, анализрисков, неопределенность информации, система нечеткого вывода.

The paper presents the results of research of the risk management process in the context of uncertainty information. A lot of fuzzy expert rules built on the example of the five basic risks of the software development process. The fuzzy inference system for analysis, monitoring and evaluation of risks in software development using the Fuzzy Logic Toolbox of Matlabis represented. Given the results of the functioning of the fuzzy inference system.

Keywords: software, analizriskov, the uncertainty of information, system of fuzzy logic.

Введение

Разработка программного обеспечения (ПО) – это наукоемкая деятельность, которая требует досконального знания предметной области и понимания целей разрабатываемого продукта. Диагноз о наличии кризиса в программной инженерии был поставлен более полувека назад. С тех пор разработано много новых языков программирования, методов, методик, технологий и достигнут значительный прогресс в этой отрасли. Однако сложность и комплексность современных программных продуктов существенно возросли, поэтому качество ПО по-прежнему зависит от знаний разработчиков и опыта их участия в разного рода проектах.

По приблизительным оценкам, затраты на разработку ПО сегодня составляют около 275 миллиардов долларов, 72 % проектов достигают этапа внедрения и только 26 % всех проектов завершаются успешно. Для эффективной деятельности в области индустрии ПО софтверные компании должны разрабатывать, внедрять и сопровождать свои продукты быстро, укладываясь в сроки, с удовлетворительным качеством. Поэтому компании вкладывают значительные средства в модернизацию технологий разработки и средств обеспечения качества ПО[1].

Тем не менее, инциденты, связанные со сбоями в программном обеспечении, не перестают появляться. В декабре 2011 года специалисты NASA успешно исправили ошибку в программном обеспечении бортового вычислительного комплекса движущегося к Марсу космического аппарата с марсоходом "Кьюриосити" на борту. Зонд стартовал с космодрома на мысе Канаверал 26 ноября 2011 года. Через три дня после старта, 29 ноября, при включении датчика звездной ориентации, компьютер аппарата самопроизвольно перезагрузился.Специалисты выяснили, что причиной сбоя была ранее неизвестная конструктивная особенность модуля управления памятью в процессоре компьютера зонда. При некоторых редких стечениях обстоятельств при попытке доступа к быстрой памяти возникали ошибки, которые приводили к некорректному выполнению команд и перезагрузке. Чтобы исключить новые перезагрузки, специалисты изменили систему распределения данных в памяти.Проверки на наземных стендах показали, что сбоев в новой конфигурации не происходит. Новое программное обеспечение было загружено в бортовой компьютер марсохода. После этого аппарат получил возможность использовать датчики солнечной и звездной ориентации в обычном режиме [2].

Удачный исход устранения ошибок на этапе эксплуатации ПО является весьма редким явлением. Обычно такие инциденты приводят к значительным человеческим и экономическим потерям, поэтому в софтверных организациях внедряется ряд мер по обеспечению качества. Одной из таких мер являются мероприятия по выявлению, анализу, мониторингу и устранению рисков при разработке программного обеспечения.

Риск – это вероятность проявления каких-либо неблагоприятных обстоятельств, негативно влияющих на реализацию проекта по разработке программного обеспечения. Например, если при написании программного кода используется новый язык программирования, то риск может заключаться в том, что компилятор этого языка ненадежен или результирующий код не достаточно эффективен.Рисками являются также превышение сроков или стоимости проекта. Уменьшение (разрешение) рисков – важный элемент управления программными проектами.

Ряд моделей процесса создания программного обеспечения уделяют значительное внимание

анализу рисков и требуют точного определения и оценивания рисков [3]. Например, в спиральной модели (рис. 1) каждый виток спирали разбит на четыре сектора, из которых три непосредственно связаны с анализом рисков.

1. *Определение целей* каждой итерации проекта. В этом секторе определяются проектные риски (например, риск превышения сроков или риск превышения стоимости проекта). В зависимости от «проявленных» рисков, могут планироваться альтернативные стратегии разработки ПО.

2. *Оценка и разрешение рисков.* Для каждого определенного проектного риска проводится его детальный анализ. Планируются мероприятия для уменьшения (разрешения) рисков. Например, если существует риск, что системные требования определены неверно, планируется разработать прототип системы.

3. *Разработка и тестирование.* После оценки рисков выбирается модель процесса создания системы. Если доминируют риски, связанные с разработкой интерфейсов, наиболее подходящей будет эволюционная модель разработки ПО с прототипированием. Если основные риски связаны с соответствием системы и спецификации, то, скорее всего, следует применить модель формальных преобразований. Каскадная модель может быть применена в том случае, если основные риски определены как ошибки, которые могут проявиться на этапе сборки системы.

4. *Планирование.* Проект пересматривается и принимается решение о моменте начала следующего витка спирали.

