Федеральное государственное автономное образовательное
учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

На правах рукописи

Тележкин Александр Михайлович

Применение алгоритмических сетей для оценки ресурсов в программных проектах

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата технических наук

Научные руководители:

кандидат технических наук, старший научный сотрудник

Морозов В.П.

доктор физико-математических наук, профессор

Баранов С.Н.

Санкт-Петербург – 2015

Оглавление

Введение

Актуальность темы. Оценка ресурсов необходимых для выполнения проектов разработки ПО является важным и нетривиальным процессом, требующим глубоких теоретических и практических (в том числе и плохо формализуемых) знаний. По данным экспертов [**1**, **2**], 70% программных проектов завершаются с существенно иными, чем запланированными, параметрами качества, бюджета и графика, либо не завершаются вообще. Одной из причин данной ситуации является некачественная оценка необходимых ресурсов при запуске и дальнейшем перепланировании проекта.

В свою очередь, некачественная оценка возникает вследствие неадекватной методики ее выполнения при недостаточности отводимых на нее сроков и неполноты исходной информации. При этом компании, разрабатывающие ПО, как правило, накапливают реальную информацию об уже выполненных ими проектах, но эти достаточно ценные данные не всегда анализируются и используются для оценки ресурсов в новом проекте. На этапе запуска проекта руководители выполняют оценку и планирование без учета этого опыта, часто повторяя ошибки совершенные ранее.

На данный момент существует большое количество моделей и методов для оценки ресурсов, необходимых для выполнения проектов разработки ПИ. Условно их можно разделить на две группы – алгоритмические и неалгоритмические. Алгоритмические модели (COCOMO [3], SLIM [4], SEER-SEM [5], анализ функциональных точек [7] и другие) характеризуются наличием некого формального аппарата для подсчета необходимых ресурсов исходя из некоторых начальных данных о будущем ПИ и команде его разработчиков. Основным преимуществом методов этого типа является обоснованность и повторяемость оценки, получаемой в результате вычислений. Неалгоритмические модели (оценка по Паркинсону [8], экспертная оценка [9], оценка по аналогу [10] и другие) характеризуются своими методиками, схемами и принципами

выполнения оценки, основанными на привлечении опыта и знаний эксперта, плохо поддающихся формализации. Но, несмотря на разнообразие методов и моделей оценки, до сих пор не существует единого общепринятого множества характеристик, используемого для оценки необходимых ресурсов.

В сложившейся ситуации является актуальным создание модели оценки ресурсов для выполнения проектов разработки ПИ, которая бы работала в условиях неполноты и неточности исходных данных, а также совмещала бы преимущества алгоритмических и неалгоритмических методов. В качестве формального аппарата для модели предлагается использовать алгоритмические сети [11-16], т.к. их формализм, с одной стороны, можно рассматривать как полноценный язык высокого уровня, а с другой стороны, этот формализм ориентирован на конечного пользователя, не обладающего специальными знаниями в области программирования. Подходы на основе алгоритмических сетей хорошо зарекомендовали себя в таких областях, как теория моделирования [17, 18], моделирование экологических систем [19], экономика [20], параллельные вычисления [21]; таким образом, их применение в области оценки ресурсов при планировании программного проекта – представляется актуальной и многообещающей задачей.

Предметом исследования являются технологические процессы планирования и отслеживания проектов, выполняемых разработчиками программных продуктов, их характеристики и взаимосвязи между ними.

Целью работы является повышения точности и оперативности оценки ресурсов, необходимых программному проекту.

Решаемая научная задача: разработка на базе формализма алгоритмических сетей автоматизированного метода для обоснования оценок необходимых программному проекту ресурсов, который бы позволил увеличить точность и оперативность такой оценки. Метод должен быть ориентирован на эксперта,

имеющего целостное представление о предметной области, но при этом не обладающего специальными знаниями в программировании.

В интересах достижения цели работы и решения научной задачи осуществлялись:

- сравнительный анализ существующих подходов (методы, модели, программные средства) оценки ресурсов, необходимых для разработки ПО в рамках программного проекта;
- разработка метода и технологии оценки необходимых ресурсов (метод гибких оценок) на базе формализма алгоритмических сетей;
- проектирование, разработка и апробация системы для создания исторических баз данных по выполненным разработкам, которая реализует метод гибких оценок.

Методы исследования. При выполнении диссертационного исследования использовались методы математического моделирования на основе алгоритмических сетей, теории распознавания образов, методы и модели теории принятия решений.

Основные положения, выносимые на защиту. На основе проведенных теоретических работ и их экспериментальной апробации на защиту выносятся следующие положения:

1) модель формирования базы выполненных проектов для поиска проектов-аналогов.
2) метод формирования пространства характеристик для оценки ресурсов, необходимых для выполнения проектов разработки программных изделий на основе алгоритмических сетей (метод гибких оценок).
3) модель программной системы для автоматизированного поиска ближайших проектов-аналогов по базе выполненных проектов для оценки необходимых ресурсов по принципу подобия.

Научная новизна работы.

Новизна первого результата состоит в разработке модели формирования базы выполненных проектов с учетом неопределенного характера информация о значениях характеристик, описывающих проекты, неполноты исходных данных, слабо формализуемого опыта и высокой квалификация экспертов, проводящих оценку.

Особенность второго научного результата заключается в специфике использования формализма алгоритмических сетей для формирования пространства характеристик, используемого для оценки ресурсов, необходимых для выполнения проектов разработки программных изделий (метод гибких оценок). Данный формализм используется для привлечения опыта экспертов, плохо поддающегося формализации, но при этом не требующего от самих экспертов специальных знаний в области программирования.

Отличительным признаком третьего научного результата выступают процедура и система для автоматизированного поиска проекта-аналога ранее выполнявшихся в компании для использования фактических данных этих проектов в обосновании оценок ресурсов нового проекта по принципу подобия. Особенностью процедуры поиска проекта-аналога является использование предварительно подготовленной (отфильтрованной) БД на основе количественных и качественных характеристик метрик. Помимо этого, для использования в системе было собрано и классифицировано множество, содержащее более 100 метрик, характеризующих программный проект, создаваемый продукт и процесс разработки, учет которых позволяет производить обоснованную оценку ресурсов, необходимых программному проекту.

Теоретическую ценность диссертационной работы составляют разработанная модель формирования базы выполненных проектов и метод гибких оценок, использующий класс алгоритмов распознавания, основанных на вычислении оценок (АВО), предложенный Ю.И. Журавлевым [74].

Практическая ценность диссертационной работы состоит в расширении области применения формализма алгоритмических сетей и в повышении точности и объективности оценки ресурсов, необходимых для выполнения проектов разработки программных изделий, при сокращении времени на получение такой оценки.

Реализация и внедрение результатов исследования

- Результаты, полученные в ходе исследования, нашли свое отражение в виде системы САМПО+ для поддержки создания базы выполненных проектов, которая является надстройкой над Microsoft Excel и реализована на языке Visual Basic for Applications (VBA).
- С 2009 по 2010 гг. система САМПО+ использовалась в процессе оценки ресурсов для проектов разработки ПО компании ООО «Эксиджен Сервисес» (в конце 2009 года компания претерпела изменения и стала именоваться «Ф-Лайн Софтвер») [22] (копия акта в приложении 1). С 2011 по 2014 гг. метод формирования пространства характеристик и множество метрических характеристик программных проектов использовались в НП «Объединение подземных строителей» для разработки и внедрения системы автоматизации документооборота «Эскулап» (копия акта в приложении 2).

Публикация и апробация работы. По теме диссертации опубликовано 7 работ, в том числе 2 из них в изданиях, рекомендованных ВАК. Основные положения, выносимые на защиту, результаты исследования и выводы, содержащиеся в диссертационной работе, обсуждались в на научных семинарах в Санкт-петербургском институте информатики и автоматизации Российской академии наук, а также на конференциях: XI Санкт-Петербургская Международная конференция «Региональная информатика (РИ-2008)» (Санкт-Петербург, 2008), Четвертая всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика » (ИММОД-2009) (Санкт-Петербург, 2009), XII Санкт-Петербургская Международная конференция

«Региональная информатика (РИ-2010)» (Санкт-Петербург, 2010), Юбилейная XIII Санкт-Петербургская Международная конференция «Региональная информатика (РИ-2012)» (Санкт-Петербург, 2012).

Структура и объем диссертации. Диссертационная работа включает введение, четыре главы, заключение, список используемых источников (78 наименования) и три приложения. Текст диссертации изложен на 129 страницах, включая 20 рисунков и 24 таблицы.

Глава 1 Анализ подходов к оцениванию ресурсов для программных проектов

1.1 Методы оценки ресурсов

Создание высококачественного программного обеспечения (ПО) в современную эпоху информационных технологий давно стало одним из важных и существенных высокотехнологичных производств в развитых и развивающихся странах. Высокая сложность ПО как технического объекта, обусловленная как огромными размерами самого ПО, так и огромным количеством состояний, в которых это ПО может находиться [23]. Некоторые программы (такие как операционные системы) содержат миллионы строк кода и должны выполняться так, как задумано разработчиками. На данный момент сложность ПО уже превысила сложность наиболее технологичных машин нашего времени, таких как самолеты, которые, по сути, представляют собой программно-аппаратные комплексы. Поэтому задача правильного оценивания ресурсов для проекта разработки ПИ не только остается актуальной, но и становится все более важной в процессе разработки ПО.

Существует большое количество методов для оценки ресурсов как специфичных для области разработки ПО, так и подходящих для всех отраслей производства. Условно их можно разделить на алгоритмические и неалгоритмические методы оценки ресурсов разработки ПО.

1.1.1 Алгоритмические методы

В основе любого алгоритмического метода оценки ПО лежит одна или несколько функций, описывающая зависимость характеристик проекта (оценочные значения размера продукта, различные отраслевые коэффициенты) от затрат на его выполнение.

В качестве примеров алгоритмических моделей оценки ресурсов ПО рассмотрим две наиболее документированные и успешно применяемые модели: COCOMO / COCOMOII [3] и SLIM (модель Путнэма) [4].

Для того чтобы воспользоваться алгоритмическими методами оценки ресурсов, необходимо ввести понятие размера ПО. В качестве единиц размера ПО обычно рассматриваются либо LOC (lines of code, строки кода) [24], либо FP (functional points, функциональные точки) [7].

Lines of code (LOC)

Lines of code (LOC) – строки кода, метрика ПО, характеризующие его объем. Определяется путем подсчета строк исходного кода ПО, причем специфические операторы, комментарии и пустые строки не учитываются.

Метрика LOC в основном используется для оценки трудозатрат на разработку ПО, исходя из производительности команды разработчиков (S), которая рассчитывается по следующей формуле $S = n/m$, где n – количество строк кода (LOC) в программном продукте, а m – суммарное время (в человеко-часах), затраченное на его разработку.

В связи с тем, что размеры некоторых программных продуктов индустрии превышают миллионы строк кода, помимо LOC используются следующие связанные величины: KLOC – тысячи строк кода, MLOC – миллионы строк кода, GLOK – миллиарды строк кода.

На рисунке 1.1 приведены оценочные размеры некоторых программных продуктов индустрии разработки ПО размером от 25 до 85 MLOC [6].

Если в ходе разработки ПИ используется несколько разных языков программирования, то общий объем может быть оценен в KAELOC – K of Assembler Equivalent Lines Of Code (тысячи строк кода в ассемблерном эквиваленте). Для перевода KLOC в KAELOC используются коэффициенты, установленные опытным путем для различных языков программирования.

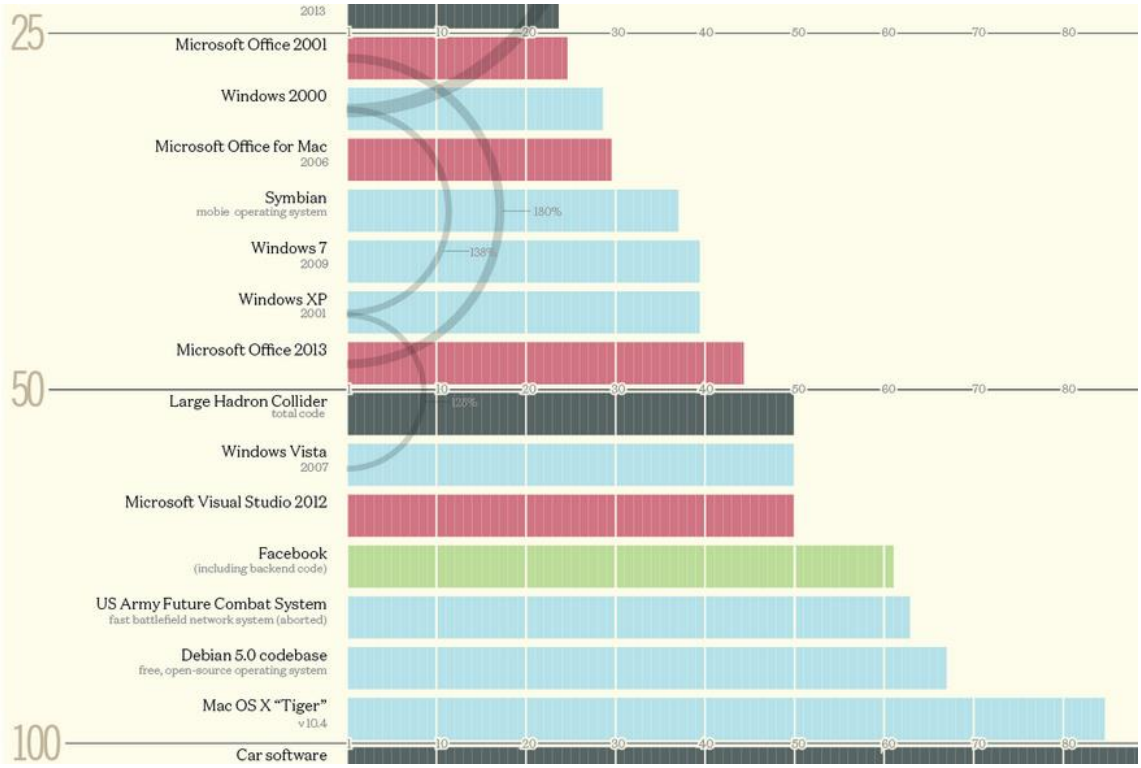


Рисунок 1.1. Размеры некоторых программных продуктов в единицах MLOC. Данные из [6]

В таблице 1.1 представлены коэффициенты пересчета KLOC в KAELOC для распространённых языков программирования. Данные в этой таблице получены эмпирическим путем, в результате сравнения кода, создаваемого типичными компиляторами с этих языков.

Таблица.1.1. Коэффициенты пересчета в KAELOC

Язык программирования	**Коэффициент пересчета**	**Язык программирования**	**Коэффициент пересчета**
Assembler	1,0	FORTRAN	3,0
Macro-Assembler	1,0	Pascal	3,5
LISP	1,5	C++	11,0
Unix shell scripts	1,5	Query languages	25,0
C	2,5		

Традиционно к преимуществам использования метрики LOC относят:

- простоту сбора метрики (так как легко можно автоматизировать подсчет строк кода продукта);
- простота анализа метрики;
- повсеместное использование в проектах.

Основные минусы использования LOC:

- сложность в определении точного размера продукта на ранних стадиях проекта;
- величина LOC зависит от языка программирования и стандартов кодирования. Очевидно, что 1000 LOC языка LISPне эквивалентны 1000 LOC языка C++. Оценка в LOC мало эффективна при использовании автоматизированных средств программирования (например, разработка пользовательских интерфейсов), т.к. в этом случае значительный объем кода может создаваться все лишь за несколько кликов мыши;
- сложность в оценке продуктивности, основанная на неверном представлении о том, что чем большое строк кода написано, тем выше продуктивность программиста.

Functional Points (FP)

Functional point (FP) – функциональная точка – единица измерения функциональности программного обеспечения, известная с 1979 г. [7] как альтернатива LOC. Потребность в ней возникла при оценке затрат труда на разработку ПО, которая бы не зависела от языка программирования и среды разработки.

Со второй половины 80-х годов развитие методики и разработку стандарта FP ведет Международная группа пользователей функциональных точек IFPUG (International Function Point User Group). Эта группа подготовила руководство по методике расчёта функциональных точек Function Point Counting Practices Manual (FPCPM), впоследствии ставшее стандартом ISO [25]. На данный момент существует 5 признанных стандартов ISO для расчета функциональных точек и 1 спецификация [26] для автоматизированного расчета функциональных точек, принятая международной некоммерческой организацией OMG (Object Management Group) [27].

Суть метода расчета функциональных точек состоит в том, что объем проекта разработки ПО оценивается на основе логической модели количества функционала требуемого заказчиком, и предоставляемого разработчиком (см. рис. 1.2).

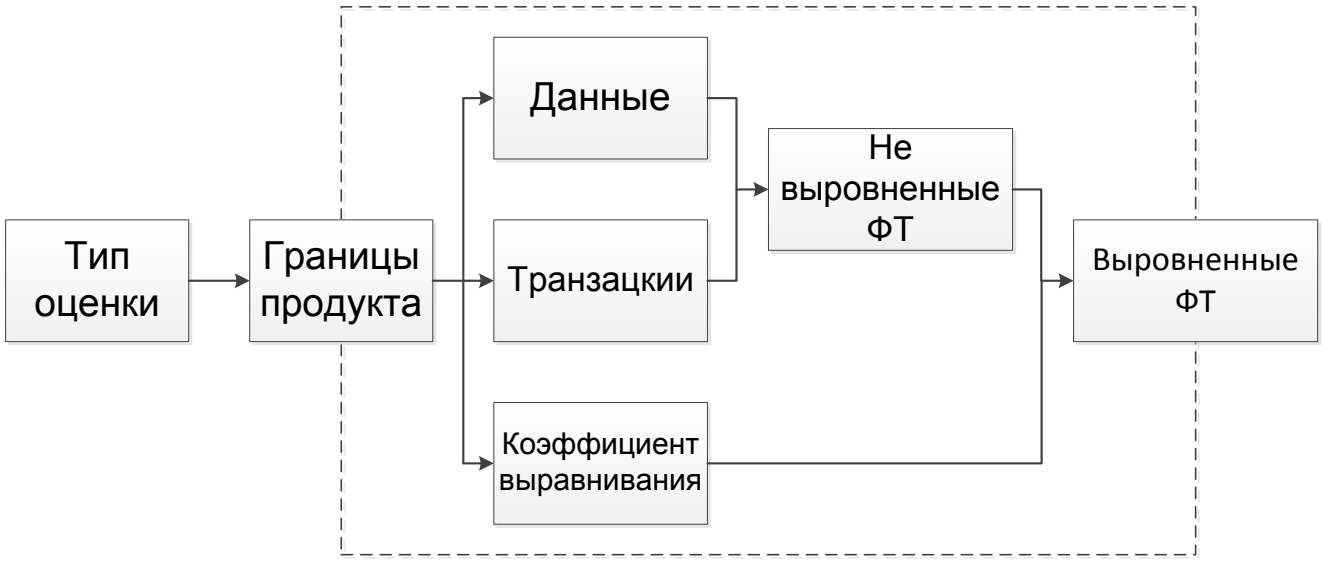


Рисунок 1.2. Анализ функциональных точек

Различают невыровненные функциональные точки и выровненные, которые отличаются от не выровненных тем, что учитывают общесистемные требования, увеличивающие сложность разработки.