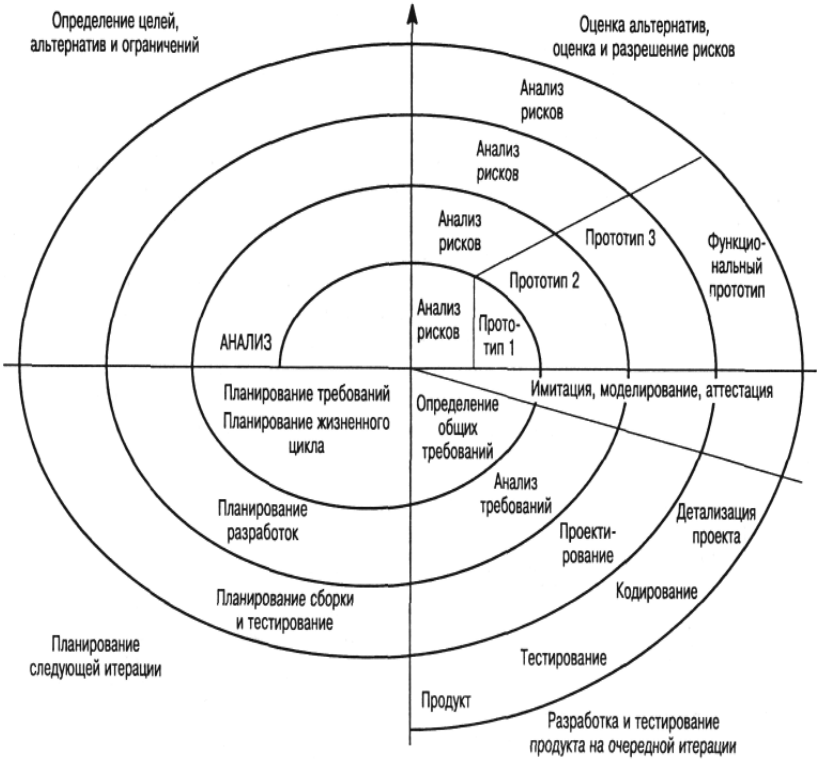

Рис. 1. Спиральная модель создания ПО [3]

Постановка задачи

Анализ, оценка и мониторинг рисков, которые могут повлиять на график работ или на качество создаваемого программного продуктаявляется *актуальной задачей* и важной частью жизненного цикла ПО. Основными факторами, препятствующими успешному решению этой задачи, являются: неточная и неполная информация о ПО, процессе его разработки и предметной области, субъективизм и недостаток опыта специалиста по управлению рисками. Для повышения эффективности принятия решений относительно рисков при разработке ПО и их влияния на качество программных продуктов необходимо разработать систему, которая позволит проводить анализ, мониторинг и оценку рисков в условиях неполноты и неточности входящей информации с учетом опыта разработки предыдущих программных проектов.

Процесс управления рисками в контексте неопределенности информации

Для оценки рисков используется детальный анализ альтернатив достижения целей разработки ПО, прототипирование, имитационное моделирование и т.п. С учетом полученных оценок рисков выбирается тот или иной подход к разработке системных компонентов, далее он реализуется, затем осуществляется планирование следующего этапа процесса создания ПО. Определение рисков и разработка мероприятий по уменьшению их влияния на ход выполнения проекта называется *управлением рисками*.

Риски могут угрожать проекту в целом, создаваемому программному продукту или организации-разработчику. Выделяют три типа рисков: *риски для проекта,* которые влияют на график работ или ресурсы,

необходимые для выполнения проекта; *риски для разрабатываемого продукта,* влияющие на качество или производительность разрабатываемого программного продукта; *бизнес-риски,* которыеотносятся к организации-разработчику или поставщикам.

Эти типы рисков могут пересекаться, что усложняет их анализ и оценку. Например, если опытный программист покидает проект, это будет риском для проекта (поскольку задерживается срок сдачи готового продукта), риском для продукта (новый программист может оказаться не слишком опытным и сделать ошибки в программе) и бизнес-риском (поскольку задержка данного проекта может негативно повлиять на будущие деловые контакты между заказчиком и организацией-разработчиком).

Существует ряд признаков, которые помогают определить тип риска[3]. Технологические риски характеризуются задержками в поставке оборудования или программных средств поддержки процесса создания ПО, многочисленными документированными технологическими проблемами. Риски, связанные с персоналом учитывают моральное состояние персонала, отношения между членами команды разработчиков, качество выполненной работы. Организационные риски оцениваются по мнению персонала о компетентности высшего руководства организации. Инструментальные риски проявляются в нежелании разработчиков использовать программные средства поддержки, неодобрительных отзывах о CASE-средствах, запросах разработчиков на более мощные инструментальные средства. Риски, связанные с системными требованиями, возрастают при необходимости пересмотра многих системных требований, недовольстве заказчика ПО. Риски оценивания проявляются в частых изменениях графика работ, многочисленных отчетах о нарушении графика работ.

Конкретные типы рисков, которые могут оказать влияние на данный проект, зависят от вида создаваемого программного продукта и от организационного окружения, в котором реализуется программный проект.

Схемапроцесса управления рисками показана на рис. 2. Этот процесс состоит из четырех этапов.

1. *Определение рисков.* Определяются возможные риски для проекта, для разрабатываемого продукта и бизнес-риски. Определение рисков обычно выполняется в режиме командной работы с использованием подхода «мозговой штурм». Эксперты указывают возможные риски, используя понятия естественного языка. При этом появляется информация, которая является неточной, неполной или недостоверной.

2. *Анализ рисков.* Оценивается вероятность и последовательность появления рисковых ситуаций. Большинство оценок является неточными и субъективными. Употребляются лингвистические понятия и оценки типа: «в большинстве случаев», «часто», «иногда», «высокий», «быстро», «малый», «сильный», «катастрофический». При малых объемах статистических данных о рисках в данной организации для схожего ПО оценка вероятности становится практически невозможной. Речь может идти лишь о «возможности» возникновения тех или иных рисков.

3. *Планирование рисков.* Планируются мероприятия по предотвращению рисков или минимизации их воздействия на проект. При этом ни руководитель проекта, ни эксперты зачастую не могут дать четкого обоснования значимости мероприятий для предотвращения рисков.

4. *Мониторинг рисков.* Постоянное оценивание вероятностей рисков и выполнение мероприятий по смягчению последствий проявления рисковых ситуаций. На этом этапе количество неточной, неполной и субъективной информации только возрастает.

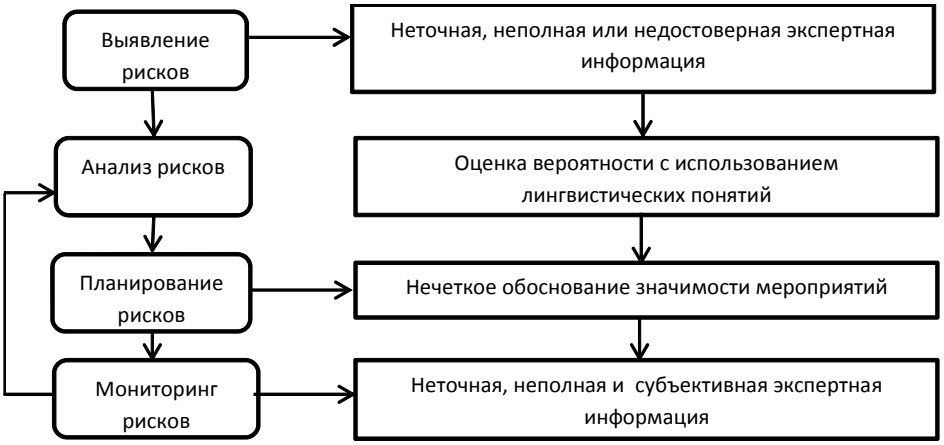

Рис. 2. Процесс управления рисками в контексте неопределенности информации

При анализе для каждого определенного риска подсчитывается вероятность его проявления и ущерб, который он может нанести. Не существует простых методов выполнения анализа рисков – в значительной мере он основан на мнении и опыте руководителя проекта. Для большинства проектов используется следующая шкала вероятностей рисков и их последствий: вероятность риска считается очень низкой, если она имеет значение менее 10 %; низкой от 10 до 25 %; средней при значениях от 25 до 50 %; высокой, если значение колеблется от 50 до 75 %; очень высокой при значениях более 75 %. В данном

случае вероятность в действительности является возможностью проявления риска.

Возможный ущерб от рисковых ситуаций можно разделить на катастрофический, серьезный, терпимый и незначительный. Грань между последствиями рисков четко не определена и для разных проектов масштабы, например, катастрофических последствий, могут значительно отличаться.

После проведения анализа рисков определяются наиболее значимые риски, которые затем отслеживаются на протяжении всего срока выполнения проекта. Мониторинг рисков является непрерывным процессом, отслеживающим ход выполнения мероприятий по управлению рисками, при этом каждый основной риск должен рассматриваться отдельно.

Формальные статистические модели для анализа и мониторинга рисков не являются адекватными. Это связано с недостаточным объемом статистических выборок, изменяющимися с течением времени бизнес условиями и характеристиками проекта и программного продукта.

Ввиду наличия значительных объемов неточной, неполной и недостоверной информацией в процессе управления рисками имеет смысл проводить анализ рисков с использованием механизма нечеткого вывода.