Подсчета не выровненных функциональных точек осуществляется за шесть шагов 1-6, а для расчета выровненных требуется еще дополнительные шаги 7-8:

1) определение типа оценки, одного из трех: разработка (первая версия продукта), развитие (последующие версии продукта – добавление, удавление или модернизация функционала), продукт (существующий и поставленный продукт);
2) определение области оценки исходя из ее типа: все разрабатываемые функции (для типа «разработка»), все добавляемые, удаляемые или модернизируемые функции (для типа «развитие»), реально используемые или все функции (для типа «продукт»);

3) определение границ продукта. Определяется «граница продукта» исходя из типа транзакций (передаваемые или принимаемые продуктом) и типа данных: внутренние (поддерживаемые приложением) и все остальные (внешние). На рис. 1.2 граница продукта обозначена пунктирной линией;
4) подсчет функциональных точек, связанных с данными. Определяется, к какой группе относятся логически связанные группы данных или блоки управляющей информации. Внешние интерфейсные файлы (EIFs – External Interface Files) – файлы на которые ссылается продукт, но которые поддерживается вне его. Внутренние логические файлы (ILFs – Internal Logical Files) – поддерживаются внутри продукта. Сложность данных определяется по двум показателям: элемент типа данных (DET – data element type) – уникальное поле данных и элемент типа запись (RET – record element type) – логическая группа данных, состоящая из некоторого количества DET;
5) подсчет функциональных точек, связанных с транзакциями, понимаемыми как неделимый и замкнутый процесс, переводящий продукт из одного состояния в другое. Различают три типа транзакций: внешние входные (EI – external inputs), поступающие извне, внешние выходные (EO – external outputs), выходящие за пределы системы, и внешние запросы (EQ – external inquiries), извлекающие данные по запросу. Сложность определяют по двум характеристикам: количество ссылочных файлов (FTR – file type referenced) – количество различных файлов типа ILFили EIF, которые считываются или изменяются в транзакции; элемент типа данных (DET – data element type) –уникальное поле данных;
6) определение объема продукта в не выровненных функциональных точках как суммы по всем объектам (ILF, EIF) и операциям (EI, EO, EQ);
7) определение коэффициента выравнивания (VAF – value adjustment factor), зависящего от 14 параметров DI_i (degree of influence) системных характеристик продукта. Каждый параметр DI_i оценивается по шкале от 0

до 5. Примеры факторов: эргономика (0 – особых требований нет, 5 – требования очень жесткие), повторное использование (0 – не требуется, 5 – высокая степень повторного использования, стандартная библиотека), гибкость (0 – не требуется, 5 – высокая степень гибкости) и пр. Значения всех параметров суммируются, а коэффициент выравнивания вычисляется по следующей формуле:

$$VAF = \left(0.01 \times \sum_{i=1}^{14} DI_i\right) + 0.65$$

8) определение количества выровненных функциональных точек. Оценка количества выровненных функциональных точек (AFP – adjusted functional points), отражающих новую функциональность, определяется умножением коэффициента выравнивания на количество невыровненных функциональных точек. Если требуется рассчитать количество выровненных функциональных точек для дополнительного функционала, то количество невыровненных функциональных точек складывается с количеством функциональных точек, отражающим дополнительный функционал, и умножается на коэффициент выравнивания.

Положительные стороны использования функциональных точек:

- метод функциональных точек не зависит от языка программирования и среды разработки, поэтому может применяться практически для любого проекта;
- расчет функциональных точек может быть проведен по первичным документам проекта (требованиями, техническим заданиями, руководствам и пр.), что позволяет оценивать размер проекта уже на первых стадиях разработки.

К основным недостаткам использования функциональных точек относят:

- достаточно трудоемкая процедура подсчета, по сравнению с использованием метрики LOC;

- при определении количества функциональных точек, связанных с данными и транзакциями (шаги 3 и 4),оценка их сложности производится экспертом, поэтому разные эксперты могут получать разные результаты.

Модель COCOMO

Модель издержек разработки (COnstructive COst MOdel, COCOMO) [3] является алгоритмической моделью оценки стоимость программного обеспечения, в основе которой лежит формула регрессии с параметрами, полученными из данных выполнявшихся ранее проектов. Модель впервые была опубликована в 1981 г. в качестве формализованной модели для оценки трудоёмкости, затрат и графика программных проектов. Данные по выполненным ранее проектам были получены на основе анализа 63 проектов компании TRW Aerospace, где создатель модели был руководителем отдела исследований и технологий в области ПО. Исследуемые проекты были размером от 2 до 100 KLOC, а используемые языки программирования – от Ассемблера до PL/I. В качестве модели ЖЦ разработки ПО для всех проектов применялась водопадная модель [3].

Модель COCOMO представляет собой иерархию из трех последовательно уточняемых уровней. Базовый уровень подходит для ранних и оперативных оценок стоимости разработки ПО, но его минусом является большая неточность, так как некоторые факторы невозможно учесть на ранних стадиях разработки. Средний уровень COCOMO учитывает эти факторы и позволяет более точно провести оценку. На детальном уровне появляется возможность рассмотреть влияние отдельных фаз проекта на его общие трудозатраты и график.

На *базовом (basic)* уровне рассчитывается стоимость и трудоемкость разработки как функция от оценки размера программы, выраженной в тысячах строк кода (KLOC). Трудоемкость (PM – person-months), срок разработки (TDEV– time of development) и число разработчиков (TM – team members) определяются по следующим трем формулам:

$$\text{PM} = a_b \times (KLOC)^{b_b}\text{[человеко-месяцев]}$$

$$TDEV = c_b \times (PM)^{d_b}\text{ [месяцев]} TM = \frac{PM}{TDEV}\text{ [человек]},$$

где коэффициенты a_b, b_b, c_b и d_b получены эмпирическим путем и представлены в таблице 1.2 для проектов трех типов: органический (organic), предполагает малый размер команды разработчиков с необходимым опытом разработки и нежестким требованиями к создаваемому ПО; полуразделенный (semi-detached), характеризуется командой среднего размера со смешанным опытом разработки и требованиями различной жесткости, предъявляемыми к ПО; встроенный (embedded), который отличается от вышеуказанных наличием большого количество жестких требований к ПО.

Таблица 1.2. Коэффициенты базовой модели COCOMO

Тип проекта	$\mathbf{a_b}$	$\mathbf{b_b}$	$\mathbf{c_b}$	$\mathbf{d_b}$
Органический	2,4	1,05	2,5	0,38
Полуразделенный	3,0	1,12	2,5	0,35
Встроенный	4,6	1,20	2,5	0,32

Исходя из вышесказанного входными параметрами для получения оценок трудоемкости, срока разработки и числа разработчиков в этой модели являются оценка объема кода, который нужно создать, и тип проекта. Собирая и накапливая данные по фактическим трудозатратам в конкретной организации, можно увеличивать точность оценки по этой модели, уточняя значения ее коэффициентов в свете этих данных.

На *среднем (intermediate)* уровне трудоемкость разработки рассчитывается как функция от оценки размера продукта и множества дополнительных «факторов стоимости», таких как субъективные оценки характеристик проекта, продукта, аппаратного обеспечения и персонала. Каждый из этих четырёх факторов, имеет от трех до пяти дочерних характеристик:

1) характеристики проекта – требования соблюдения графика разработки, применение методов разработки ПО, использование инструментария разработки ПО;
2) характеристики продукта – сложность продукта, требуемая надежность ПО, размер БД приложения);
3) характеристики аппаратного обеспечения – ограничения быстродействия при выполнении программы, требуемое время восстановления, ограничения памяти, неустойчивость окружения виртуальной машины;
4) характеристики персонала – опыт разработки, способности к разработке ПО, аналитические способности, опыт разработки на языках программирования, опыт использования виртуальных машин.

Каждой из этих 15 характеристик сопоставляется качественная оценка, начиная от «очень низкого» и до «сверхвысокого», после чего эти оценки заменяются количественными коэффициентами трудоемкости. В таблице 1.3 приведена часть этих коэффициентов на примере фактора «характеристики продукта». Произведение всех коэффициентов трудоемкости составляет регулирующий коэффициент трудоемкости (РКТ), который изменяется в диапазоне от 0,9 до 1,4.

Таблица 1.3. Коэффициенты трудоемкости среднего уровня модели COCOMO

Факторы стоимости	**Рейтинг**					
	очень низкий	низкий	средний	высокий	очень высокий	критический
характеристика продукта						
Сложность продукта	0,70	0,85	1,00	1,15	1,30	1,65
Требуемая надежность ПО	0,75	0,88	1,00	1,15	1,40	-
Размер БД приложения	-	0,94	1,00	1,08	1,16	-

Формула для расчета трудоемкости (PM) среднего уровня имеет вид:

$$PM = a_i \times (KLOC)^{b_i} \times \text{РКТ}\ [\text{человеко-месяцы}],$$

где KLOC – предполагаемый размер программы в тысячах строк кода, РКТ – это регулирующий коэффициент трудоёмкости, рассчитанный ранее, значения

коэффициентов a_i и b_i равны значениям коэффициентов a_b и b_b для базового уровня, за исключением встроенного типа проекта (a_i = 4,6).

Формулы расчета сроков разработки и числа разработчиков те же, что и для базового уровня COCOMO.

Детальный (detailed) уровень COCOMO включает все характеристики среднего уровня с оценкой воздействия факторов, влияющих на стоимость на каждом этапе (анализ, проектирование и т.д.) разработки ПО.

Детальная модель использует свои коэффициенты трудоемкости для каждого атрибута фактора затрат, зависящие от фазы разработки и позволяющие оценить трудозатраты, для каждого этапа. В этой модели COCOMO, вся программа разделена на модули, оценка применяется к каждому модулю в отдельности, после чего все полученные трудоёмкости суммируются.

В детальной модели COCOMO, трудозатраты рассчитываются как функция от размера программы и набора факторов, влияющих на стоимость разработки на каждой фазе жизненного цикла ПО.

Модель COCOMO II

В конце 1990-х годов была предложена модель COCOMO II [28] как более развитая и точная в силу наличия большего числа характеристик программного проекта, продукта и процесса разработки, чем предшествующая модель. По сравнению с базовой моделью COCOMO, новшеством стали масштабируемые показатели программного обеспечения для оценки трудоемкости.

Как и предшествующая модель, COCOMO II разделяется на 3 уровня [29]:

- модель композиции приложения (Application Composition model);
- модель раннего проектирования (Early Design model);
- пост-архитектурная модель(Post Architecture model).

На рисунке 1.3 представлены 3 уровня модели COCOMO II и их влияние на оценку размера ПО на разных стадиях разработки [30].

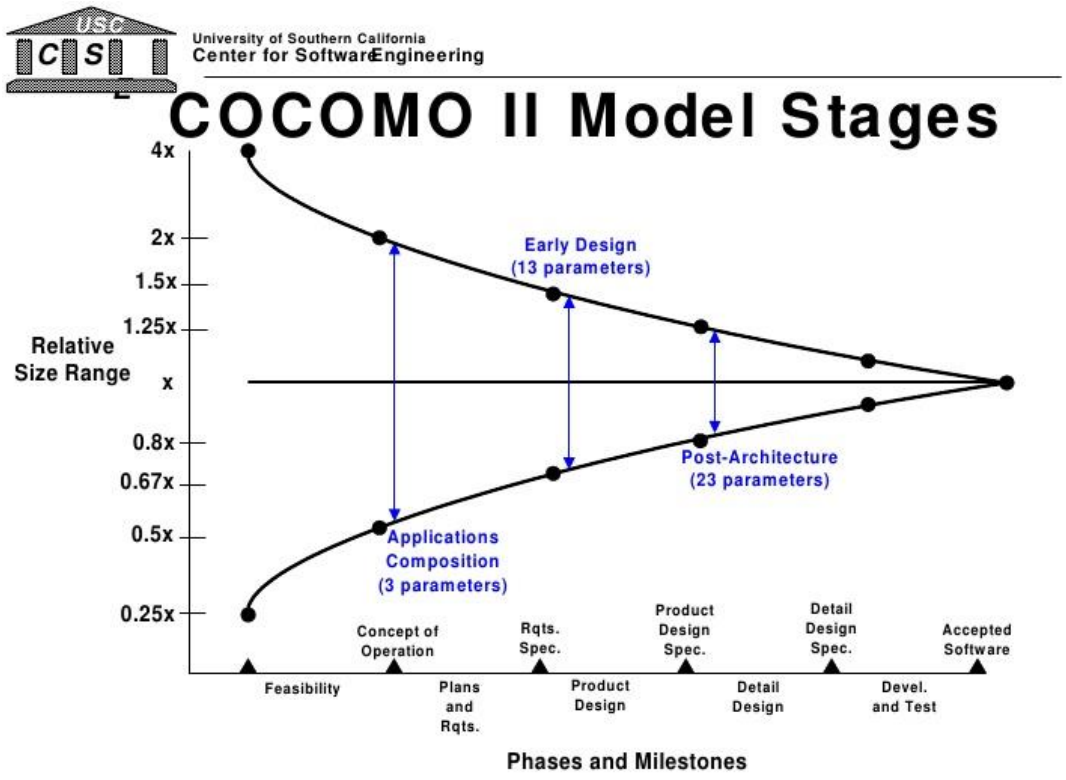


Рисунок 1.3. Оценка размера ПО на различных стадиях разработки. Данные из [30]

Формула для вычисления оценки трудоемкости в модели раннего проектирования и пост-архитектурной модели разработки ПО имеет следующий вид [31]:

$$PM = A \times Size^{E} \times E \times \prod_{i=1}^{n} M_i,$$

где

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j,$$

Формула для расчета времени разработки в модели раннего проектирования и пост-архитектурной модели ПО имеет следующий вид:

$$TDEV = C \times PM^{F},$$

где

$$F = D + 0.2 \times 0.01 \times \sum_{j=0}^{5} SF_j, \text{или}$$

$$F = D + 0.2 \times (E - B)$$

В этих формулах значения *A*, *B*, C и *D* являются эмпирически подобранными значениями в результате анализа графиков и трудоемкости 161 проекта разработки ПО. Их значения: *A* = 2.94, *B* = 0.91, *C* = 3.67 и *D* = 0.28.

Оценка объема продукта *Size* может выражаться в KLOC, либо в не выровненных функциональных точках (UFP); в последнем случае в модели используются либо исторические данные по уже выполненным проектам, собранные в организации, либо средние по отрасли [31]. Объем исходного кода, который может рассматриваться как одна функциональная точка, зависит от типа языка программирования. Например, для языка низкого уровня Assembler оптимистичная оценка – 86, пессимистичная – 320, а наиболее вероятная – 172. Для языка высокого уровня, такого как Java Script, оптимистичная оценка – 44, пессимистичная – 65, а наиболее вероятная – 56.

M_i – это факторы стоимости; для модели раннего проектирования их 6, для пост-архитектурной модели –16.

SF_j – это масштабируемые показатели ПО.

В качестве примеров факторов стоимости (Software Cost Drivers) можно привести следующие: соответствие документации продукта жизненному циклу ПО (documentation match to lifecycle needs), атрибуты продукта (product attributes), преемственность коллектива (personnel continuity), код для повторного использования (developed for reusability), атрибуты команды разработчиков (personnel attributes), опыт работы с платформой (platform experience) и пр.

Масштабируемые показатели ПО (Software Scale Drivers): уровень зрелости процесса разработки (process maturity), наличие опыта аналогичных разработок (precedentness), решения по архитектуре/рискам (architectural/risk resolution),

гибкость самой разработки (development flexibility), сплоченность команды разработчиков (team cohesion).

Как и на промежуточном уровне модели COCOMO, значения факторов стоимости и масштабируемых показателей ПО выбираются по 6-ти бальной шкале: очень низкий, низкий, номинальный, высокий, очень высокий и сверхвысокий. В значении «номинальный» все коэффициенты равны 1 и отклоняются в ту или иную сторону для повышающих или понижающих оценок.

Модель SLIM (модель Путнэма)

Модель Путнэма (SLIM – Software LIfe-cycle Model – модель жизненного цикла ПО) – это алгоритмическая модель оценки трудоёмкости разработки ПО, предложенная в 1978 г. [4] и основанная на распределении Рэлея. Данная модель наиболее приспособлена для оценки крупных проектов, так как ее базу составляют реальные данные Министерства обороны США. Это одна из первых моделей, в которых применялись эмпирические данные, так как с 1970-х годов появились научные публикации и ряд работ [2], в которых указывалось, что трудоёмкость и стоимость находятся в нелинейной зависимости от объема работ.

Трудоемкость в данной модели рассчитывается по следующей формуле:

$$PM = \left(\frac{Size}{P \times T^{4/3}}\right)^3 \times B,$$

где *Size* – размер программного обеспечения в любых используемых в организации единицах, P – продуктивность, способность организации разрабатывать ПО требуемого объема и качества, T – ограничения на время разработки, B – масштабируемый коэффициент, определяемый эмпирическим путем для каждой организации-разработчика.

Суть метода в том, что собранные данные о проекте разработки ПО (например, трудоемкость и размер) «подгоняются» к кривой трудоемкости. На

рисунке 1.4 представлены 3 кривых трудоемкости, зависящих от времени разработки ПО.

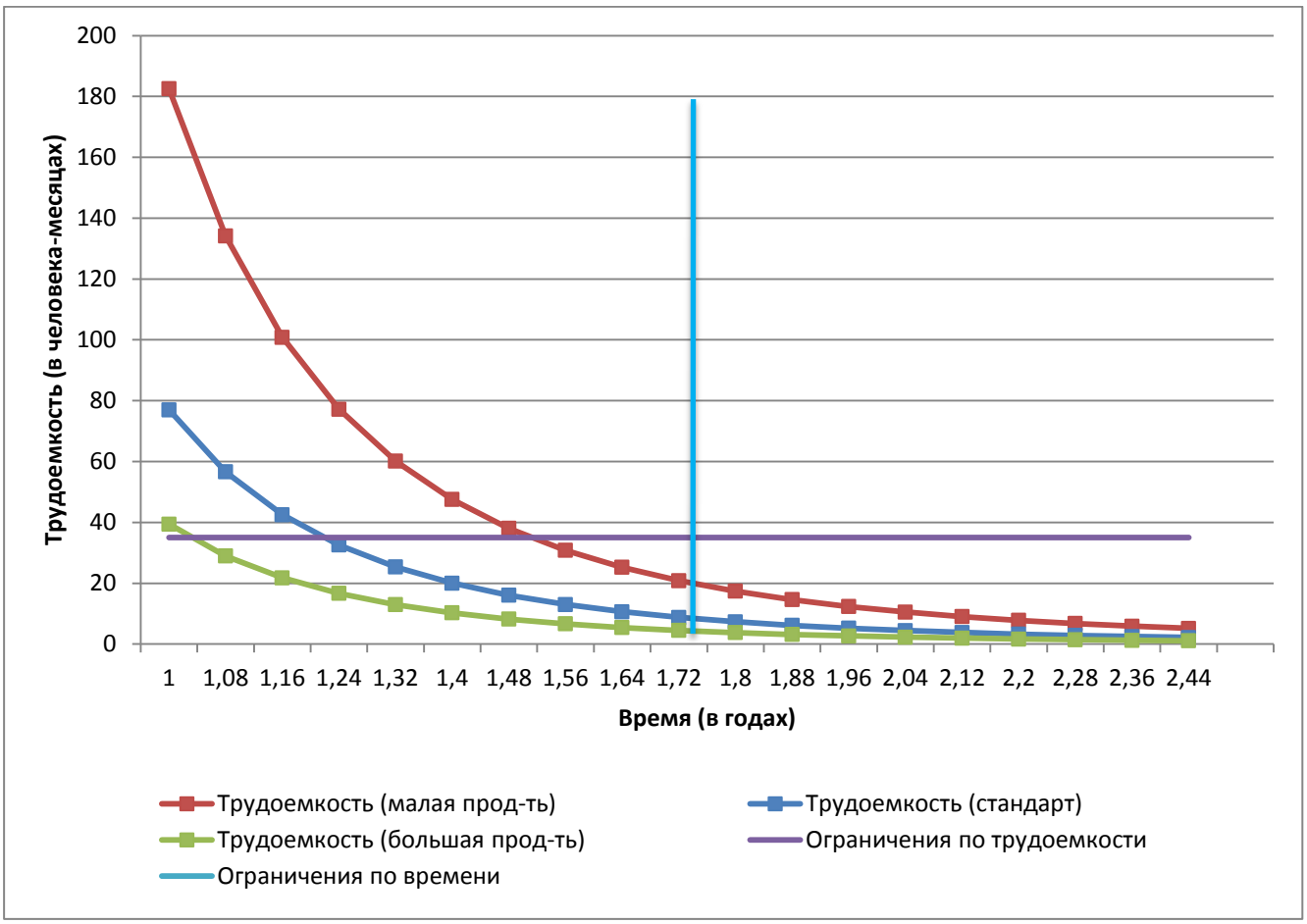


Рисунок 1.4. Зависимость трудоемкости от времени разработки в модели SLIM

Как видно из рисунка, особенность модели состоит в том, что значение трудоемкости уменьшается, с увеличением времени на завершение проекта разработки ПО.

К преимуществам модели SLIM можно отнести: широкие возможности по калибровке модели, простая зависимость, связывающая затраты на программное обеспечение и трудоемкость на его разработку, а также то, что для расчета используется меньше параметров, по сравнению с моделями COCOMO/COCOMO II.

К недостаткам модели SLIM можно отнести то, что ее оценки очень чувствительны к изменениям в параметрах разработки, а также слабую применимость этой модели к малым проектам.

1.1.2 Неалгоритмические методы

В отличие от алгоритмических методов, неалгоритмические основываются не на математических формулах, а на основополагающих принципах и схемах, которые рассмотрены ниже.

Оценка по Паркинсону

Принцип, лежащий в основе вышесказанного метода [8] гласит, что объем работы напрямую зависит от времени, которое выделено на ее выполнение. То есть объем работы растет так, чтобы занять все выделенное для нее время: «Work expands to fill the available volume» [24]. Таким образом, указанный принцип описывает взаимодействие некоторых структурных элементов крупных институтов, административных центров между собой, когда ресурсы используются абсолютно неэффективно, без учета производственных ресурсов, по причине их бюрократического обременения. В этом случае, чтобы изменить существующую ситуацию, данный принцип можно применить в следующем виде: "*С целью повышения производительности труда разработчика, уменьшают время, отводимое на разработку программного обеспечения*".

Oracle AIM

Oracle AIM (Application Implementation Methodology) – это методология внедрения готовых приложений от компании Oracle [33]. Несмотря на то, что методология рассчитана на использование при внедрении уже готовых приложений, ее можно использовать и для оценки ресурсов. Суть методологии заключается в том, что для каждой задачи проекта готовится подробное описание с определением последовательности выполнения, а также ответственных лиц (ЛПР). Под задачей в данной методике понимается неделимый объем работ, оканчивающийся фиксируемым результатом. На рисунке 1.5 представлена схема разбиения задач по фазам внедрения и процессам.

Рисунок 1.5. Разбиение задач по фазам и процессам

Группировка задач в процессы, осуществляется по принципу единства результата. Далее происходит оценка каждой задачи, после чего оценки суммируются. Для каждой работы по проекту определяется временная фаза. Так как ход работ разделен на отдельные задачи, появляется возможность отслеживать ход выполнения проекта в рамках каждой фазы внедрения и, тем самым, контролировать ход проекта.

Экспертная оценка

Данный метод основывается на сборе оценочных мнений инженеров-разработчиков программного обеспечения, которые впоследствии сами оценивают разрабатываемый продукт, проект или его часть. В дальнейшем предложения и оценки протоколируют и обсуждают на общем собрании, а результаты оценки интегрируют в единую систему. Вся процедура основывается на методе Дельфи (дельфийском методе) [33], который ориентирован на достижение консенсуса при принятии решения. Таким образом, при грамотном обобщении оценок можно получить коллективное мнение профессионалов, которое будет обладать более высокой степенью достоверности, чем простое усреднение этих оценок. Данный метод уместен при оценке проектов и

процессов, направленных на решение инновационных задач. По мере увеличения итераций на следующей стадии покомпонентной оценки, ее точность увеличивается.

Оценка по аналогии

Некоторые специалисты считают этот метод разновидностью метода «Экспертная оценка» [33], однако, принято выделять его в самостоятельный метод. Метод оценки по аналогии основан на использовании эмпирических данных о ранее выполнявшихся проектах, только в отличие от алгоритмических методов (где эти данные используются в основном для настройки модели), метод оценки по аналогии позволяет определить похожие проекты-аналоги и состоит из трех шагов:

1) *сбор данных по проекту.* На данном шаге выполняется сбор экспертных оценок и отбор характеристик, на основании которых уже будут сравниваться новый проект с выполнявшимися ранее проектами;
2) *поиск и анализ проектов-аналогов.* На основе характеристик, полученных на предыдущем шаге, выбираются несколько проектов, у которых наименьшая разница в оценках их численных характеристик. Помимо этого, для отбора проектов можно применить метод измерения евклидова расстояния в *n*-мерном пространстве этих характеристик. Далее для каждой характеристики определяется ее вес (множитель), который определяет ее значимость для проекта в целом. Для простоты вычисления можно принять вес характеристики равный единице, таким образом, все характеристики проекта являются равнозначными. Далее характеристики и проекты отображаются в *n*-мерном пространстве как точки, и по из значениями вычисляется евклидово расстояние между ними. В результате, наиболее схожие проекты с наименьшим расстоянием от нового проекта будут считаться его проектами-аналогами;

3) *экспертная оценка нового проекта.* Этот этап является последним, за основу оценки берутся значения характеристик найденных проектов-аналогов и непосредственно по ним проводится оценка нового проекта.

1.2 Программные системы оценки ресурсов

Как правило, функцию для оценки ресурсов не выделяют в отдельную программу, так как она представляет один из шагов процесса разработки ПО и является частью ПО для управления проектами. В данном обзоре рассмотрим программные системы для управления проектами, которые делят на 3 группы по типу используемой технологии:

1) настольные системы (desktop): Deltec Open Plan, семействоArtemis Views, Spider Project, Microsoft Project;
2) клиент-серверные (client-server): Basecamp, JIRA;
3) веб-сервисы (web-based): Wrike, Worksection.

Настольные системы

Настольные системы предоставляют развитые средства планирования, контроля и анализа проектов, но требуют высокую квалификацию пользователей, а также существенных временных затрат на подготовку и анализ исходных данных.

Особенностями настольных систем, которые отличают их от других классов, являются практически неограниченное количество планируемых задач, поддержка различных уровней детализации проекта, оптимизационные алгоритмы составления расписания и пр.

Ниже представлены краткие характеристики нескольких типичных настольных систем планирования.

Deltec Open Plan – мощная и гибкая система планирования профессионального уровня для управления большими и средними проектами [35]. Это программное решение предоставляет функции планирования и отслеживания

выполнения проекта в рамках установленного графика и бюджета. При помощи мультипроектного анализа, планирования критического пути, Open Plan обеспечивает мощь и гибкость в обслуживании различных потребностей бизнеса.

Особенности Deltec Open Plan [36]:

- баланс – обеспечивается сбалансированность использования ограниченных ресурсов между несколькими проектами; руководитель проекта может видеть все множество выполняемых в данный момент проектов, в которых эти ресурсы задействованы, что позволяет эффективно управлять их распределением;
- управление ресурсами – есть возможность выделять приоритетные проекты в соответствии с текущими целями организации, причем руководители проектов могут создавать и совместно использовать общие схемы разделения ресурсов между несколькими проектами;
- эффективное планирование проекта –обеспечивается оперативное и реалистичное планирование путем быстрого ввода данных и анализа, а также возможностью следить за разработкой за счет эффективной системы отчетов о состоянии проекта и хода разработки.

Семейство**Artemis Views** – это целый спектр программ, состоящий из 4-х модулей для автоматизации различных аспектов управления проектами [37]: Project View (проект в целом), Track View (отслеживание), Resource View (ресурсы), Cost View (себестоимость). Модули этих семейств имеют совместимый формат данных и благодаря поддержке ODBC (Open Database Connectivity – программный интерфейс доступа к базам данных, разработанный фирмой Microsoft) легко стыкуются с распространенными СУБД SQL Server, Sybase, Oracle, так как работают в клиент-серверной архитектуре. Каждый модуль семейства может функционировать как совместно с другими, так и по отдельности.

Модуль *Project View* предоставляет многопользовательскую и многопроектную систему для планирования и управления, обеспечивающую механизм разграничения доступа. Также модуль обладает целым набором встроенных средств для формирования различных отчетов, как при помощи собственного, так и стороннего ПО (Quest и пр.).

Модуль *Track View* предоставляет средства для отслеживания и анализа плана выполнения работ, включая обзор бюджета, графика и пр. Модуль обеспечивает различную глубину информации: подробные отчеты для руководителей проектов и групп, и более укрупненные отчеты с общими показателями по проектам для руководителей организации.

Модуль *Resource View* – это подсистема для планирования и контроля использования ресурсов. Обеспечивает средства по оптимизации распределения ресурсов и выравниванию загрузки.

Модуль *Cost View* отвечает за сбор и централизованное хранение проектной информации по доходам и расходам в проекте. При помощи данного модуля можно как рассчитать экономическую эффективность проекта, так и спрогнозировать затраты, необходимые для его завершения.

Spider Project – это система управления проектами [38], разрабатываемая российской компанией «Спайдер Проджект» [39]. Система проектировалась и разрабатывалась с учётом значительного практического опыта, потребностей и особенностей российского рынка информационных технологий. Упор на визуализацию данных (гистограммы, организационные, поточные и сетевые диаграммы, диаграммы Гантта и таблицы с различными разрезами данных) позволяет пользователям охватить все проект целиком и всесторонне его анализировать.

Особенности Spider Project:

- оптимальное распределение ресурсов по проектам за счет построения такого расписания выполнения работ, при котором снижается общее время простоев в ожидании освобождения занятых ресурсов;
- число работ, задач и ресурсов по проектам ограничено максимально возможным целым (2^{64}–1) в реализации системы;
- возможность создания и подключения различных БД, в том числе по расходам материалов, нормативным расценкам, по производительности типовых работ и пр.;
- наличие подсистемы анализа и управления рисками;
- наличие подсистемы управления резервами;
- наличие подсистемы расчета вероятностей успеха (трендов).

Microsoft Project – это система управления проектами [40], разрабатываемая и продвигаемая компанией Microsoft. При помощи Microsoft Project можно составлять планы, распределять ресурсы, отслеживать ход разработки и анализировать состояние работ по проекту. Программа обеспечивает расчет критического пути, учитывая имеющиеся ресурсы, которые отображаются на диаграмме Гантта.

Microsoft Project – это целое семейство программных продуктов. Microsoft Project Standard – однопользовательская версия программы для средних и малых организаций. Microsoft Project для Office – версия для корпоративного использования, реализующая совместное управление ресурсами и проектами. Microsoft Project Online – облачный сервис, реализующий функционал настольных приложений. Microsoft Project Server–решение для работы с портфелем проектов в рамках корпоративной сети. Microsoft Project Lite – приложения для работы с проектами, управление которыми осуществляется с помощью Project Online и Project Server.

Клиент-серверные системы

Basecamp – система, имеющая онлайн-сервис для управления проектами [41], совместимый со многими приложениями и программами, первоначально был создан для малых предприятий. С 2005 года разработчики «37 signals» [42] проводят работу по расширению функциональные возможности программы.

Особенности системы:

- просмотр на одном экране ключевой информации о проектах и клиентах;
- отслеживание плана работ;
- загрузка, классификация и поддержка версионности файлов;
- форумы для ведения обсуждения;
- создание расписания и управление ключевыми вехами проекта;
- сохранения файлов в облачном сервисе компании;
- возможность разрабатывать дополнения к Basecamp, используя стандартный набор API.

Одним из основных недостатков Basecamp считается малая пригодность для больших предприятий и ведения крупных и долгосрочных проектов.

JIRA/Atlassian JIRA – система для отслеживания ошибок и управления проектами [43] с веб-интерфейсом, разработана Atlassian Software Systems [44]. Изначально разработчики создавали систему JIRA, с целью вытеснить конкурирующий продукт – Bugzilla, поэтому система во многом повторяет ее архитектуру. JIRA основана на Java EE и обеспечивает поддержку распространённых БД (Oracle, PostgreSQL, MySQL, Microsoft SQL) и операционных систем (Microsoft Windows, Linux, Solaris) [45].

JIRA поддерживает интеграцию с популярными системами управления версиями (CVS, Subversion, Clearcase, Git и др.) [46]. Развитая архитектура плагинов системы позволяет создавать расширения для нее, что делает систему применимой для управления проектами во многих сферах.

Для каждого создаваемого проекта в системе создается схема безопасности и оповещения. Пользователь создает задачу, содержащую название проекта и другие его параметры, в дальнейшем он может ее расширить дополнительными полями, комментариями и пр. В свою очередь, для каждого приложения тоже может быть определен тип задачи, с несколькими видами представления. С помощью создаваемых схем пользователь может определить для конкретного проекта тип доступа, поведение и видимость полей.

Веб-сервисы

Wrike – онлайн-инструмент [47], разрабатываемый одноименной компанией [48]. Его главная особенность – интеграция с электронной почтой. Сервис позволяет планировать проекты, назначать задачи, одновременно работать в режиме онлайн нескольким пользователям. Планы подписки сервиса ограничены по количеству пользователей, так как он разрабатывался, прежде всего, для предприятий малого и среднего бизнеса.

Основные функции сервиса Wrike:

- задание приоритетов для задач;
- диаграмма Гантта с динамическим обновлением;
- учёт времени работы и нагрузки персонала;
- новостная лента в режиме онлайн;
- приложения для мобильных телефонов;
- интеграция с почтовыми сервисами (Outlook и Apple Mail), а также с сервисами облачного хранения (Google Docs и Dropbox) с возможностью онлайн изменения файлов онлайн;
- настраиваемые отчеты, комментарии пользователей.

Worksection – веб-сервис по планированию проектов предприятия, которая появилась на рынке в 2008 году [49]. Система доступна для пользователей на русском, английском и украинском языке, имеет следующие функции:

- создавать задачи и подзадач, указывать приоритет, ответственных и сроки завершения;
- календарь с поддержкой диаграммы Гантта;
- учет времени и таймер;
- учет финансов;
- SSL-шифрование данных;
- хранение файлов с возможностью подключить FTP хранилище;
- размещение аккаунта на домене клиента;
- резервное копирование данных.

Многие пользователи считают интерфейс сервиса перегруженным и неудобным [50]. Планирование задач в системе строится в виде иерархии, состоящей из 3-х уровней. Для каждой задачи есть возможность настроить доступ и права отдельно, пользователю доступна возможность прикрепить к каждой задачи отдельный файл. Проекты и задачи можно ранжировать по степени важности от 1 до 10. Еще один недостаток – во внешнем FTP хранилище нельзя создавать отдельные папки, а только ставить метки, что в дальнейшем приведет к затруднениям при поиске некоторых файлов в хранилище. Также отсутствует аналитика по проектам. В основном сервис рассчитан для небольших предприятий.

Ниже представлена сравнительная таблица сервисов по ведению проектов.

Таблица 1.4. Программные системы управления проектами

Наименование	**Лицензия**	**Интерфейс**	**Интеграция с системами управления версиями**	**Настраиваемые поля**	**Расчет критического пути**
Deltec Open Plan	проприетарная	настольная	-	+	+
Artemis Views	проприетарная	настольная	-	-	+
Spider Project	проприетарная	настольная	-	+	+
Microsoft Project	проприетарная	настольная, web	-	+	+

Наименование	Лицензия	Интерфейс	Интеграция с системами управления версиями	Настраиваемые поля	Расчет критического пути
Basecamp	проприетарная	web	-	+	-
JIRA	проприетарная	web	CVS, Subversion, Clearcase, Git	+	+
Wrike	проприетарная	web-интерфейс, эл. почта	Subversion	+ (в корпоративном плане)	+
Worksection	проприетарная	web	-	+	-

1.3 Язык алгоритмических сетей

Во второй половине 1980-х годов был предложен подход, названный *новой информационной технологии* (НИТ) [51, 52] для создания ПО, с которым конечный пользователь мог бы работать напрямую, исключая посредников в лице математиков и программистов.

В рамках этого подхода, который развивал д.т.н., профессор, Заслуженный деятель науки РФ Иванищев Вячеслав Васильевич, были опубликованы работы, в которых были сформулированы подходы к созданию технологии моделирования, ориентированной на конечного пользователя (экономиста), в том числе предлагался графический язык для представления моделей [11, 12].

Дальнейшие работы в этом направлении позволили создать методологию, ориентированную на решение проблем конечного пользователя в сфере моделирования (методология моделирования на основе алгоритмических сетей), в основе которой лежит представление математической модели исследуемого объекта в виде алгоритма, структуризация вычислительных процедур которого осуществлена в соответствии с представлениями пользователя о причинно-следственных или временных связях явлений, которые характеризуют моделируемый объект.

В качестве языка представления используется графический язык, оперирующий исключительно функциональными зависимостями, имеющими

место в предметной области, которой принадлежит исследуемый объект. Язык получил название язык алгоритмических сетей (ЯАС).

Основными понятиями языка алгоритмических сетей являются понятия функционального оператора и интерфейсной дуги.

Функциональный оператор реализует функциональное отношение, связывающее явления, которые представляют объект моделирования. Графическим отображением функционального оператора является круг, внутри которого помещен символ реализуемого оператором класса функций. Конкретный вид функции определяется множеством входных и выходных дуг, связанных с данным оператором.

1.4 Постановка задачи

Вследствие того, что в методе гибких оценок, для определения проекта-аналога используется множество характеристик проекта, можно отнести данный метод к классу многокритериальных методов принятия решений. Факторы, определяющие процесс формирования множеств характеристик, можно представить в виде кортежа:

$$<M_D, O, D_O, C, D_C, R>,$$

где:

M_D – формализованная модель, описывающая формирование исходного, уточненного и рекомендуемого множеств в виде функции, значения которой задаются при помощи таблицы и задают варианты рекомендуемого множества характеристик;

O – исходное множество характеристик, которое данный момент использует компания-разработчик;

D_O – решающее правило перевода характеристик из исходного в уточненное в виде количественной оценки; Критерием перевода характеристики является

обеспеченность информацией, иными словами, характеристики значения по которым отсутствуют, остаются в исходном множестве.

C – множество уточнённых характеристик, полученное из множества *O* путем количественного анализа;

D_C – решающее правило перевода характеристик из уточненного в рекомендуемое множество в виде качественной оценки; Качественная оценка подразумевает проверку возможность классификации множества характеристик по базе проектов компания-разработчика. Если классификация успешно проведена, то все характеристики, которые попали в классификатор переводятся в рекомендуемое множество, не попавшие – остаются в уточненном.

R – множество рекомендуемых характеристик, полученное из уточненного множества *C* путем качественного анализа.

Факторы, определяющие процесс выбора проекта-аналога, можно представить в виде следующего кортежа:

$$<M_D, O, D_O, C, D_C, R, DMP, D_{NF}>,$$

где:

M_D – формализованная модель, которая описывает множество возможных решений (функция, описывающая модель выбора проекта-аналога) в виде функции, значения которой задаются при помощи таблицы с вариантами рекомендуемого множества характеристик;

O – исходное множество проектов, которые выполнила компания-разработчик.

D_O – решающее правило перевода проекта из исходного в уточненное в виде количественной оценки; Критерием перевода является обеспеченность проекта информацией, иными словами, проект, у которого заполненные значения характеристик составляют менее 30%, остаются в исходном множестве.

C – множество уточнённых проектов, полученное из множества *O* путем количественного анализа;

D_C – решающее правило перевода проекта из уточненного в рекомендуемое в виде качественной оценки; Качественная оценка подразумевает проверку возможность классификации множества проектов содержащихся в базе компании-разработчика. Если классификация успешно проведена, то все проекты, которые попали в классификатор переводятся в рекомендуемое множество, не попавшие – остаются в уточненном.

R – множество рекомендуемых проектов, полученное из уточненного *C*путем качественного анализа, из которых лицо, принимающее решение, выбирает проект-аналог;

DMP – лицо, приникающее решение (ЛПР), например, руководитель проекта, которое является компетентным в данной области и имеет целостное представление образа проекта, что делает возможным привлечение его плохо формализуемых знаний;

D_{NF} – решающее правило, руководствуюсь которым, ЛПР осуществляет выбор на основе неформализованных знания о данной предметной области.

В итоге, мы имеем симбиоз формализованного метода (в нашем случае метода вычисления оценок) и неформализуемого аспекта (опыт, мышление) эксперта.

1.5 Выводы

Высокая сложность ПО обусловлена огромными размерами и огромным количеством состояний, в которых это ПО может находиться. Некоторые программы содержат миллионы строк кода и должны выполняться так, как задумано разработчиками. По этому задача правильного оценивания ресурсов не только остается актуальной, но и становится все более важной в процессе разработки ПО.

Анализ существующих методов оценки ресурсов показывает: алгоритмические методы в целом справляются с задачей оценки, но отражают специфику той предметной области, в которой поводилась разработка и апробация данных методов, то есть они работают на «своем поле». Помимо этого в каждом из методов присутствуют настроечные коэффициенты, значения которых необходимо «подгонять» для более точной оценки. Неалгоритмические методы позволяют привлечь слабо формализуемые знания эксперта, но к их недостаткам можно отнести слабую обоснованность полученной оценки, а также необходимость согласования, при наличии нескольких оценок.

Программные системы оценки ресурсов позволяют отслеживать выполнение проекта на всем сроке, следить за установленными параметрами графика и бюджета, обеспечивают расчет критического пути, построение диаграммы Гантта и прочее, но не отражают специфику той организации, в которой они применяются.

Учитывая эти факты, является целесообразным разработки метода, который бы объединял преимущества алгоритмических и неалгоритмических методов оценки ресурсов, то есть использовал некоторый формальный аппарат для подсчета ресурсов и привлекал слабо формализуемые знания эксперта, а также отражал бы специфику той организации, в которой применяется.