Нечеткая модель для анализа рисков

Системы нечеткого вывода предназначены для реализации процесса нечеткого вывода и служат концептуальным базисом всей современной нечеткой логики. Такие системы позволяют решать задачи автоматического управления, классификации данных, распознавания образов, принятия решений, машинного обучения и др. [4].

Система нечеткого вывода для анализа рисков является частным случаем продукционных нечетких систем, в которых условия и заключения отдельных правил формулируются в форме нечетких высказываний относительно значений тех или иных лингвистических переменных. Для получения заключений в системах нечеткого вывода предложены несколько алгоритмов, описание которых базируется на разделении процесса вывода на ряд последовательных этапов:

- формирование базы правил системы нечеткого вывода;
- фаззификация входных переменных;
- агрегирование подусловий в нечетких правилах;
- активизация или композиция подзаключений в нечетких правилах;
- аккумулирование заключений нечетких правил продукций;
- дефаззификация выходных лингвистических переменных.

База правил нечетких продукций представляет собой конечное множество продукционных правил нечетких продукций, согласованных относительно используемых в них лингвистических переменных.

Под *фаззификацией* понимается не только отдельный этап выполнения нечеткого вывода, но и собственно процесс или процедура нахождения значений функций принадлежности нечетких множеств (термов) на основе обычных (не нечетких) исходных данных. Целью этапа фаззификации является установление соответствия между численным значением отдельной входной переменной системы нечеткого вывода и значением функции принадлежности соответствующего ей терма входной лингвистической переменной.

Агрегирование представляет собой процедуру определения степени истинности условий по каждому из правил системы нечеткого вывода.

Активизация- это процесс нахождения степени истинности каждого из подзаключений нечетких правил продукций.

Аккумуляция или *аккумулирование* представляет собой процедуру нахождения функции принадлежности для каждой из выходных лингвистических переменных. Цель аккумуляции заключается в том, чтобы объединить или аккумулировать все степени истинности заключений (подзаключений) для получения функции принадлежности каждой из выходных переменных. Причина необходимости выполнения этого этапа состоит в том, что подзаключения, относящиеся к одной и той же выходной лингвистической переменной, принадлежат различным правилам системы нечеткого вывода.

Дефаззификация- это процесс нахождения обычного (не нечеткого) значения для каждой из выходных лингвистических переменных. Цель дефаззификации заключается в том, чтобы, используя результаты аккумуляции всех выходных лингвистических переменных, получить обычное количественное значение каждой из выходных переменных, которое может быть использовано специальными устройствами, внешними по отношению к системе нечеткого вывода. Этап дефаззификации считается законченным, когда для каждой из выходных лингвистических переменных будут определены итоговые количественные значения в форме некоторого действительного числа.

Для построения системы нечеткого вывода для анализа, мониторинга и оценки рисков при разработке программного обеспеченияобыла разработана*нечеткая модель* задачи анализа рисков.

В нечеткой модели использованы входные переменные, которые представляют набор наиболее вероятных для программного проекта рисков и степени влияния рисков на качество ПО:

- текучесть кадров;
- изменение пользовательских требований;
- уровень квалификации разработчиков (владение CASE-средствами разработки и языком программирования);

- вероятность появления конкурирующего ПО;
- степень влияния других (кроме этих четырех) рисков на качество ПО.

В качестве выходной переменной использовано значение степени ущерба от наступления рисковых ситуаций, которое является непрямой оценкой качества ПО. Чем выше возможность проявления каждого из рисков, тем более значимым будет ущерб, а чем выше возможный ущерб, тем хуже качество продукта.

Все рассматриваемые переменные измеряются в баллах в интервале действительных чисел от 0 до 10. Для переменной «уровень квалификации разработчиков» наихудшая оценка переменной равна 0, а наилучшее значение равно 10. Для всех остальных входных переменных и для выходной переменной наихудшая оценка равна 10, а наилучшее значение равно 0.

Для анализа степени возможного ущерба экспертами было составленомножество нечетких эвристических правил (более 40). Примеры правил:

1) если текучесть кадров средняя и уровень квалификации разработчиков очень высокий и изменения пользовательских требований редкие, то степень ущерба незначительная;

2) если вероятность появления конкурирующего ПО низкая и уровень квалификации разработчиков очень высокий и изменения пользовательских требований частые, то степень ущерба терпимая;

3) если текучесть кадров низкая и степень влияния других рисков на качество ПО средняя и уровень квалификации разработчиков очень высокий, то степень ущерба незначительная;

4) если вероятность появления конкурирующего ПО низкая и изменения пользовательских требований средние и уровень квалификации разработчиков низкий, то степень ущерба терпимая;

5) если вероятность появления конкурирующего ПО высокая и изменения пользовательских требований частые и текучесть кадров средняя, то степень ущерба катастрофическая.

После формирования базы знаний был проведен этап фаззификации входных и выходных переменных нечеткой модели.