Глава 2 Модель формирования базы выполненных проектов

В ходе создания базы выполненных проектов для компании, разрабатывающей программное обеспечение («Эксиджен Сервисис») [55], Санкт-Петербургским институтом информатики и автоматизации РАН была предложена и апробирована модель [22] формирования базы выполненных проектов для оценки необходимых ресурсов.

Модель опирается на принципы моделирования на основе алгоритмических сетей, а также методологии принятия решений на основе алгоритмических сетей (прозрачные технологии), разработанных в Санкт-Петербургском институте информатики и автоматизации РАН (СПИИРАН) [15, 56-63].

Модель включает следующие этапы:

1) формирование *исходного множества источников*, в которых может (должна) содержаться информация о выполняемых (выполненных) проектах, их содержании, особенностях и ходе.
2) формирование и структуризация *исходного множества характеристик*, содержащих информацию о выполняемых (выполненных) проектах, их содержании, особенностях и ходе выполнения.
3) формирование *исходного множества выполнявшихся проектов*, обеспеченных источниками информации и перечня источников для каждого проекта.
4) уточнение исходного множества характеристик, исходя из степени их обеспеченности проектной информацией и представления об их значимости для поиска проектов-аналогов – формирование *уточненного множества характеристик*.
5) уточнение исходного множества проектов, исходя из степени их обеспеченности информацией уточненного множества характеристик – формирование *уточненного множества проектов*.

6) уточнение исходного множества источников, исходя из степени их использования в уточненных множествах проектов и характеристик – формирование *уточненного множества источников*.
7) проведение испытаний, подтверждающих достаточность уточненного множества характеристик – формирование *рекомендуемых множеств источников, характеристик и проектов*.

На рисунке 2.1 представлена принципиальная схема модели.

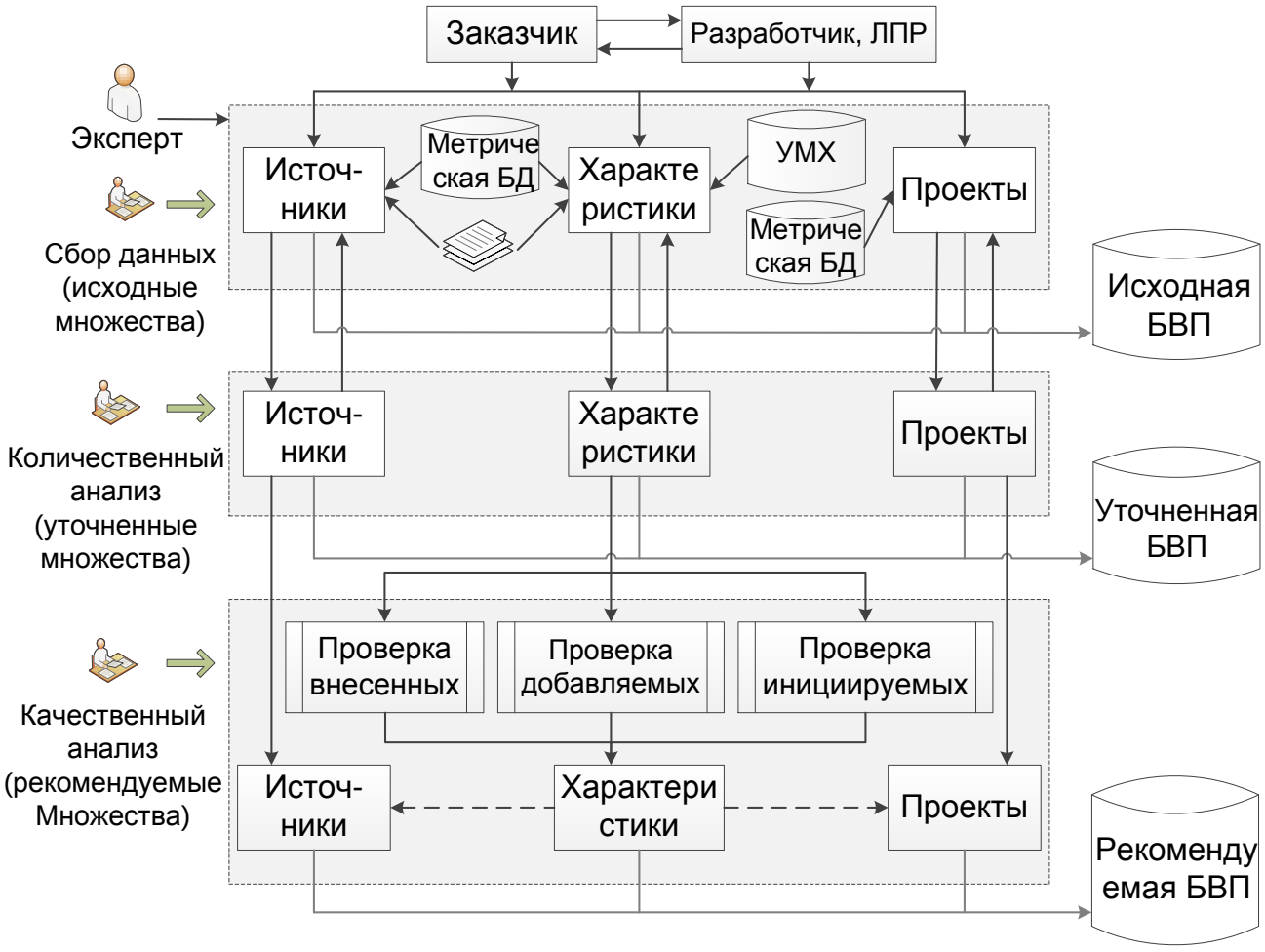


Рисунок 2.1. Модель формирования базы выполненных проектов

Агрегировано модель предстаёт в виде трех крупных этапов:

- *формирование исходных множеств* источников, характеристик и проектов (этапы 1-3). Исходные множества формируются на основании анализа процесса компании и не содержат синонимов и объектов, обладающих тождественной семантикой;

- *формирование уточненных множеств* источников, характеристик и проектов (этапы 4-6). Уточнение множества формируются на основе анализа обеспеченности каждого элемента проектной информацией и представления об их значимости для поиска проектов-аналогов;
- *формирование рекомендуемых множеств* источников, характеристик и проектов (этап 7). Рекомендуемые множества формируются на основании испытаний, подтверждающих достаточность исследуемого множества характеристик для решения задачи поиска проекта-аналога.

2.1 Формирование исходных множеств

2.1.1 Источники

На данном этапе происходит анализ источников, в том числе документов, создаваемых в ходе выполнения проектов. Документы просматриваются на предмет нахождения в них характеристик, которые могут быть использованы для поиска проектов-аналогов и последующего их анализа, выполняемых для снижения риска неудачного завершения инициируемых проектов.

В основе анализа лежит множество шаблонов документов, используемых в стандартном производственном процессе организации (СППО) [64, 65].

Результатом анализа является исходное множество источников, из которых могут быть взяты значения характеристик, помещаемых в базы выполненных проектов компании. В качестве примеров таких источников могут быть названы *знания эксперта*, *метрическая корпоративная база данных компании*, а также *документы, создаваемые на основе шаблонов*, принятых в СППО. Документы, создаваемые на основе шаблонов в ООО «Эксиджен Сервисис»:

1) Simplified Statement of Work (SSOW) – краткое положение о работе. В данном документе кратко перечисляется основные параметры проекта, такие как: наименование проекта, описание проекта, даты начала и окончания, модель ЖЦ, тип проекта, тип бюджета, проектные цели и ограничения, процедуры и критерии приемки и контакты.

2) Statement of Work (SOW) – развернутое положение о работе. Данный документ является более подробной версий предыдущего. SOWфиксирует и определяет основные работы по проекту, которые должен выполнить исполнитель, их сроки и результаты. SOW, как правило, включает в себя подробные требования и финансовую информацию.
3) Project Plan (PP) – план проекта. В плане проекта содержится информация о ЖЦ проекта и основы вехах, метриках проекта, ресурсах, проектных рисках, информация по обеспечению качества, управлению требованиями и пр. Это основной документ проекта, в который вносится информация по ходу его выполнения.
4) Project closure Report (CR) – заключительный ретроспективный отчет. В данном документе приводится информация по трудоёмкости каждого члена команды, с разбиением по дням; Опросный лист с оценками руководителя проекта, в который входят вопросы по успешности проекта, удовлетворённости заказчика, полноты требований, точности планирования, производительности разработчиков, достаточности ресурсов, следованию стандартному процессу организации и пр., а также выводы по проекту («Lessons learnt»); т.е., та информация, которую руководитель проекта хочет донести до руководства организации.

Атрибуты исходного множества источников представлены в табл. 2.1.

Таблица 2.1. Атрибуты источника

№	Наименование атрибута	Возможные значения	Примечания
1	Уникальный номер источника	Число	Используется для однозначной идентификации источника
2	Наименование источника	Текст	Символьная строка, определяющая источник (SSOW, PP, Эксперт и пр.)
3	Описание источника	Текст	Символьная строка. Раскрывает смысловое содержание источника.

№	Наименование атрибута	Возможные значения	Примечания
4	Множество, которому принадлежит источник	Возможные значения: исходное множество (ИМ); уточненное множество (УМ); рекомендуемое множество (РМ)	Принадлежность источника к УМ указывает на принадлежность его к ИМ. Принадлежность источника к РМ указывает на принадлежность его к УМ, а также к ИМ (см. рис. 2.2).
5	Используемость в характеристиках	Число	Количество характеристик, использующих источник. Формируется автоматически
6	Используемость в проектах	Число	Количество проектов, использующих данный источник. Формируется автоматически
7	Дата последнего изменения	Дата	Формируется автоматически
	Автор изменения	Текст	Формируется автоматически

На рисунке 2.2 представлена связь исходного, уточненного и рекомендуемого множества.

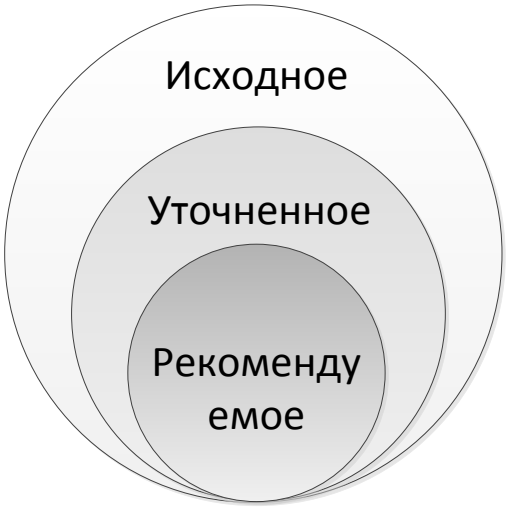


Рисунок 2.2. Связь исходного, уточненного и рекомендуемого множества

Необходимо отметить, что в ходе работы компании, по мере совершенствования ее процесса, шаблоны подвергаются модернизации, а, следовательно, меняется структура и форма представления составляемых на их основе документов. Иными словами характеристики и их значения, а также местоположение в документах для различных проектов могут отличаться. Данный

факт обуславливает необходимость *хранения прямых ссылок на местоположение в документе того или иного значения*.

2.1.2 Характеристики

Параллельно с формированием множества источников определяется исходное множество характеристик, позволяющих описать любой проект, выполнявшийся или выполняющийся в компании с целью последующего их анализа при запуске новых проектов. Исходное множество характеристик проектов определяется на основании анализа:

- множества характеристик проектов, определяемого документацией, хранящейся в метрической базе данных организации [66];
- множества характеристик проектов, содержащихся в аналогичной документации других компаний разработчиков, которые находятся в открытом доступе [67];
- множества характеристик, используемых при выборе модели процесса разработки ПИ [68];
- универсального множества характеристик (см. приложение 3).

Формирование исходного множества выполняется в 3 шага:

1) формируется *общее* множество характеристик, путем объединения множеств характеристик содержащихся в перечисленных выше источниках, а также множеств характеристик содержащихся в стандартах и публикациях по вопросам управления проектами [1, 2, 24, 64, 65, 69-70];
2) на основании семантического анализа универсального множества характеристик, заключающегося в поиске дублирующих и аналогичных по смыслу характеристик, происходит переход от универсального множества характеристик к множеству, содержащему исключительно *уникальные характеристики* в рамках данного множества;
3) для каждой уникальной характеристики происходит определение значения атрибутов, описывающих эту характеристику в исходном

множестве, на основании чего и происходит окончательное ее формирование (*уникальные характеристики, которые могут быть описаны в атрибутах характеристик исходного множества*). Атрибуты характеристик, представлены в табл. 2.2.

Таблица 2.2. Атрибуты характеристики исходного множества

№	Наименование атрибута	Возможные значения	Примечания
1	Уникальный номер характеристики	Число	Используется для однозначной идентификации характеристики
2	Наименование характеристики	Текст	Символьная строка, определяющая характеристику
3	Описание характеристики	Текст	Используется в качестве подсказки: раскрывает смысловое содержание характеристики
4	Тип характеристики	Одно из возможных значений: интегральная, вычисляемая, число, текст, шкала, дата	Интегральная – используется для формирования структуры характеристик. Вычисляемая – значение характеристики определяются путем преобразования значений других характеристик. Шкала — некоторое множество меток и их описания.
5	Описание области задания характеристики	Текст	Используется в качестве: 1. Подсказки и контроля при вводе значений характеристик выполненных проектов; 2. Подсказки при задании значений характеристик исследуемого проекта.
6	Значение характеристики по умолчанию	Текст/число	Используется для сокращения трудоемкости ввода значений и снижения вероятности ошибок при вводе

№	Наименование атрибута	Возможные значения	Примечания
7	Документы, в которых характеристика может находиться	Текст. Некоторые или все, кроме 1и 2, возможные значения: «эксперт» (по умолчанию), корпоративная база данных, документ Simplified Statement of Work (SSOW), документ Statement of Work (SOW), документ Project Plan (PP), документ Project Closure Report (CR).	Используется в качестве подсказки при поиске источников, из которых может быть взято значение характеристики
8	Уникальный номер характеристики-родителя	Число	Используется для структуризации характеристик, чтобы обеспечить уменьшение пространства поиска
9	Уникальные номера характеристик-потомков	Число	Используется для структуризации характеристик, чтобы обеспечить уменьшение пространства поиска
10	Формула, по которой характеристика вычисляется (внутреннее и внешнее представление)	Текст	Не определена. Аналитическое выражение (для вычисляемых характеристик). Формируется путём редактирования «исходной» формулы, включающей уникальные номера соответствующих характеристик: «= un_1+ un_2; …+ un_n»
11	Множество, которому принадлежит характеристика	Число. Возможные значения: исходное множество (ИМ), уточненное множество (УМ), рекомендуемое множество (РМ)	Аналогично множествам источников
12	Используемость в проектах	Число	Количество проектов, содержащих значение данной характеристики. Формируется автоматически
13	Дата последнего изменения	Дата	Формируется автоматически
14	Автор изменения	Текст	Формируется автоматически

Для уменьшения пространства поиска предлагается структуризация исходного множества характеристик, для чего всё множество характеристик разбивается на три подмножества: «категории», «интегральные характеристики» и «терминальные характеристики».

Характеристики-категории объединяют характеристики (интегральные и терминальные) по таким исходным понятиям как продукт, команда разработчиков, качество и т.п. (в терминах родитель-потомок имеют исключительно потомков).

Интегральные характеристики, как и категории, используются для описания некоторого подмножества характеристик (интегральных и терминальных), однако сами при этом входят в понятия более высокого уровня. В терминах родитель-потомок имеют как родителей, так и потомков.

Терминальные характеристики не описывают ничего, кроме самих себя. В терминах родитель-потомок имеют исключительно родителей.

Множество возможных категорий характеристик представлено в табл. 2.3.

Таблица 2.3. Множество категорий характеристик

№	Наименование категории	Описание категории и примеры характеристик
1	General characteristics (общие характеристики)	Описывают проект в целом. Примеры: наименование проекта, предметная область, модель ЖЦ, даты начала и завершения, цели, методология и пр.
2	Business characteristics (бизнес характеристики)	Описывают проект с точки зрения заказчика. Примеры: ограничения заказчика по бюджету, качеству, вехам и пр.
3	Product characteristics (характеристики продукта)	Метрики, описывающие разрабатываемый продукт. Примеры: объем продукта (LOC), стоимость дефектов, пакет поставки и пр.
4	Team characteristics (характеристики команды)	Описывают команду, работающую над продуктом. Примеры: число членов команды, опыт членов команды, достаточность ресурсов, опыт совместной работы и пр.

№	Наименование категории	Описание категории и примеры характеристик
5	Project portfolio (портфель проекта)	Описывают технологии, используемые в проекте. Примеры: базы данных, операционные системы, технологии, инструменты тестирования и пр.
6	Project process characteristics(характеристики процесса проекта)	Описывают характеристики, предназначенные для описания процесса проекта. Примеры: интеграция команды с заказчиком, ограничения по безопасности, скорость обратной связи с заказчиком и пр.
7	Project efforts characteristics (характеристики трудоемкости проекта)	Описают трудоемкость разработки проекта. Примеры: трудоёмкость анализа требования, разработки, тестирования, выполнения документации, управления и пр.
8	Quality characteristics (характеристики качества)	Описывают характеристики, управляющие качеством проекта. Примеры: уровень использования практик обеспечения качества, число отчетов о дефектах, плотность дефектов и пр.
9	Service characteristics (служебные характеристики)	Описывают дополнительные характеристики проекта, необходимые для функционирования системы: ссылки на местоположение документации проекта (метрики, исходный код, отчеты и пр.)
10	Risks characteristics (характеристики рисков)	Описывают риски проекта. Примеры: описание риска, вероятность возникновения, влияние, меры по предотвращению возникновения риска и пр.

Предполагается, что всё множество характеристик можно описать с помощью трёх уровней рассмотрения: 1-й уровень – категории, 2-ой уровень – потомки категорий (интегральные и терминальные характеристики) и 3-ий уровень – терминальные характеристики.

На рисунке 2.3 представлен фрагмент характеристик одной из категорий. Жирным шрифтом выделены категории и интегральные характеристики.

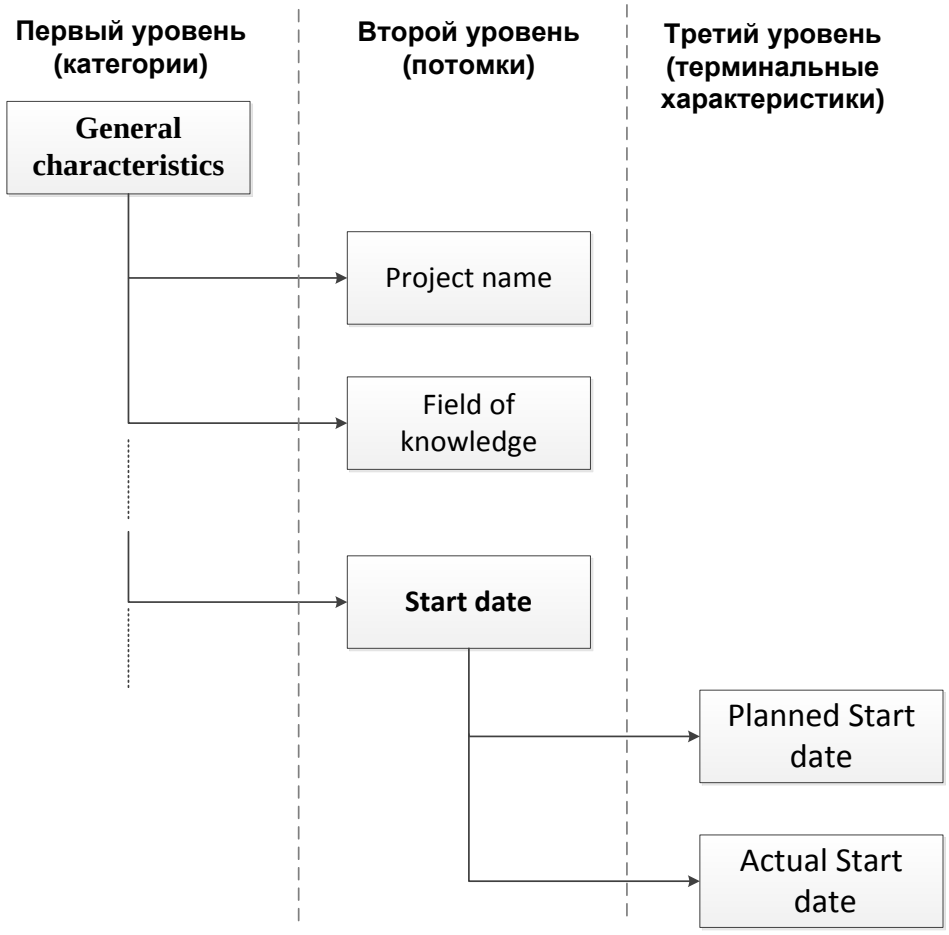


Рисунок 2.3. Фрагмент характеристик категории «General characteristics»

2.1.3 Проекты

Формирование исходного множества проектов происходит в 3 этапа.