Для представления терм-множества лингвистической переменной «текучесть кадров» используется множество SET1={«низкая», «средняя», «высокая»}. В символическом виде зададим его как SET1={L, M, H}.

В качестве терм-множества переменной «изменения пользовательских требований» используется аналогичное множество SET2={«редкие», «средние», «частые»}. В символическом виде зададим его как SET2={L, M, H}.

В качестве терм-множества переменной «уровень квалификации разработчиков» используется множество SET3={«очень высокий», «высокий», «средний», «низкий»}. В символическом виде зададим его как SET3={VH, H, M, L}.Функция принадлежности термов изображена на рис. 3.

В качестве терм-множества лингвистической переменной «вероятность появления конкурирующего ПО» используется множество SET4={«низкая», «средняя», «высокая»}. В символическом виде зададим его как SET4={L, M, H}.Функция принадлежности термов изображена на рис. 4.

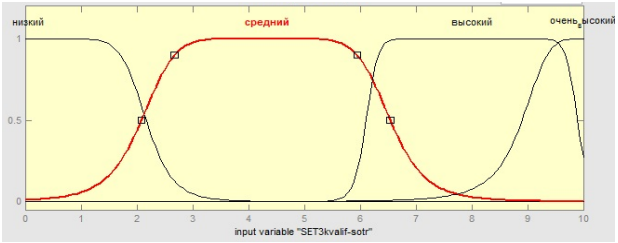

Рис. 3. График функции принадлежности термов переменной «уровень квалификации разработчиков»

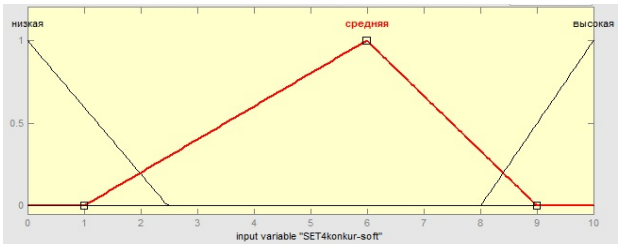

Рис. 4. График функции принадлежности термов переменной «появление конкурирующего ПО»

В качестве терм-множества лингвистической переменной «степень влияния других рисков на качество ПО» используется множество SET5={«низкая», «средняя», «высокая»}. В символическом виде зададим его как SET5={L, M, H}.Функция принадлежности термов изображена на рис. 5.

Для представления терм-множества выходной лингвистической переменной «степень ущерба» используется множество SET6={«незначительная», «серьёзная», «терпимая», «катастрофическая»}. В символическом виде зададим его как SET6={LL, L, M, H}.Функция принадлежности термов изображена на рис. 6.

В качестве схемы нечеткого вывода используется метод Мамдани[4, 5].

Метод активации-**min**, который рассчитывается по формуле:

$$\mu'(y) = min\{c_i, \mu(y)\},$$

где c_i – степени истинности подзаключений для каждого из правил, $\mu(y)$ – функция принадлежности терма выходной переменной.

Методы агрегирования подусловий. Поскольку во всех правилах в качестве логической связки для подусловий применяется только нечеткая конъюнкция (операция "И"), то в качестве метода агрегирования используется операция **min** конъюнкции.

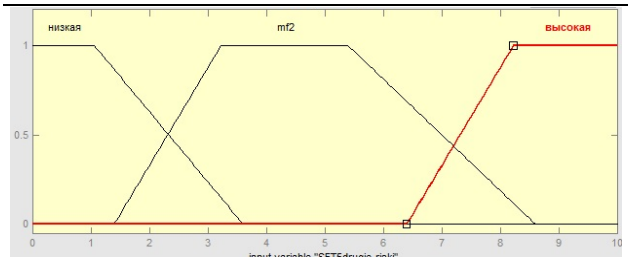

Рис. 5. График функции принадлежности термов переменной «степень влияния других рисков на качество ПО»

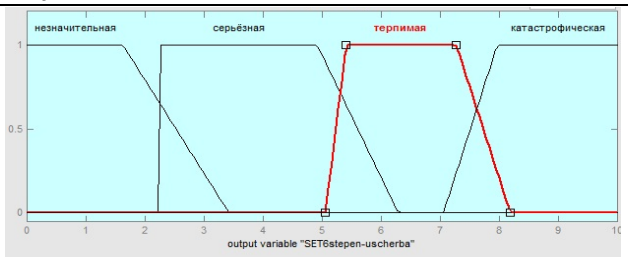

Рис. 6. График функции принадлежности термов выходной переменной «степень ущерба»

Для аккумуляции заключений правил используется метод **max** дизъюнкции, который также применяется в случае схемы нечеткого вывода методом Мамдани. В качестве метода дефаззификации используется метод центра тяжести. Согласно этому методу центр тяжести вычисляется по формуле:

$$y = \int_{min}^{max} x \cdot \mu(x)\,dx \Big/ \int_{min}^{max} \mu(x)\,dx .$$

где y – результат дефаззификации; x - переменная, соответствующая выходной лингвистической переменной, $\mu(x)$ - функция принадлежности нечеткого множества, соответствующего выходной лингвистической переменной после этапа аккумуляции; min и max - левая и правая точки интервала носителя нечеткого множества рассматриваемой выходной лингвистической переменной.

При дефаззификации методом центра тяжести обычное (не нечеткое) значение выходной переменной равно абсциссе центра тяжести площади, ограниченной графиком кривой функции принадлежности соответствующей выходной переменной.

Система нечеткого вывода

Для построения системы нечеткого вывода использован пакет Fuzzy Logic Toolbox системы Matlab. С помощью редактора нечеткого вывода FIS Editor задана следующие параметры системы нечеткого вывода: тип – Мамдани, метод агрегирования, метод аккумуляции, метод деффазификации, все входы и выходы системы нечеткого вывода и их названия. В нашем случае с помощью меню заданы пять входов и один выход (рис. 7).

Функции принадлежности были отредактированы и переименованы соответственно значениям, которые они могут принимать. Процесс редактирования представлен на рисунке 8. Для переменной «текучесть кадров» установлена трапециевидная функция принадлежности и указан вид функции для каждого члена терм-множества. Аналогичным образом определены другие переменные.

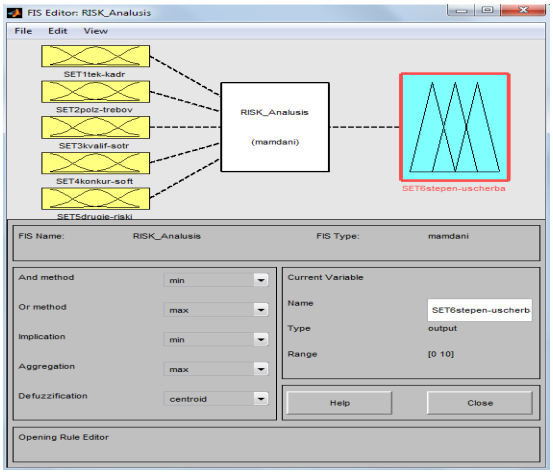

Рис. 7. Окно с указанием входов и выходов системы нечеткого вывода

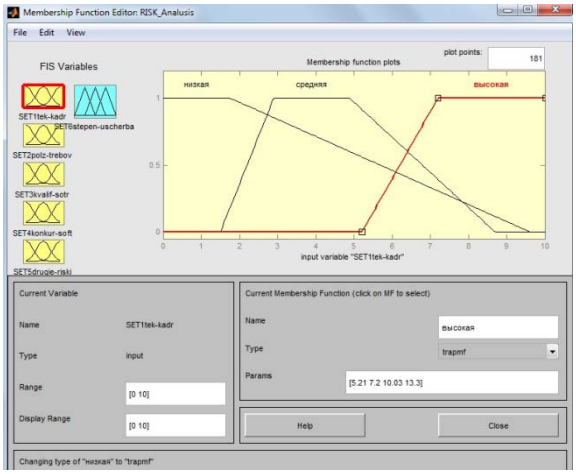

Рис. 8. Редактирование функций принадлежности для переменной «текучесть кадров»