На первом этапе формируется *исходное* множество проектов, выполнявшихся в компании, по которым имеется какая-либо информация.

На втором этапе для каждого проекта из исходного множества происходит определение значения атрибутов, описывающих проект, на основании чего и происходит окончательное его формирование (*проекты, которые могут быть описаны в атрибутах проектов исходного множества*). Атрибуты проектов исходного множества, представлены в табл. 2.4.

Таблица 2.4. Атрибуты проекта

№	Наименование атрибута	Возможные значения	Примечания
	Уникальный номер проекта	Число	Используется для однозначной идентификации характеристики

№	Наименование атрибута	Возможные значения	Примечания
1	Наименование проекта	Текст	Символьная строка, определяющая характеристику
2	Содержание проекта	Текст	Используется в качестве подсказки: раскрывает смысловое содержание проекта
3	Перечень источников, из которых могут быть взяты значения характеристик рассматриваемого проекта	Текст. Некоторые или все, кроме 1и 2, возможные значения: «эксперт» (по умолчанию), корпоративная база данных, документ Simplified Statement of Work (SSOW), документ Statement of Work (SOW), документ Project Plan (PP), документ Project Closure Report (CR).	Используется в качестве подсказки при поиске источников, из которых может быть взято значение характеристики
4	Местоположение источников, из которых могут быть взяты значения характеристик	Текст. Пути к документам, которые могут содержать значения характеристик проекта	Связывают идентификатор документа источника с его конкретным местоположением
5	Значение характеристик проекта	Одно из возможных значений: не определено (в начальный момент), не используется (в данном проекте); неизвестно (в данном проекте), число, текст, дата, метка шкалы.	Каждая характеристика проекта описывается ссылкой на источник и значением
6	Источники введенных значений характеристики проекта	Текст. Ссылка на источник, из которого взято значение характеристики проекта, и местоположение значения в документе (указатель).	Позволяют быстро определить местонахождение значения характеристики в документе в случае возможной смены шаблона
7	Множество, которому принадлежит проект	Число. Возможные значения: исходное множество (ИМ), уточненное множество (УМ), рекомендуемое множество (РМ)	Аналогично множествам источников и характеристик
8	Обеспеченность информацией	Число	Количество характеристик проекта, обеспеченных значениями. Формируется автоматически

№	Наименование атрибута	Возможные значения	Примечания
9	Дата последнего изменения	Дата	Формируется автоматически
10	Автор изменения	Текст	Формируется автоматически

На третьем этапе осуществляется ввод значений характеристик сформированного (исходного) множества проектов. Возможны два сценария ввода:

- ввод значений характеристик для выбранного проекта (проект → характеристики);
- ввод значений выбранной характеристики для сформированного множества проектов (характеристика → проекты).

Результатом этапа является *база выполненных проектов на основе исходного множества характеристик*.

2.2 Формирование уточнённых множеств

После того, как сформированы исходные множества источников, характеристик и проектов, то есть, собрана вся существующая и доступная информация, необходимо провести количественный анализ данных множеств, с целью выявления элементов не обеспеченных информацией, иными словами, тех, значениях по которым отсутствуют.

2.2.1 Характеристики

На данном этапе происходит уточнение исходного множества, исходя из обеспеченности информацией значений входящих в него характеристик, и их важности (атрибут 12, табл. 2.2 – «*Используемость в проектах*»). Характеристики, информация о значениях которых недостоверна либо отсутствует, должны быть оставлены в исходном множестве. Если, по мнению экспертов, характеристика всё же важна для решаемой задачи, то она переводиться в уточненное множество с отметкой о необходимости сбора её значений в дальнейшем.

Результатом данного этапа является уточненное множество характеристик, значения каждой из которых обеспечены информацией, либо должны быть ею обеспечены в ходе дальнейшей работы над БВП.

2.2.2 Проекты

После уточнения множества характеристик, уточняется множество проектов. Уточнение происходит, исходя из степени их обеспеченности информацией (атрибут 8, табл. 2.4– «*Обеспеченность информацией»*). Проекты, имеющее недостаточное количество заполненных значений характеристик (менее 30%), должны быть отставлены в исходном множестве.

Если, по мнению экспертов, проект всё же важен для решаемой задачи, то он переводиться в уточненное множество с отметкой о необходимости сбора всех недостающих характеристик.

Результатом данного этапа является уточненное множество проектов, значения характеристик которых обеспечены информацией, либо должны быть обеспечены в ходе дальнейшей работы над БВП.

2.2.3 Источники

После уточнения множества характеристик и проектов, уточняется множество источников. Уточнение происходит, исходя из степени их использования (атрибуты 5 – *«Используемость в характеристиках»* и 6 – *Используемость в проектах»*). Не используемые источники остаются в исходном множестве источников.

Если, по мнению экспертов, источник всё же важен для решаемой задачи, то он переводиться в уточненное множество с отметкой о необходимости сбора данных по этому источнику.

Результатом данного этапа является уточненное множество источников, из которых должна извлекаться информация для, подтверждающих достаточность уточненного множества характеристик.

Окончательным результатом этапов формирования уточненных множеств характеристик, проектов и источников является *база проектов на основе уточненного множества характеристик*.

2.3 Формирование рекомендуемых множеств

После того, как сформированы уточненные множества источников, характеристик и проектов, необходимо провести качественный анализ данных множеств. Качественный анализ подразумевает попытку классификации характеристик на ансамбли, а проектов на классы внутри этих ансамблей.

2.3.1 Характеристики

Для формирования рекомендуемого множества характеристик, необходимо провести его качественный анализ, который происходит в 3 этапа:

1) *классификация относительно выполненных проектов*. Рекомендуемое множество характеристик считается сформированными относительно выполненных проектов, если классификация проектов *на всех* предметно значимых поисковых наборах согласуется с мнением экспертов. В случае выявленного несогласия предпринимается расширение состава характеристик базы и повторное исследование на полноту.

   В рамках разработанной модели данная процедура называется «проверка функциональной пригодности пространства характеристик относительно внесенных проектов» и подробно рассматривается в разделе 3.2.

2) *классификация относительно добавляемых проектов.* Рекомендуемое множество характеристик считается сформированными относительно добавляемых проектов, если исследуемый проект может быть вписан в существующие классификаторы. Признаком того, что проект может быть вписан в классификатор, является наличие проекта-аналога, уже существующего в базе. Добавление проекта происходит либо путём включения в существующий класс, либо путём создания нового класса в

существующем классификаторе. В случае если проект не может быть вписан, предпринимается расширение состава характеристик базы и повторное исследование ее на полноту.

В рамках разработанной модели данная процедура называется «проверка функциональной пригодности пространства характеристик относительно добавляемых проектов» и подробно рассматривается в разделе 3.3.

3) *классификация относительно инициируемых проектов.* Рекомендуемое множество характеристик считается сформированными относительно инициируемых проектов, если инициируемому проекту может быть сопоставлен проект-аналог, выполнявшийся ранее. Если для какого-либо проекта проект-аналог не находится, то происходит расширение состава характеристик базы и повторное исследование на полноту.

В рамках разработанной модели данная процедура называется «проверка функциональной пригодности пространства характеристик относительно инициируемых проектов» и подробно рассматривается в разделе 3.4.

После проведения трех этапов классификации рекомендуемое множество характеристик можно считать полностью сформированным, а не попавшие в ансамбли характеристики остаются в уточненном множестве.

2.3.2 Проекты

Формирование рекомендуемого множества проектов заключается в переводе из уточненного тех проектов, которые прошли классификацию.

2.3.3 Источники

Формирование рекомендуемого множества источников заключается в переводе из уточненного тех источников, в которых находится информация о характеристиках рекомендуемого множества.

2.4 Выводы

Описанная в главе модель предназначена для формирования пространства характеристик с последующей оценкой необходимых программному проекту ресурсов. По ходу работы с моделью происходит преобразование исходной базы данных на основе количественных и качественных оценок.

На первом этапе работы формируются исходные множества источников, характеристик и проектов. Исходные множества формируются на основании анализа процесса организации и включают максимум доступной информации.

На втором этапе происходит формирование уточненных множеств источников, характеристик и проектов Уточнение множества формируются на основе анализа обеспеченности каждого элемента информацией. Если, по мнению эксперта, объект обладает полнотой информации, то объект переносится их исходного в уточненное множество.

На третьем этапе происходит формирование рекомендуемых множеств источников, характеристик и проектов. Рекомендуемые множества формируются на основании испытаний, подтверждающих достаточность исследуемого множества характеристик для решения задачи поиска проекта-аналога.

В результате организация получает БВП на основе рекомендуемого множества характеристик, информация в которой является функционально пригодной для решения задачи поиска проекта-аналога.

Глава 3 Процедуры проверки функциональной пригодности пространства характеристик

3.1 Функциональная пригодность информации в базах данных

Согласно [72, 73], для анализа свойств БД предлагается использовать *характеристики качества* СУБД и содержащейся в ней *информации*. Их перечень, описание и подхарактеристики опираются на стандарт ISO 9126 [71]. Среди главных характеристик качества можно выделить: функциональную пригодность информации базы данных, достоверность и корректность данных, надежность информации, защищенность информации, используемость ресурсов и т.д.

В ходе формирования рекомендуемого множества характеристик необходимо максимально увеличить функциональную пригодность БВП, а также полноту и непротиворечивость хранящихся в ней данных.

Функциональная пригодность базы данных – это уровень достижения целей, функций и назначения баз данных информацией, доступной пользователям. Рассмотрим свойства, которые определяют функциональную пригодность информационной базы данных [72]:

- полнота накопленных описаний объектов – относительное число объектов или документов, имеющихся в БД, к общему числу объектов по данной тематике или по отношению к числу объектов в доступных аналогичных базах данных, которые находятся в открытом доступе (в рассматриваемом случае характеристика отражает способность базы отыскивать аналоги для инициируемых проектов);
- достоверность (корректность) данных – это экспертная оценка степени соответствия описания данных об объектах в БД реальным объектам (в рассматриваемом случае данное свойство отражает точку зрения эксперта на достоверность, хранящейся в БВП информации);

- идентичность – способность системы однозначно идентифицировать объект в БД (в рассматриваемом случае свойство показывает, что в БВП нет дублирующих по смыслу характеристик).

Формирование рекомендуемого множества характеристик в общем случае предлагается рассматривать как последовательность следующих этапов исследования:

1) проверка функциональной пригодности информации относительно *внесённых* в базу выполненных проектов (то есть тех, которые уже находятся в БВП);
2) проверка функциональной пригодности информации относительно *добавляемых* в базу выполненных проектов (то есть тех, которые по завершении вносятся в БВП);
3) проверка функциональной пригодности информации относительно *инициируемых* проектов.

На рисунке 3.1 представлен обобщенный алгоритм формирования рекомендуемого множества характеристик.

Решение задачи формирования рекомендуемого множества характеристик предполагается осуществить путём сведения её к задачам кластеризации и классификации[1], методами, основывающимися на привлечении профессиональных знаний экспертов. В качестве основного критерия успешности решения названных задач считается получение результатов имеющих, по мнению экспертов, приемлемую предметную интерпретацию и охватывающих всю исследуемую область.

Особенности подхода: возможность компенсировать неопределенный характер исходной информации за счет использования методов, позволяющих

[1] *Кластеризация* – «естественное» разбиения на классы, свободное от субъективизма исследователя, выделение групп однородных объектов, сходных между собой, при резком отличии этих групп друг от друга

Классификация – отнесение некоторого объекта какой-либо предметной области, к какому либо классу. В общем случае: процедура построения классификации, построенная классификацию и процедура ее использования

привлекать в процесс выработки окончательного решения, плохо формализуемые знания эксперта.

Рисунок 3.1. Обобщенный алгоритм формирования пространства характеристик

Особенность реализуемых процедур: процедуры представляет собой последовательность шагов, часть которых реализует формальные методы, обеспечивающие информационную поддержку лица, принимающего решение, а часть – поддержку решений, принимаемых экспертом на основе полученной информации.

Особенности используемых формальных методов:

- формирование в качестве конечного результата работы формальных алгоритмов (методов) некоторого множества решений и дополнительной информации, обеспечивающей вовлечение в процесс принятия решения плохо формализуемых знаний эксперта;
- возможность интерпретации процедур, обеспечивающих получение конечного и промежуточных результатов в тезаурусе предметных знаний ЛПР с целью недопущения некорректного использования применяемого математического аппарата;
- применение операций по переработке информации, предлагаемых ЛПР, исключительно с учетом их допустимой сложности.

3.2 Проверка функциональной пригодности

3.2.1 Внесенные проекты

Тезис: информация, хранимая в базе выполненных проектов, является функционально пригодной относительно выполненных проектов, если классификация проектов, полученная после применения процедур кластеризации *на всех* предметно значимых поисковых наборах, согласуется с мнением экспертов. В случае выявленного несогласия предпринимается расширение состава характеристик базы и повторное исследование на полноту.

Предметно значимый поисковый набор (ансамбль, классификатор)[2] – множество характеристик, используемое на практике для идентификации проектов или их группы.

Исходное опорное множество характеристик – множество характеристик, полученное путём объединения характеристик всех предметно значимых наборов. Очевидно, что множество предметно значимых наборов является подмножеством

[2]*Классификатор* — систематизированный перечень наименований объектов, каждому из которых в соответствие дан уникальный код

множества всех теоретически возможных наборов характеристик исходного множества.

Таким образом, проверку функциональной пригодности информации относительно выполненных проектов (полноту накопленных описаний и идентичность) предлагается свести к задаче проверки полноты предметно значимых наборов характеристик, *формируемых на основе исследуемой базы*, а также корректности соответствующих им классификаций.

Как известно, в кластерном анализе существенной проблемой является *выбор необходимого числа кластеров (классов)*. В некоторых случаях число классов может быть выбрано из априорных соображений, однако чаще это число определяется в процессе формирования классов на основе значений некоторых показателей их однородности и степени удаленности друг от друга (например, показателей внутригрупповой дисперсии или вариации).

Очевидно, что множество ансамблей характеристик будет определять множество возможных классификаций, тогда как множество имеющихся сочетаний значений характеристик определит множество различных классов внутри классификации. Тогда *минимально возможное число классов* для исследуемого ансамбля характеристик будет определяться как максимальное значение из множества возможных значений характеристик, входящих в данный ансамбль. Например, пусть имеется набор из трёх характеристик, при этом первая характеристика имеет 3 значения, вторая –2 и третья – 4. Минимально возможное число классов в этом случае будет равно 4 = *max (3, 2, 4)*.

Максимально возможное число классов – число исследуемых проектов.

В таблице 3.1 даны примеры возможных классификаций выполненных проектов. В данной таблице можно выделить 3 классификатора: на 3 класса (характеристики x2 и x5), классификатор на 4 класса (характеристики x3, x4, x6 и x7) и классификатор на 5 классов (характеристики с x2 по x7).

Таблица 3.1. Пример возможных классификаций проектов

Проекты	**Характеристики (исходный ансамбль)**							**№ класса**		
	x_1	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**(x_2, x_5)**	**(x_3,x_4, x_6, x_7)**	**(x_2-x_7)**
проект «А»		0	0	0	0	0	0	1	1	1
проект «Б»		1	1	1	1	1	1	2	2	2
проект «В»		1	1	1	1	1	1	2	2	2
проект «Г»		0	0	0	0	0	0	1	1	1
проект «Д»		1	1	1	1	1	1	2	2	2
проект «Е»		2	3	3	2	3	3	3	3	3
проект «Ж»		0	3	3	0	3	3	1	3	4
проект «З»		0	2	0	0	5	2	1	4	5
Кол-во		**3**	**4**	**3**	**3**	**4**	**4**	**3**	**4**	**5**

В рамках подхода, ориентированного на привлечение предметных знаний эксперта, предметно значимые наборы (ансамбли), обеспечивающие *минимальное возможное число классов* (прежде всего, конечно, максимальные) предлагается рассматривать как исходные в процессе окончательной кластеризации исследуемых проектов. При этом исключительно прерогативой эксперта является:

- *включение новых характеристик* в исследуемый ансамбль (система может рекомендовать характеристики, изменение значений которых ведет к появлению минимального числа дополнительных классов, превышающих число минимально возможное);
- *коррекция значений добавляемой характеристики* (система может указать проекты, для которых подобная коррекция будет минимальной);
- определение правильности полученной классификации.

В таблице 3.2 представлена переменные алгоритмическая сеть описывающей проверку функциональной пригодности относительно внесенных проектов, на рисунке 3.2 представлена сама сеть.

Таблица 3.2. Переменные алгоритмической сети. Проверка относительно внесенных проектов

Наименование	**Описание**
x_{1R1}	множество характеристик, предложенное экспертом
x_{2R1}	множество характеристик, полученное из метрической БД организации
x_{3R1}	множеств характеристик, полученное из документов СППО

Наименование	Описание
x_{4R1}	универсальное множество характеристик (метрики, характеризующих программный проект, создаваемый продукт и процесс разработки)
x_{1R2}	множество проектов, предложенное экспертом
x_{2R2}	множество проектов, полученное из метрической БД организации
R_1, R_2	исходное множество характеристик и проектов соответственно
CR_1, CR_2	количество элементов во множестве характеристик и проектов соответственно
T_1, T_2, T_3	счетчики цикла
x_1, x_4	анализируемая характеристика и проект соответственно
x_3, x_6	обеспеченная информацией характеристика и проект соответственно
x_2, x_5	не обеспеченная информацией характеристика и проект соответственно
R_3, R_4	уточненное множество источников, характеристик и проектов соответственно
Подсеть 1	процедура формального создания классификаторов
R_5	множество сформированных классификаторов
CR_3	количество сформированных классификаторов
x_7	анализируемый классификатор
x_9	классификатор, подходящий для решения задачи
x_8	классификатор, не подходящий для решения задачи
x_{10}	классификаторы не сформированы, требуется повтор процедуры экспертного создания классификаторов
x_{11}	классификаторы не сформированы, требуется расширение состава уточненных характеристик
R_6	исходное опорное множество характеристик (объединение всех характеристик, которые попали в классификаторы)

В качестве модельных переменных (x_{1R1}, x_{2R1} и пр.) используются множества, а в качестве функциональных отношений (операторов) используются операции над множествами.

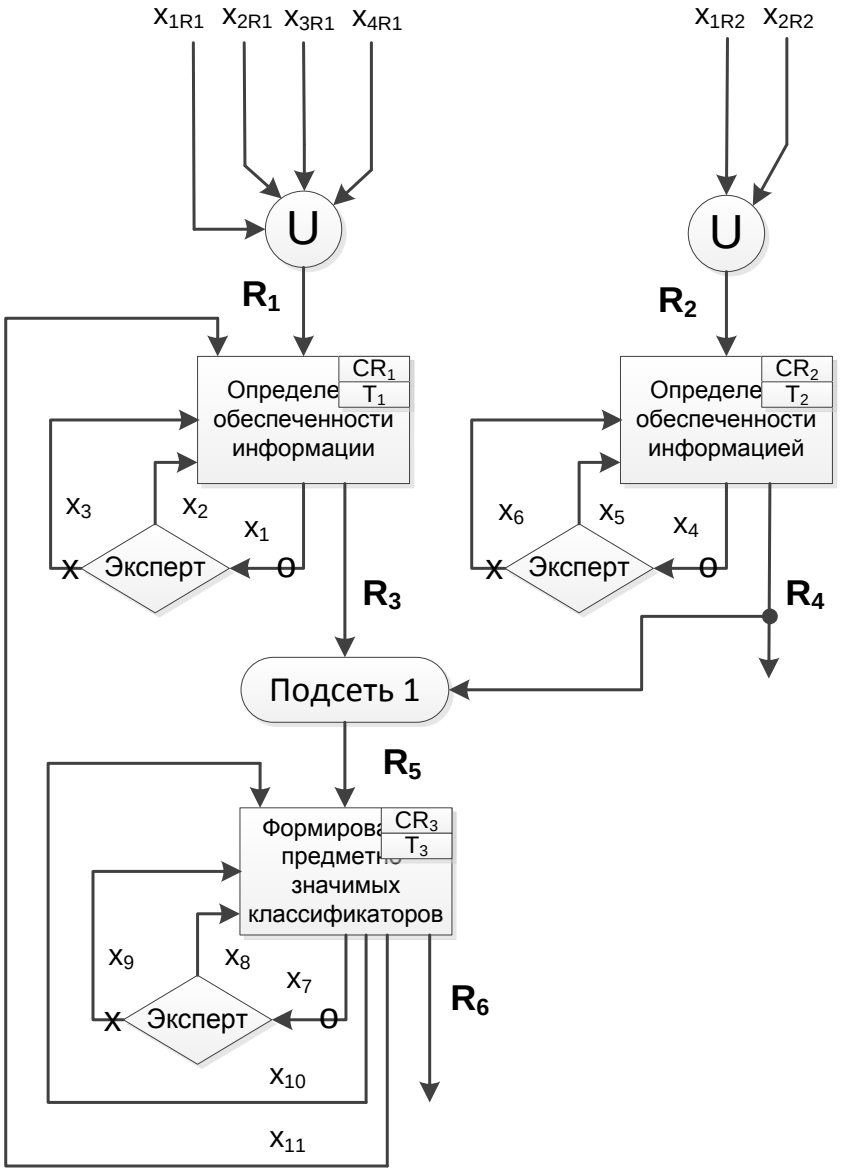


Рисунок 3.2. Проверка функциональной пригодности информации относительно внесенных проектов в формализме алгоритмических сетей

Процедура проверка относительно **внесенных** проектов:

1) формируется множество «минимальных» классификаторов (алгоритм формального создания классификаторов).
2) выявляются кандидаты в предметно значимые классификаторы (экспертная оценка возможности предметной интерпретации того или иного элемента из ранее полученного множества классификаторов и правильность осуществляемой им классификации).
3) формируется множество предметно значимых классификаторов (алгоритм экспертного создания классификаторов).

4) определяется полнота полученных классификаторов (анализ мнений экспертов о достаточности полученного множества для решения задач, стоящих перед формируемой базой и, прежде всего, задачи поиска проектов-аналогов).
5) в зависимости от полученных результатов реализуются различные завершающие этапы:
   - если множество предметных классификаторов, по мнению экспертов, полно, формируется *исходное опорное множество характеристик* путем объединения множеств всех предметно значимых наборов (успешное завершение проверки полноты);
   - если множество предметных классификаторов, по мнению экспертов, не полно, осуществляется попытка его расширения в рамках уточненного множества характеристик с повторным анализом полноты (попытка решить проблему полноты в рамках исследуемого множества характеристик);
   - если множество предметных классификаторов, по мнению экспертов, не полно и попытка его расширения в рамках уточненного множества характеристик не привела к желаемому результату, осуществляется расширение уточненного множества характеристик, с последующим сбором информации о значениях и повторным исследованием функциональной пригодности информации (необходимость решить проблему полноты за счёт расширения исследуемого множества характеристик).

Алгоритм формального создания классификаторов

Алгоритм призван сформировать множество классификаторов, обусловленных содержанием базы проектов, исходя из *формального критерия* минимальности содержащихся в них классов.

Ниже представлен алгоритм формального создания классификаторов на псевдокоде:

```
# ch - характеристика
# noAnsCh[] – множество характеристик, не принадлежащих ни одному ансамблю
# ansCh[] – формируемый ансамбль характеристик
# cValues – число уникальных значений характеристик в исследуемом множестве проектов
# cClasses – число классов, получаемых при добавлении харакатеристики в ансамбль
# cMinClass – число минимальных возможных классов
begin
 ansCh ← ∅
 for all ch ϵ noAnsCh do
  if ch <> ∅ then
   if noAnsCh == ∅ then
    begin
     Добавить ch в ansCh; Исключить ch из noAnsCh;
     ОпределитьcValues; cMinClass = cValues;
    end
   else
    begin
     Определить cValues; Определить cClasses;
     if((cMinClass>cClasses) || (cMinClass>cValues)) then do
      begin
       Добавить ch в ansCh; Исключить ch из noAnsCh;
      end
      if cValues > cMinClass then do cMinClass = cValues;
    end
  else
   begin
    Запоминание ansCh;
    ansCh ← ∅
   end
end
```

При определении количества классов, получаемых с помощь формируемого ансамбля (оператор «*Определить cClasses;*») необходимо воспользоваться понятием о сходстве объектов. Как известно о сходстве объектов можно судить по расстоянию между соответствующими точками. Содержательный смысл такого понимания сходства означает, что объекты тем более близки, похожи в рассматриваемом аспекте, чем меньше различий между значениями одноименных показателей. В рассматриваемом случае в качестве критерия принадлежности к одному классу принимается *полное совпадение значений рассматриваемых характеристик*.

Алгоритм экспертного создания классификаторов

Алгоритм призван максимально расширить полученное формальным путем множество классификаторов базы, в том числе за счет коррекции экспертом значений характеристик проектов базы, а также отказа от критерия минимальности количества классов за счет признания экспертом права на расширение их числа.

Ниже представлен алгоритм экспертного создания классификаторов на псевдокоде:

```
    # ans - ансамбль
    # cAns[] – множество ансамблей (ans ϵ cAns)
    # ch– характеристикадля добавления в ансамбль
    # cCh[] – множество характеристик для добавления в ансамбль
(ch ϵ cCh)
    # cMinClass – число минимальных возможных классов
    begin
     for all ans ϵ cAns do
      if ans <> Ø then
       Определить cMinClass;
        for all ch ϵ cCh do
         if ch <> Ø then
          begin
           Классификация проектов согласно сформированному ансамблю;
```

```
            if классификация удовлетворяет требованиям эксперта then
             begin
              Внесение ch в ans; Определение cMinClass;
             end
            else
             if ch є другим ансамблям then
              begin
               if допустимо ли изменение значения ch в других ансамблях then
                if возможно ли изменение значения ch, удовлетворяющее условью неизменности количества классов then
                 Внесение изменение в БВП;
              end
             else
              if возможно ли изменение значения ch, удовлетворяющее условью неизменности количества классов then Внесение изменение в БВП;
          end
    end
```

3.2.2 Добавляемые проекты

Тезис: информация, хранимая в базе выполненных проектов, является функционально пригодной относительно добавляемого проекта, если исследуемый проект может быть вписан в существующие классификаторы. Признаком того, что проект может быть вписан в классификатор, является наличие проекта-аналога, уже существующего в базе. Добавление проекта происходит либо путём включения в существующий класс, либо путём создания нового класса в существующем классификаторе. В случае если проект не может быть вписан, предпринимается расширение состава характеристик базы и повторное исследование ее на полноту.

В рамках подхода, ориентированного на привлечение предметных знаний эксперта, с помощью алгоритма, основанного на вычислении оценок (АВО) [74], предложенного Ю.И. Журавлевым в 70-х годах и модифицированного для решения поставленной задачи поиска проектов-аналогов, формируется множество проектов (рекомендуемое множество), которые представляются эксперту в

качестве возможных аналогов исследуемому проекту. При этом исключительно прерогативой эксперта является:

- *выбор проекта-аналога* исследуемому проекту (система формирует рекомендуемое множество проектов, а также сравнительную информацию о не совпавших значениях характеристик, входящих в классифицирующий ансамбль);
- *уточнение значений* характеристик исследуемого проекта;
- *определение корректности проведенной классификации* в рамках выбранного классификатора с точки зрения эксперта (система предоставляет информацию обо всех не совпавших значениях характеристик);
- *подтверждение возможности расширения* выбранного классификатора путём включения в него нового класса (система предоставляет информацию обо всех не совпавших значениях характеристик).

В таблице 3.3 представлена переменные алгоритмическая сеть описывающей проверку функциональной пригодности относительно добавляемых проектов, на рисунке 3.3 представлена сама сеть.

Таблица 3.3. Переменные алгоритмической сети. Проверка относительно добавляемых проектов

Наименование	**Описание**
x_{1R1}	множество характеристик, предложенное экспертом
x_{2R1}	множество характеристик, полученное из метрической БД организации
x_{3R1}	множеств характеристик, полученное из документов СППО
x_{4R1}	универсальное множество характеристик (метрики, характеризующих программный проект, создаваемый продукт и процесс разработки)
x_{1R2}	множество проектов, предложенное экспертом
x_{2R2}	множество проектов, полученное из метрической БД организации
R_1, R_2	исходное множество характеристик и проектов соответственно
CR_1, CR_2	количество элементов во множестве характеристик и проектов соответственно
T_1, T_2, T_3	счетчики цикла
x_1, x_4	анализируемая характеристика и проект соответственно
x_3, x_6	обеспеченная информацией характеристика и проект соответственно
x_2, x_5	не обеспеченная информацией характеристика и проект соответственно

Наименование	Описание
R_3, R_4	уточненное множество источников, характеристик и проектов соответственно
Подсеть 1	процедура поиска проектов-аналогов
R_5	множество найденных проектов-аналогов
CR_3	количество найденных проектов-аналогов
x_7	анализируемый проект-аналог
x_9	отвергнутый проект-аналог
x_8	выбранный проект-аналог
x_{10}	проект-аналог не найден, требуется расширение уточненного множества характеристик
P_1	проект-аналог, используемый в процедуре оценки ресурсов

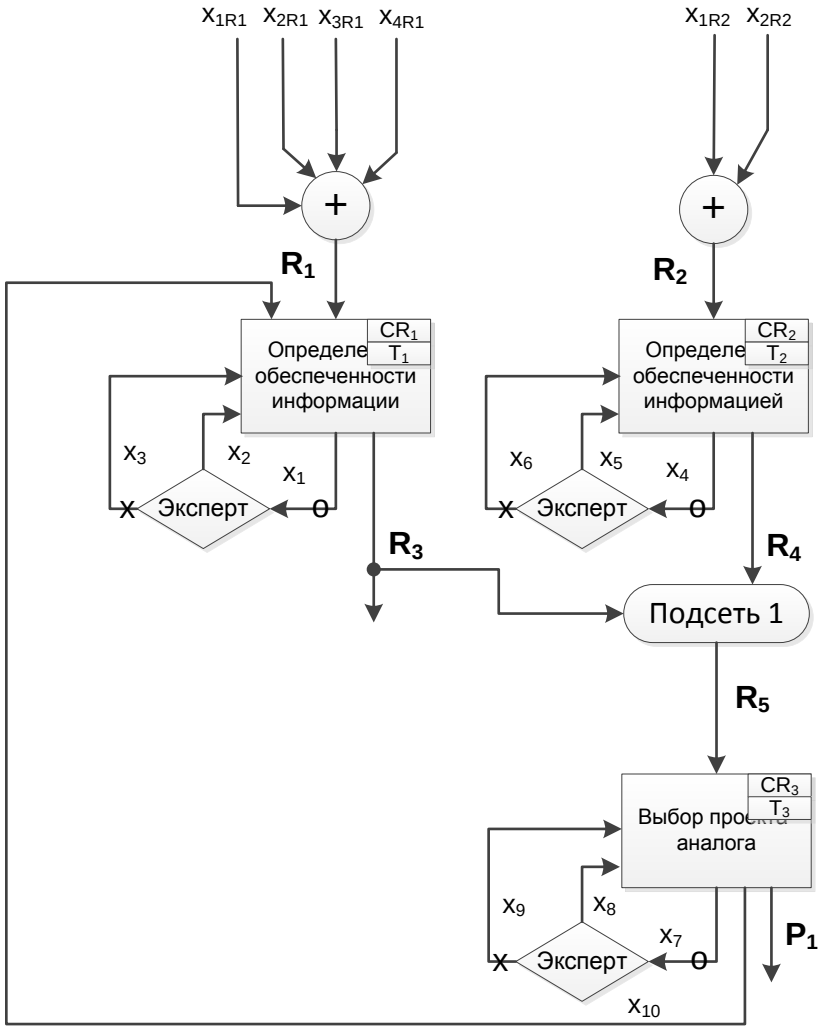


Рисунок 3.3. Проверка функциональной пригодности информации относительно добавляемых проектов в формализме алгоритмических сетей

Процедура проверка относительно **добавляемых** проектов:

1) вводятся значения *всех* характеристик добавляемого проекта (эксперт).

2) формируется множество проектов, рекомендуемых системой в качестве возможного аналога добавляемому проекту (алгоритм формирования рекомендуемого множества).
3) анализируется информация о не совпавших значениях характеристик проектов рекомендуемого множества и добавляемого проекта (эксперт).
4) выбирается проект, являющийся, по мнению эксперта, кандидатом в аналоги добавляемому (эксперт).
5) уточняются или не уточняются значения характеристик исследуемого проекта в соответствии со значениями выбранного аналога (эксперт).
6) определяется правильность отнесения добавляемого проекта к выбранному классу либо возможность расширения анализируемого классификатора на основе анализа значений прочих не совпавших характеристик (эксперт).
7) в зависимости от полученных результатов реализуются различные завершающие этапы:
    - если, мнение экспертов положительное, но просмотрены не все добавляемые проекты, переход к следующему проекту и повторение ранее описанных этапов (1 – 7);
    - если, мнение экспертов положительное и просмотрены все исследуемые проекты, успешное завершение проверки полноты;
    - если, мнение экспертов отрицательное, и попыток решения проблемы за счет расширения множества предметных классификаторов не предпринималось, осуществляется расширение множества предметных классификаторов с повторным анализом его полноты;
    - если, мнение экспертов отрицательное, и попытка решения проблемы за счет расширения множества предметных классификаторов не привела к желаемому результату, осуществляется расширение уточненного множества характеристик, сбор информации о значениях и повторное исследование

функциональной пригодности информации относительно выполненных проектов.

Алгоритм формирования рекомендуемого множества

1) определение числа полных совпадений значений для характеристик исходного опорного множества проектов базы и добавляемого проекта.
2) упорядочивание множества проектов базы по максимальному числу совпадающих значений.
3) формирование рекомендуемого множества проектов. Объем информации, обрабатываемой экспертом в ходе решения задачи, не должен превышать рекомендуемых в [68, 75] объёмов. В ходе предварительного выбора эксперт сравнивает проекты, претендующие на роль аналогов, анализируя при этом значимость имеющихся несовпадений значений характеристик внутри каждого проекта. Отсюда в рекомендуемое множество включаются проекты с числом не совпавших характеристик не более 5 [68, 75]. Общее количество анализируемых проектов также не должно превышать это число.

   Если количество проектов предлагаемых эксперту для анализа более 5, то система предлагает эксперту увеличить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики). Если количество не совпавших значений более 5, то система предлагает эксперту сократить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики).

3.2.3 Инициируемые проекты

Тезис: информация, хранимая в базе выполненных проектов, является функционально пригодной относительно инициируемого проекта, если инициируемому проекту может быть сопоставлен проект-аналог, выполнявшийся ранее. Если для какого-либо проекта проект-аналог не находится, то происходит расширение состава характеристик базы и повторное исследование на полноту.

В рамках подхода, ориентированного на привлечение предметных знаний эксперта, с помощью алгоритма, основанного на вычислении оценок (ABO) [74], формируется множество проектов (рекомендуемое множество), которые представляются эксперту в качестве возможных аналогов исследуемому проекту. При этом исключительно прерогативой эксперта является:

- *определение множества характеристики, описывающих инициируемый проект, и их значений*;
- *выбор проекта аналога* инициируемому проекту (система формирует рекомендуемое множество проектов, а также сравнительную информацию о не совпавших значениях характеристик, входящих в классифицирующий ансамбль);
- *уточнение значений* характеристик инициируемого проекта.

В таблице 3.4 представлена переменные алгоритмическая сеть описывающей проверку функциональной пригодности относительно инициируемых проектов, на рисунке 3.4 представлена сама сеть.

Таблица 3.4. Переменные алгоритмической сети. Проверка относительно инициируемых проектов

Наименование	**Описание**
x_{1R1}	множество характеристик, предложенное экспертом
x_{2R1}	множество характеристик, полученное из метрической БД организации
x_{3R1}	множеств характеристик, полученное из документов СППО
x_{4R1}	универсальное множество характеристик (метрики, характеризующих программный проект, создаваемый продукт и процесс разработки)
R_1, R_2	исходное множество характеристик и проектов соответственно
CR_1, CR_2	количество элементов во множестве характеристик и проектов соответственно
T_1, T_2, T_3, T_4	счетчики цикла
x_1, x_4	анализируемая характеристика и проект соответственно
x_3, x_6	обеспеченная информацией характеристика и проект соответственно
x_2, x_5	не обеспеченная информацией характеристика и проект соответственно
R_3, R_4	уточненное множество источников, характеристик и проектов соответственно
CR_3	количество элементов в уточненном множестве характеристик
x_7	анализируемая характеристика уточненного множества

Наименование	Описание
x_9	характеристика, выбранная экспертом, для описания проекта
x_8	характеристика, не выбранная экспертом, для описания проекта
R_5	множество характеристик описывающих проект, выбранных экспертом
Подсеть 1	процедура поиска проектов-аналогов
R_6	множество проектов-аналогов
CR_4	количество элементов множества проектов-аналогов
x_{10}	анализируемый проект
x_{12}	проект, являющийся, по мнению эксперта, аналогом
x_{11}	проект, не являющийся, по мнению эксперта, аналогом
x_{13}	проект-аналог не найден, требуется расширение состава характеристик, описывающих проект
x_{14}	проект-аналог не найден, требуется расширение состава уточненных характеристик
P_1	проект-аналог, выбранный экспертом

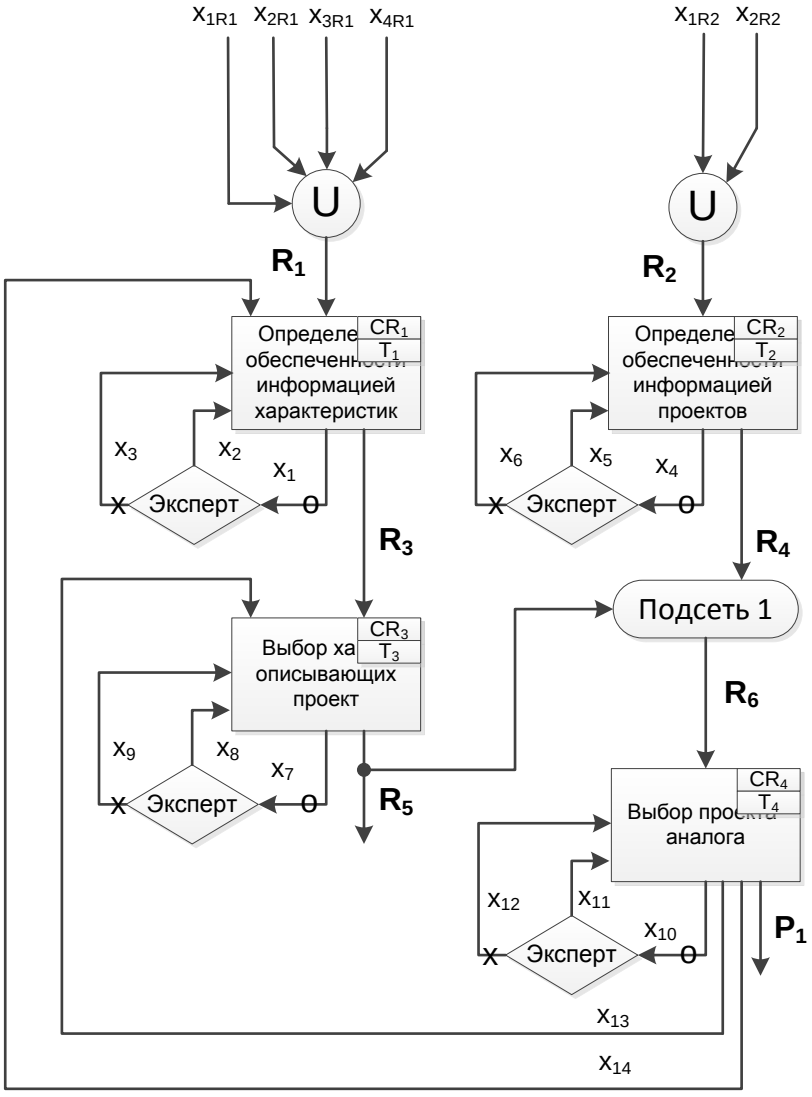


Рисунок 3.4. Проверка функциональной пригодности информации относительно добавляемых проектов в формализме алгоритмических сетей

Процедура проверка относительно **инициируемых** проектов:

1) определяется множество характеристик, описывающих инициируемый проект и их значения – начальное опорное множество (эксперт).
2) формируется множество проектов рекомендуемых системой в качестве возможного аналога добавляемому проекту (расширенный алгоритм формирования рекомендуемого множества).
3) анализируется информация о не совпавших значениях характеристик проектов рекомендуемого множества и добавляемого проекта (эксперт).
4) выбирается проект, являющийся, по мнению эксперта, кандидатом в аналоги добавляемому (эксперт).
5) уточняются или не уточняются значения характеристик исследуемого проекта в соответствии со значениями выбранного аналога (эксперт).
6) определяется правильность отнесения добавляемого проекта к выбранному классу либо возможность расширения анализируемого классификатора на основе анализа значений прочих характеристик (эксперт).
7) в зависимости от полученных результатов реализуются различные завершающие этапы:
   - если, мнение экспертов положительное, успешное завершение проверки полноты;
   - если, мнение экспертов отрицательное, и возможно попытаться решить проблему за счет уточнения *начального опорного множества*, осуществляется уточнения *начального опорного множества* (состав/значения характеристик) с повторным поиском проекта-аналога;
   - если, мнение экспертов отрицательное, и попытка решения проблемы за счет уточнения *начального опорного множества* не привела к желаемому результату, осуществляется расширение уточненного множества характеристик, сбор информации о значениях и повторное

исследование функциональной пригодности информации относительно выполненных проектов.