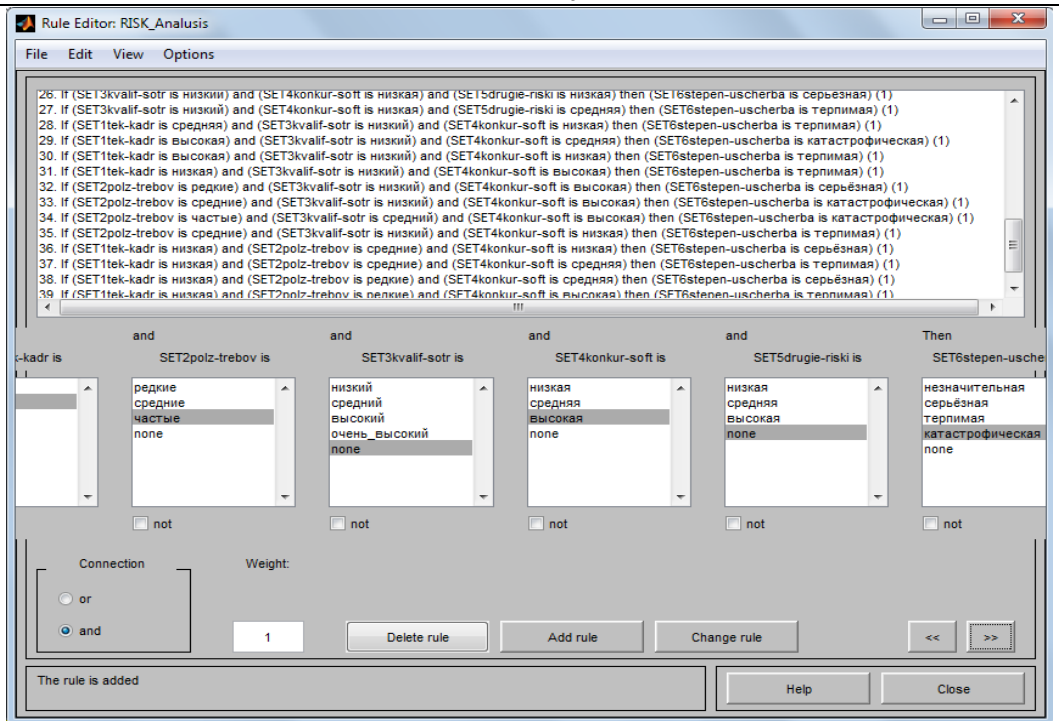

Рис. 9. Набор правил для нечеткого логического вывода

Оценка степени ущерба от реализации рисков с помощью системы нечеткого вывода

Для оценки степени ущерба от реализации рисков с помощью системы нечеткого логического вывода в окне просмотра базы правил **Rule Viewer**указываются значения входных переменных (рис. 10, внизу). Результат оценки степени ущерба выводится в правом верхнем углу окна базы правил (рис. 10, сверху). Также в этом окне отображаются результаты срабатывания каждого правила (выделены желтым цветом) и их влияние на результирующую оценку (выделены синим цветом).

Система нечеткого вывода обеспечивает возможность визуализации поверхности зависимости степени ущерба от двух входных рисков при фиксированных значениях остальных рисков.

На рис. 11 представлена зависимость степени ущерба от уровня квалификации сотрудников и от текучести кадров при фиксированных значениях изменения пользовательских требований (значение риска равно 4), вероятности появления конкурирующего ПО (значение риска равно 7) и степени влияния других рисков проекта (значение риска равно 2).

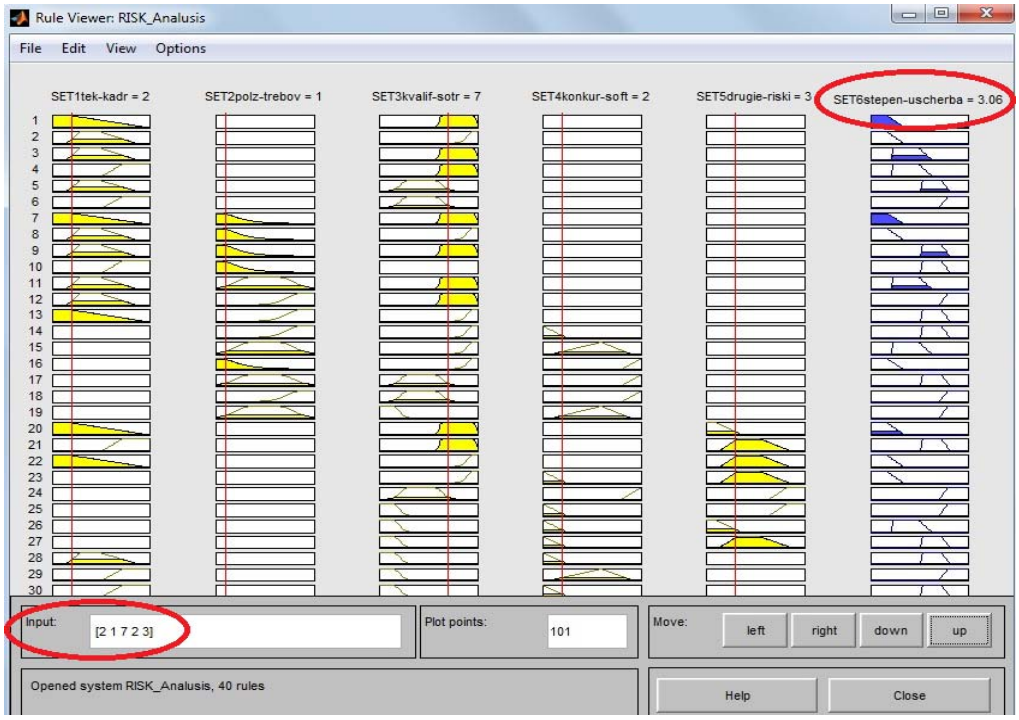

Рис. 10. Результаты нечеткого логического вывода при заданных значениях входных переменных

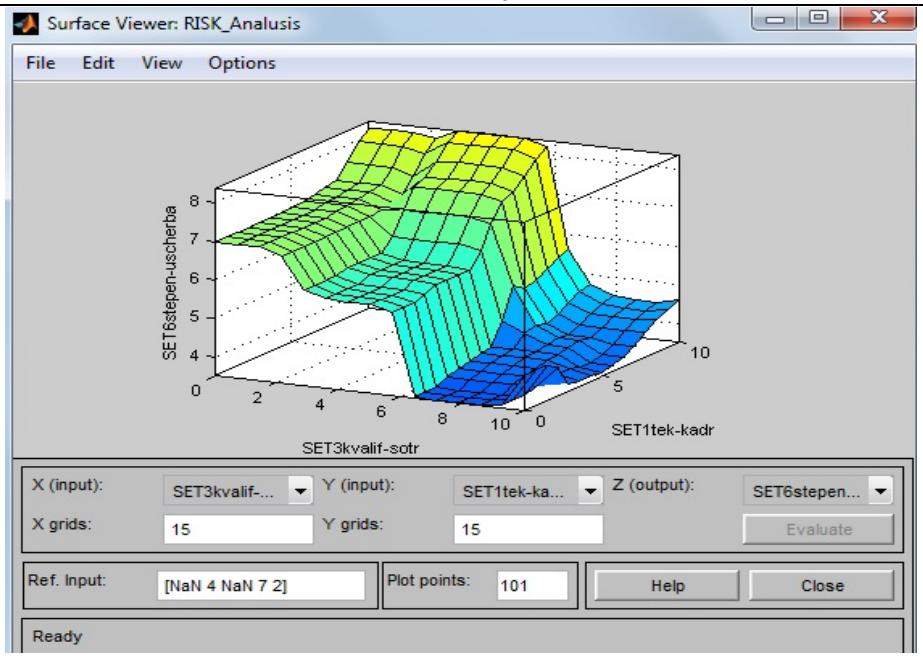

Рис. 11. Зависимость степени ущерба от уровня квалификации сотрудников и от текучести кадров

Такая визуализация обеспечивает руководителя проекта информацией для принятия решений относительно оптимальных допустимых значений рисков в текущем времени. Например, анализ поверхностистепени ущерба, представленной на рис. 11, показывает, что степень ущерба равна 6при значениях риска «уровень квалификации разработчиков» в диапазоне [2,5... 6,5] и риска «текучесть кадров» в диапазоне [0... 5] при фиксированных значениях других рисков. Следовательно, если степень ущерба достигла значения 6, то нет необходимости принимать меры и тратить ресурсы на понижение значений рисков «уровень квалификации разработчиков» и «текучесть кадров» пока они не достигли 6,5 и 5 соответственно.

В процессе реализации проекта значения рисков изменяются. Система нечеткого логического вывода обеспечивает руководителя проекта информацией для анализа, мониторинга и прогнозирования возможного влияния рисков на ход проекта в будущем. Руководитель проекта может оценить насколько позитивно или негативно повлияют изменения каждого отдельного рискана возможный ущерб от реализации всех возможныхрисков.

Выводы

Исследование процесса управления рисками показало наличие неточной, неполной и субъективной информации на каждом из его этапов. В этом случае для описания влияния рисков на процесс и качество разработки программного обеспечения эффективным является использование нечеткой логики.Нечеткая модель задачи анализа рисков позволила учесть опыт разработки ПО при построении нечетких экспертных правил. Система нечеткого вывода для анализа, мониторинга и оценки рисков обеспечивает возможность оценки степени возможного ущерба от реализации рисков и исследования зависимости ущерба от изменения каждого отдельного риска.Визуализация поверхностей степени ущерба показала наличие областей значений рисков, при которых не происходит изменения значений ущерба. Такая информация способствует принятию эффективных решений относительно стратегий управления рисками.

Литература

1. Поморова О.В. CASE-оценка критических программных систем. В 3-х томах. Том 1. Качество / Мищенко В.О., Поморова О.В., Говорущенко Т.А. / Под ред. Харченко В.С. – Х.: Нац. аэрокосмический ун-т «Харьк. авиац. ин-т», 2012. – 201 с.

2. Инженеры NASA устранили неполадку компьютера марсохода Curiousity. [Электронный ресурс] Режим доступа: http://www.gazeta.ru/news/science/2012/02/10/n_2199477.shtml.

3. Иоан Соммервилл. Инженерия программного обеспечения. 6-е изд. / ИанСоммервилл– М.: Изд. дом Вильямс, 2002. – 624 с.

4. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. / Леоненков А. В. – СПб.: БХВПетербург, 2005. – 736 с.

5. Новак В. Математические принципы нечёткой логики / Новак В., Перфильева И., Мочкрож И. – Физматлит, 2006. – 352 с.

Надійшла 19.1.2013 р.
Статтю представляє: д.т.н. Поморова О.В.