Расширенный алгоритм формирования рекомендуемого множества

Опорное множество (S_i) – некоторое подмножество характеристик, которое используется для оценки степени близости проектов содержащихся с БВП и инициируемого.

Система опорных множеств (S_A) – совокупность опорных множеств, полученная в результате кластеризации: $S_A = \{S_1, S_2 \ldots, S_z\}$.

Редукция опорного множества – переход от подмножества характеристик числом k к подмножеству характеристик числом $(k–n)$, где n — *шаг редукции*.

Начальное опорное множество (S_1) – опорное множество, из которого путем редукции формируются все опорные множества, в совокупности составляющие систему опорных множеств (S_A).

Правило (функция) близости – оценивает сходство инициируемого проекта (ω'), и соответствующего ему выполненного проекта БВП (Ω_r). Возможные определения: проекты считаются похожими, если выполняется не менее чем δ неравенств вида $|\alpha_j - \beta_j| \le \varepsilon_j$, $j = 1, \ldots, k$. Здесь α_j, β_j — значение j-ой характеристики инициируемого и выполненного проектов, ε_j — порог близости характеристик, δ — порог близости сравниваемых проектов (минимальное число признаков, при совпадении которых проекты считаются похожими), k — общее число характеристик, по которым производится сравнение. Определение: полное совпадение значений всех характеристик выполненного (Ω_r) и инициируемого (ω') проектов в рамках используемого опорного множества ($\varepsilon_j = 0$, $\delta = k$).

Решающее правило — правило, на основании которого происходит соотнесение инициируемого проекта к одному из выполненных проектов БВП. Определение: $\Gamma s_A(\omega', \Omega_i) - \Gamma s_A(\omega', \Omega_j) > \eta_1, \ \forall j, j=1, \ldots, m, i \neq j; \ max\ [\Gamma s_A(\omega', \Omega_1),$

$\Gamma s_A(\omega', \Omega_2)$, ..., $\Gamma s_A(\omega', \Omega_m)]$. Здесь η_1, … — параметр АВО, $\Gamma s_A(\omega', \Omega_r)$ — оценка для проекта Ω_r по системе опорных множеств S_A.

Вычисление оценки класса по системе опорных множеств. Определение: для случая, когда система опорных множеств идентична системе всех подмножеств числом элементов k и используется принятое правило близости, в АВО существует аналитическая формула, заменяющая сложные переборные процедуры:

$$\Gamma s_A(\omega', \Omega_q) = \frac{1}{z_q - z_{q-1}} \sum_{\omega_j \in \Omega_q} \gamma_{s_i \omega_j} C^k_{p(\omega_j, \omega')}. \quad (1)$$

Здесь z_q, z_{q-1} – номер последнего проекта, представляющего текущий и предшествующий класс, соответственно; $\gamma_{s_i \omega_j}$ – коэффициент, характеризующий значимость в опорном множестве s_i проекта ω_j; $p(\omega_j, \omega')$ — число выполненных равенств значений характеристик для пары (ω_j, ω'); k — мощность рассматриваемых опорных множеств.

Определение: с учетом особенностей используемой системы опорных множеств, созданной посредством применения i шагов редукции, оценка модели Ω_q по системе опорных множеств будет:

$$\Gamma s_A(\omega', \Omega_q) = \frac{1}{z_q - z_{q-1}} \sum_{i=1}^{n} \sum_{\omega_j \in \Omega_q} C^k_{p(\omega_j, \omega')}. \quad (2)$$

Описание по шагам:

1) определение количества выполненных равенств для значений характеристик инициируемого и выполненного проектов, принадлежащих исходному опорному множеству (S_1) — число элементов множеств (k_j), начиная с которых для исследуемых пар проектов (ω_j, ω') будет выполняться правило близости.
2) сортировка проектов (Ω_q) в порядке убывания величины (k_j).

3) определение системы опорных множеств (S_A), обеспечивающей поиск проекта-аналога. S_A включает множество всех подмножеств опорных множеств.
4) определение множества рекомендуемых проектов: определяется как подмножество множества ранжированных проектов (см. п. 2).
5) вычисление по формуле 2 оценок рекомендуемых проектов по системе опорных множеств S_A.
6) определение характеристик, обеспечивших выполнение правил близости. Для каждого проекта, входящего в рекомендуемое множество, определяется множество характеристик, обеспечивших выполнение правила близости.
7) определение характеристик, не обеспечивших выполнение правил близости. Для каждого проекта, входящего в рекомендуемое множество, определяются множество характеристик, не обеспечивших выполнение правила близости, а также их значения для инициируемого и выполненного проектов.
8) формирование дополнительной информации о характеристиках выбранных проектов.
9) конец.

Результатом этапа формирования рекомендуемого множества характеристик является макет базы выполненных проектов организации, а также рекомендации по изменению стандартного производственного процесса организации в плане сбора и обработки данных о выполнявшихся ранее проектах.

Сводная информация об исследованиях, осуществляемых в ходе создания и эксплуатации БВП организации, приведена в таблице 3.5.

Таблица 3.5. Сводная информация об исследованиях БВП

Проверка относительно внесённых проектов	**Проверка относительно добавляемых проектов**	**Проверка относительно инициируемых проектов**
Попытка построить *полное множество*	Попытка вписать исследуемый проект в существующие	Попытка отыскать проект с характеристиками,

Проверка относительно внесённых проектов	**Проверка относительно добавляемых проектов**	**Проверка относительно инициируемых проектов**
классификаторов, удовлетворяющих представлениям экспертов. Если попытка неудачна, расширение исследуемого множества характеристик. Классификаторы формируются на основе формально полученных предметно значимых (эксперт) ансамблей, путём добавления задаваемых характеристик (эксперт). Критерием возможности добавления является представление эксперта о сохранение правильности полученной классификации.	классификаторы, либо подтверждение необходимости добавления нового класса в один из имеющихся классификаторов (эксперт). Если попытка неудачна, расширение исследуемого множества характеристик.	имеющими максимальную близость с *некоторым множеством* характеристик, задающих инициируемый проект. Если попытка неудачна, расширение исследуемого множества характеристик.
1. Формирование минимальных классификаторов	1. Определение исследуемого проекта – внесение проекта в базу проектов	1. Определение исследуемого проекта – задание характеристик и их значений, описывающих, по мнению эксперта, исследуемый проект.
2. Формирование предметно-значимых классификаторов за счёт включения дополнительных значимых характеристик	2. Формирование множества проектов, рекомендуемых в качестве аналога исследуемому проекту (рекомендуемое множество) – для характеристик *исходного множества* ищется некоторое множество проектов максимально близких исследуемому по числу прямых совпадений. Объем информации, обрабатываемой экспертом в ходе решения задачи, не должен превышать рекомендуемых в [75] объёмов: Если количество проектов предлагаемых эксперту для	2. Формирование множества проектов, рекомендуемых в качестве аналога исследуемому проекту (рекомендуемое множество) – для характеристик *заданного множества* ищется некоторое множество проектов максимально близких исследуемому по числу прямых совпадений. Количество проектов, включаемых во множество должно обеспечивать эксперту возможность принимать решения соответствующие поставленной задаче:

Проверка относительно внесённых проектов	**Проверка относительно добавляемых проектов**	**Проверка относительно инициируемых проектов**
	анализа более 5 система предлагает эксперту увеличить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики). Если количество не совпавших значений более 5, система предлагает эксперту сократить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики).	Если количество проектов предлагаемых эксперту для анализа более 5 система предлагает эксперту увеличить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики). Если количество не совпавших значений более 5, система предлагает эксперту сократить количество характеристик, входящих в ансамбль (рекомендуя те или иные характеристики).
2.1. *Приведение* к минимальному классификатору за счёт редактирования значений вносимой характеристики (редактируемая характеристика не должна входить в другой ансамбль). Запись нового классификатора на место старого	3. Определение проекта аналогичного исследуемому.	3. Определение проекта аналогичного исследуемому.
2.2. *Согласие с вновь полученной классификацией*, в том числе с редактирования значений вносимой характеристики (редактируемая характеристика не должна входить в другой ансамбль). Запись нового классификатора на место старого	3.1. Выбор проекта-аналога на основании анализа информации о совпавших и не совпавших значениях исходного множества, а также возможной коррекции значений характеристик исследуемого проекта.	3.1. Выбор проекта-аналога на основании анализа информации о совпавших и не совпавших значениях исходного множества, а также возможной коррекции значений характеристик исследуемого проекта.
2.3. Не согласие с вновь полученной классификацией. Переход к следующей характеристике, в случае исчерпания характеристик	3.2. Дополнительный анализ оставшихся не совпавших переменных исследуемого проекта и проекта аналога.	4. Анализ успешности поиска проекта аналогичного исследуемому

Проверка относительно внесённых проектов	**Проверка относительно добавляемых проектов**	**Проверка относительно инициируемых проектов**
переход к следующему ансамблю, в случае исчерпания ансамблей переход к анализу полноты		
3. Анализ полноты предметно-значимых ансамблей.	4. Анализ успешности поиска проекта аналогичного исследуемому	4.1. Если выбранный проект действительно является аналогом исследуемому проекту – *анализ проекта-аналога* с целью оценки ресурсов, необходимых для успешного завершения инициируемого проекта (**завершение оценки**)
3.1. Если ансамбли полны — завершение проверки (**подтверждение полноты**)	4.1. Если выбранный проект действительно является аналогом исследуемому проекту: – *проверка принадлежности* проекта одному из возможных классов, либо *включение нового класса* с последующим расширением, если требуется, исходного множества; – *переход к следующему исследуемому проекту*, если список проектов исчерпан, завершение проверки (**подтверждение полноты**)	4.2. Если выбранный проект не является аналогом исследуемому проекту и список рекомендуемого множества не исчерпан – *переход к следующему проекту рекомендованного множества*
3.2. Если ансамбли не полны, расширение состава характеристик базы и повторное исследование на полноту.	4.2. Если выбранный проект не является аналогом исследуемому проекту и список рекомендуемого множества не исчерпан – *переход к следующему проекту рекомендованного множества*	4.3. Если выбранный проект не является аналогом исследуемому проекту и список рекомендуемого множества исчерпан – *изменение состава или значений характеристик, определяющих инициируемый проект*
	4.3. Если выбранный проект не является аналогом исследуемому проекту и список рекомендуемого	

Проверка относительно внесённых проектов	Проверка относительно добавляемых проектов	Проверка относительно инициируемых проектов
	множества исчерпан — **расширение состава характеристик базы** и повторное исследование на полноту.	

3.5 Выводы

Проверка функциональной пригодности информации в БВП необходима для того, чтобы удостовериться, что БВП может выполнять свое основное назначение – служить основой для поиска проекта-аналога.

Процесс проверки функциональной пригодности состоит из трех последовательно выполняемых проверок уточненного множества характеристик. Первая проверка – возможность классификации и кластеризации уточненного множества характеристик и проектов. Вторая проверка – поиск проекта-аналога для завершённого проекта, по которому известна вся проектная информация. Третья проверка – поиск проекта-аналога для инициируемого проекта, по которому известны только некоторые прогнозные значения характеристик.

Если в результате всех трех проверок эксперт принимает положительное решение, в рекомендуемое множество характеристик попадают те характеристики, которые попали в ансамбли при кластеризации. В рекомендуемое множество проектов попадают те проекты, которые в процессе классификации попали в классы. В рекомендуемое множество источников те источники, в которых находится информация о характеристиках рекомендуемого множества.

Глава 4 Система поддержки создания баз выполненных проектов САМПО+

4.1 Описание системы

Система САМПО+ поддержки создания баз выполненных проектов компаний, разрабатывающих программное обеспечение [54-57],предназначена для снижения трудозатрат, связанных с их созданием. Она позволяет на основе полученной информации сформировать модель данных для дальнейшей разработки специализированной базы, а также проводить оценку выполнимости проектов, инициируемых в компании, для которой проводилось исследование. В этом качестве система САМПО+ служит инструментом, обеспечивающим снижения риска неудачного завершения инициируемых проектов.

Наряду с прямым назначением система САМПО+ может использоваться в качестве средства повышения уровня профессиональных знаний сотрудников компании, а также в учебном процессе, связанном с подготовкой специалистов в области управления разработкой программного обеспечения.

Система САМПО+ ориентирована на пользователя, являющегося экспертом в области управления процессом разработки программных изделий, и обеспечивает привлечение его знаний, плохо поддающихся формализации. Данная особенность системы обуславливается использованием в ней методологии моделирования на основе алгоритмических сетей [56, 59], а также методологии принятия решений на основе алгоритмических сетей (прозрачные технологии), разработанных в Санкт-Петербургском институте информатики и автоматизации РАН (СПИИРАН) [61].

Важной особенностью системы САМПО+ является хранение прямых ссылок на документы, из которых взята информация, что связано с необходимостью обеспечения возможности контроля и повторного анализа вводимых значений.

Система реализована в среде Microsoft Excel 2007 (32-разрядная) на языке Visual Basic for Applications [62] и получила название САМПО+ (**С**истема АвтоМатизированной ПОддержки создания баз выполненных проектов).

4.2 Структура системы

Система представляет собой рабочую книгу Excel с набором из 30 рабочих листов, 9 модулей классов и 2 экранных форм. В таблице 4.1 представлены рабочие листы книги с описаниями, в таблице 4.2 – формы, в таблице 4.3 – модули. На рисунке 4.1 представлен снимок экрана с Лист12 (System) на котором располагаются системные настройки, текущие значение переменных, необходимых для работы системы, список источников и коды режимов системы.

Таблица 4.1. Рабочие листы системы САМПО+

№	Наименование	Описание
1	Лист11 (Заставка)	Заставка системы
2	Лист12 (System)	Лист, содержащий настройки и прочую системную информацию
3	Лист21 (Источники)	Перечень источников и их атрибутов, хранящихся в БВП
4	Лист22 (ИсточникиА)	Форма для добавления и редактирования информации об источнике
5	Лист23 (ИстоникиИ)	Форма для проведения исследования множества источников
6	Лист31 (Характеристики)	Перечень характеристик и их атрибутов, хранящихся в БВП
7	Лист32 (ХарактеристикиА)	Форма для добавления и редактирования информации о характеристике
8	Лист33 (ХарактеристикиТип)	Перечень характеристик с указанием типа
9	Лист34 (ХарактеристикиИ)	Форма для проведения исследования множества характеристик
10	Лист41 (Проекты)	Перечень проектов и их атрибутов, хранящихся в БВП
11	Лист42 (ПроектыА)	Форма для добавления и редактирования информации о проекте
12	Лист43 (ПроектыПути)	Перечень проектов с указанием пути к документам-источникам
13	Лист44 (ПроектыИ)	Форма для проведения исследования множества проектов
14	Лист51 (Значения)	Лист, содержащий информацию о значениях характеристик по каждому из проектов
15	Лист52 (ЗначенияА)	Форма для добавления и редактирования значений характеристик по проектам
16	Лист53 (ЗначенияКом)	Перечень комментариев к характеристикам по каждому из

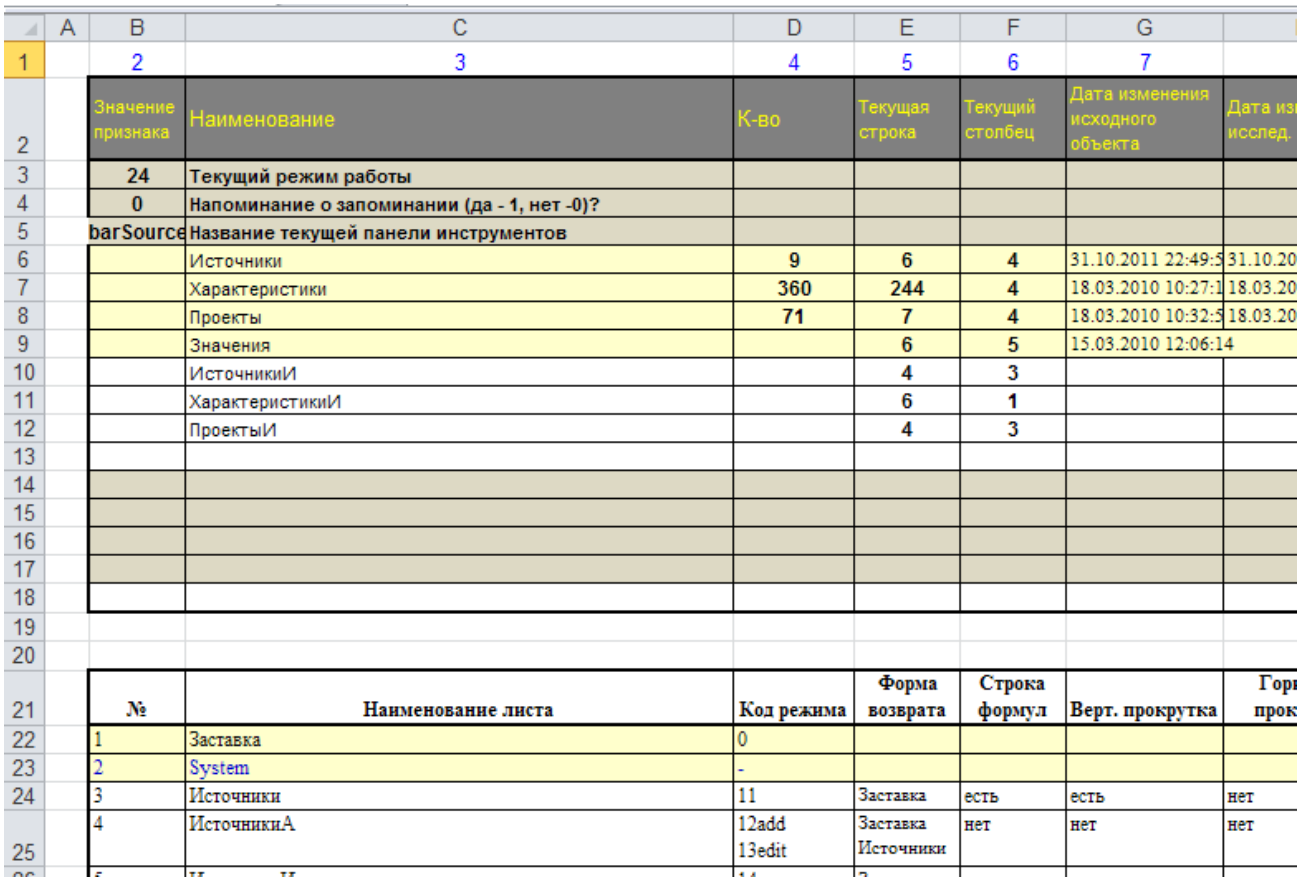
№	Наименование	Описание
		проектов
17	Лист54 (ЗначенияДок)	Перечень источников, в которых содержится характеристика по каждому из проектов
18	Лист55 (ЗначенияОбластей)	Форма для режима формирования области задания характеристики
19	Лист61 (ЗначенияИ в1)	Форма для определения функциональной пригодности БВП относительно внесенных проектов
20	Лист62 (ЗначенияИ в2)	Форма для исследования сформированных ансамблей характеристик БВП относительно внесенных проектов
21	Лист63 (ЗначенияИ д1)	Форма для определения функциональной пригодности БВП относительно добавляемых проектов
22	Лист64 (ЗначенияИ и1)	Форма для определения функциональной пригодности БВП относительно инициируемых проектов
23	Лист65 (ЗначенияИ и2)	Форма № 1 для исследования сформированных ансамблей характеристик БВП относительно инициируемых проектов
24	Лист66 (ЗначенияИ и3)	Форма №2 для исследования сформированных ансамблей характеристик БВП относительно инициируемых проектов
25	Лист81 (ПоискРмн)	Форма для формирования множества значений характеристик, описывающих исследуемый проект
26	Лист82 (ПоискРзн)	Форма для задания значений характеристик, описывающих исследуемый проект и поиска проектов аналогов
27	Лист83 (ПоискРан)	Форма для анализа значений характеристик выбранного проекта аналога
28	Лист91 (Зависимости)	Форма построения графика зависимости характеристик
29	Лист92 (БазаСостояние)	Форма, в графическом виде иллюстрирующая процент наполненности БВП
30	Лист93 (ПереченьПроектов)	Перечень просмотренных проектов

Таблица 4.2. Формы системы САМПО

№	Наименование	Описание
1	frmShow	Форма, отображаемая в процессе исследования характеристик и предлагающая выбор между отображением значений характеристик и картой вхождения
2	frmSort	Форма, отображаемая в процессе исследования характеристик и предлагающая варианты сортировки множества: по уникальным номерам, по возрастанию и убыванию заполнения

Таблица 4.3. Модули системы САМПО

№	Наименование	Описание
1	modChar	Модуль, содержащий функции работы с характеристиками
2	modCodomains	Модуль, содержащий функции обработки областей задания
3	modCommon	Модуль, содержащий функции общего назначения
4	modInf	Модуль, содержащий функции работы с зависимостями характеристик
5	modMenu	Модуль, содержащий функции обработки меню
6	modProj	Модуль, содержащий функции работы с проектами
7	modSearch	Модуль, содержащий функции поиска проектов
8	modSource	Модуль, содержащий функции работы с источниками
9	modValues	Модуль, содержащий функции работы со значениями характеристик



Значение признака	Наименование	К-во	Текущая строка	Текущий столбец	Дата изменения исходного объекта	Дата из исслед.
24	Текущий режим работы					
0	Напоминание о запоминании (да - 1, нет -0)?					
barSource	Название текущей панели инструментов					
	Источники	9	6	4	31.10.2011 22:49:5	31.10.20
	Характеристики	360	244	4	18.03.2010 10:27:1	18.03.20
	Проекты	71	7	4	18.03.2010 10:32:5	18.03.20
	Значения		6	5	15.03.2010 12:06:14	
	ИсточникиИ		4	3		
	ХарактеристикиИ		6	1		
	ПроектыИ		4	3		

№	Наименование листа	Код режима	Форма возврата	Строка формул	Верт. прокрутка	Гори прок
1	Заставка	0				
2	System	-				
3	Источники	11	Заставка	есть	есть	нет
4	ИсточникиА	12add 13edit	Заставка Источники	нет	нет	нет

Рисунок 4.1. Снимок экрана листа с системными настройками

4.3 Технология работы в системе

Система обеспечивает три основных режима работы:

- режим формирования БВП организации (подрежимы: формирование источников, проектов, характеристик, значений и областей задания);
- режим исследования БВП организации (подрежимы: исследование источников, проектов, характеристик и функциональной пригодности БД);
- режим использования БВП организации (подрежимы: поиск зависимостей между характеристиками, поиск проекта-аналога (ручной), поиск проекта-аналога (автоматизированный), прогноз значений характеристик и оценка выполнимости проекта).

Переход между режимами и подрежимами системы осуществляется через нажатие кнопки «Завершение работы в режиме» (расположена в главном меню системы).

В каждом режиме и подрежиме пользователь имеет возможность завершить работу в системе, нажав кнопку «Выход из системы» (расположена в главном меню системы). При этом система спрашивает: требуется ли запоминать изменения в файле БВП. Решение данного вопроса является прерогативой пользователя.

Также в системе существует меню «Справка», в котором находятся подменю «О системе», «Состояние базы» (в данном режиме можно ознакомиться с процентом заполнения информации о характеристиках по каждому из проектов), «Перечень просмотренных проектов» (в данном перечне указывается все рассмотренные проекты, даже те, которые не попали в исходное множество) и «Помощь», в котором пользователю предлагается ознакомиться с информацией о работе в системе в виде файла в формате Microsoft Word. На рисунке 4.2 представлена схема меню САМПО+ и соответствующие режимы работы системы.

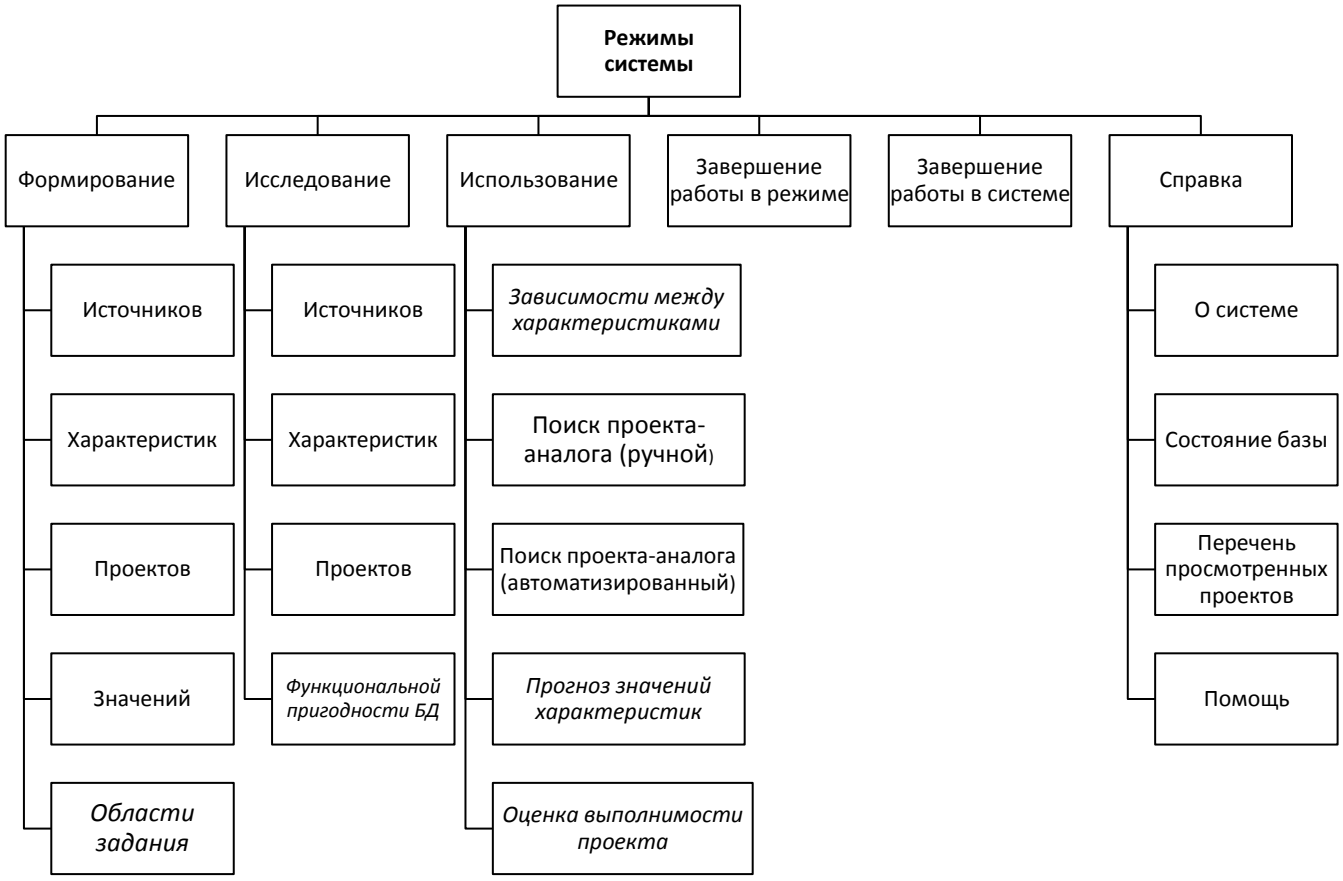


Рисунок 4.2. Режимы системы

Так как книга содержит макросы, после ее загрузки может появиться сообщение «Предупреждение системы безопасности. Запуск макросов отключен». В этом случае необходимо открыть параметры безопасности и выбрать режим «Включить это содержимое»), после чего в главном меню Excel появится режим «Надстройки». После его активации на экране появится Главное меню системы.

Рассмотрим подробнее работу в каждом из режимов.

4.3.1 Режим формирования базы выполненных проектов

Добавление, удаление и редактирование характеристик. Для ввода новой характеристики необходимо нажать кнопку «Добавить характеристику». После нажатия кнопки появляется окно «Ввод значений атрибутов характеристики», в котором продолжается работа. Для редактирования характеристики необходимо выбрать требуемую характеристику и нажать кнопку «Редактировать характеристику». После нажатия кнопки появляется окно «Ввод значений атрибутов характеристики», в котором продолжается работа. Для удаления

характеристики необходимо выбрать требуемую характеристику нажать кнопку «Удалить характеристику». После нажатия кнопки система запрашивает подтверждение на необходимость удаления выбранной характеристики («Вы действительно хотите удалить характеристику «наименование-характеристики»?»). И, в случае положительного ответа, характеристика удаляется из базы данных.

Окно, обеспечивающее ввод атрибутов характеристики, представлено на рис.4.3. Ввод атрибутов происходит в соответствии с графой «Возможные значения» определенной в табл. 2.2 главы 2.

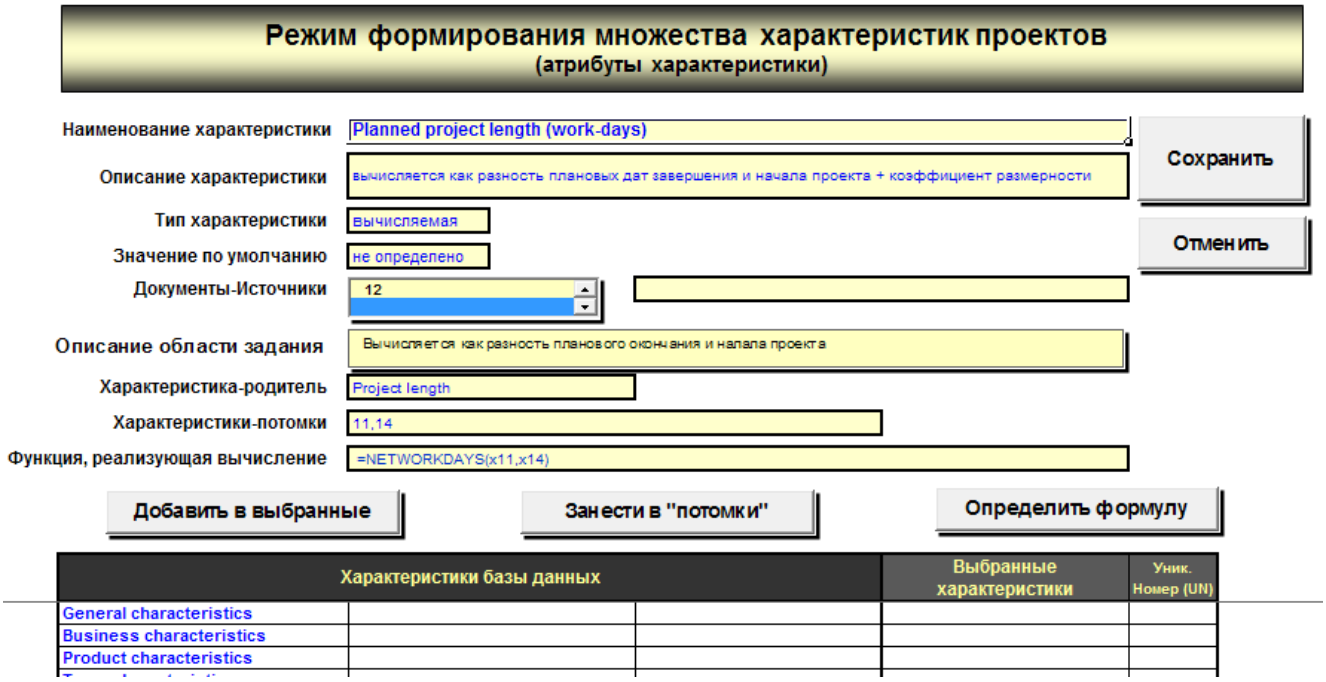


Рисунок 4.3. Ввод атрибутов характеристики

Отдельно необходимо описать процедуру выбора характеристики-родителя, характеристики-потомка и формирования формулы: выбирается ячейка, в которую должна быть занесена характеристика-родитель. Щелчок правой кнопкой мыши позволяет перейти в таблицу характеристик проектов. По умолчанию активируется ячейка «General characteristics». Все характеристики таблицы, имеющие потомков выделены жирным шрифтом и окрашены в синий цвет (интегральные характеристики). Если требуется посмотреть характеристики,

входящие в состав интегральных, активируется соответствующая ячейка и осуществляется щелчок правой кнопкой мыши. Поиск осуществляется пока не будет найдена характеристика, являющаяся родителем по отношению к вводимой. Двойной щелчок левой кнопки мыши окрашивает выбранную ячейку в желтый цвет и помещает ее значение в ячейку, которая должна содержать значение характеристики родителя. Затем система активирует ячейку, которая должна содержать значение характеристик потомком вводимой характеристики.

Процедура поиска характеристик потомков внутри одной категории аналогична описанной выше процедуре поиска характеристики родителя: щелчок правой кнопкой мыши позволяет перейти в таблицу характеристик проектов. По умолчанию активируется ячейка «General characteristics». Все характеристики таблицы, имеющие потомков выделены жирным шрифтом и окрашены в синий цвет. Если требуется посмотреть характеристики, входящие в состав интегральных, активируется соответствующая ячейка и осуществляется щелчок правой кнопкой мыши. Все характеристики, которые, по мнению пользователя, являются потомками вводимой характеристики, выделяются желтым цветом (двойной щелчок левой кнопкой мыши). Когда множество характеристик рассматриваемой категории сформировано, нажимается кнопка «Добавить в выбранные». В результате чего множество выделенных цветом характеристик помещается в столбец «выбранные характеристики», а в столбец «уникальный номер UN» помещаются их уникальные номера. Полученный список может быть еще раз отредактирован: двойной щелчок левой кнопкой мыши убирает характеристику и ее уникальный номер из списка. Нажатие кнопки «Занести в «потомки»» формирует список потомков в ячейке значений характеристик потомков и готовит материал для формулы, если характеристика вычисляемая (см. рис.4.3), формула определяется путем соответствующего редактирования «заготовки», например «=x11+x14» → «=NETWORKDAYS(x11, x14)». После приведения формулы к нужному виду нажимается кнопка «Определить формулу». Система приводит формулу к машинному виду, после чего окрашивает содержимое ячейки в синий цвет.

Добавление, удаление и редактирование проектов. Процедуры работы с множеством проектов аналогичны вышеописанным процедурам работы с множеством характеристик.

Добавление, удаление и редактирование источников. Процедуры работы с множеством источников аналогичны вышеописанным процедурам работы с множеством характеристик.

Ввод и редактирование значений характеристик. При входе в режим, можно определить, по какому сценарию вводить значения: выбрать характеристику и ввести ее значения по всем проектам, либо же выбрать проект и ввести значения всех его характеристик. На рисунке 4.4 представлен ввод значений характеристик по одному из проектов.

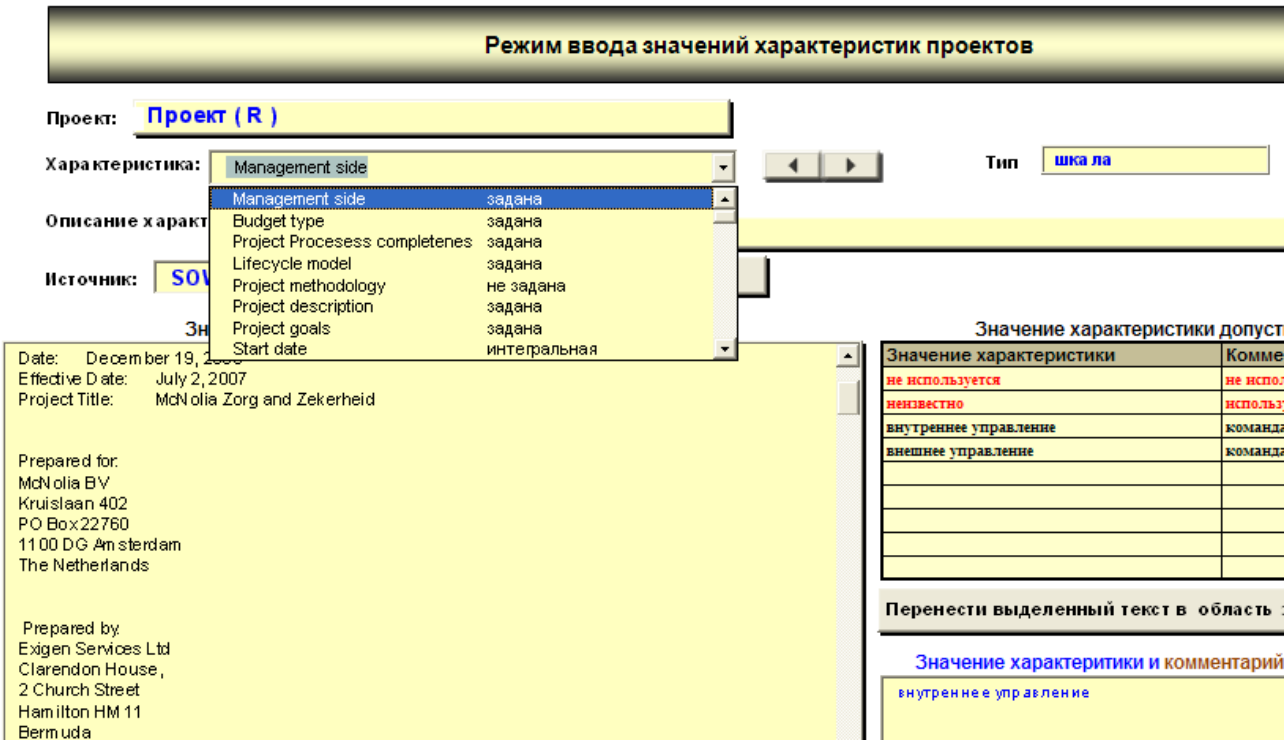


Рисунок 4.4. Ввод значений характеристик

Как уже отмечалось, наряду с собственно значением характеристики, в системе хранится информация об источнике этого значения. Для каждой характеристики система формирует список доступных источников. Список получается путем пересечения множеств источников доступных

рассматриваемому проекту и источников, с которыми связана рассматриваемая характеристика. Если пересечение не пусто, список документов отображается в выпадающем списке «Источник» и, после нажатия кнопки «Загрузить документ», выбранный документ может быть загружен в систему. В текстовом окне найденное значение характеристики выделяется и после нажатия кнопки «Перенести выделенный текст в область значений, вносимых в базу» переносится в текстовое окно «Значение характеристики и комментарий к ней, помещаемые в базу». Путь к документу источнику (ячейка «Адрес ссылки») и местоположение выделенного фрагмента в документе (ячейка «Указатель») запоминаются. Значения в текстовых полях «Значение» и «Комментарий» могут редактироваться, если тип характеристики «текст» или «число».

Если тип характеристики «шкала» значение ее может быть взято исключительно из таблицы допустимых значений. В этом случае в таблице выбирается требуемое значение и нажимается кнопка «Перенести выделенный текст в область значений, вносимых в базу». Выделенное значение заносится в текстовое поле «Значение» и недоступно для редактирования. Текстовое поле «Комментарий» может редактироваться.

Чтобы занести значение характеристики в базу, необходимо нажать кнопку «Занести значение характеристики в базу данных».

При переходе к следующей характеристике или попытке выйти из режима система контролирует, была ли осуществлена запись введенного значения.

Определение области задания характеристики. Режим позволяет изменить область задания характеристик, в случае, если значение характеристики уже заданы (см. рис 4.5).

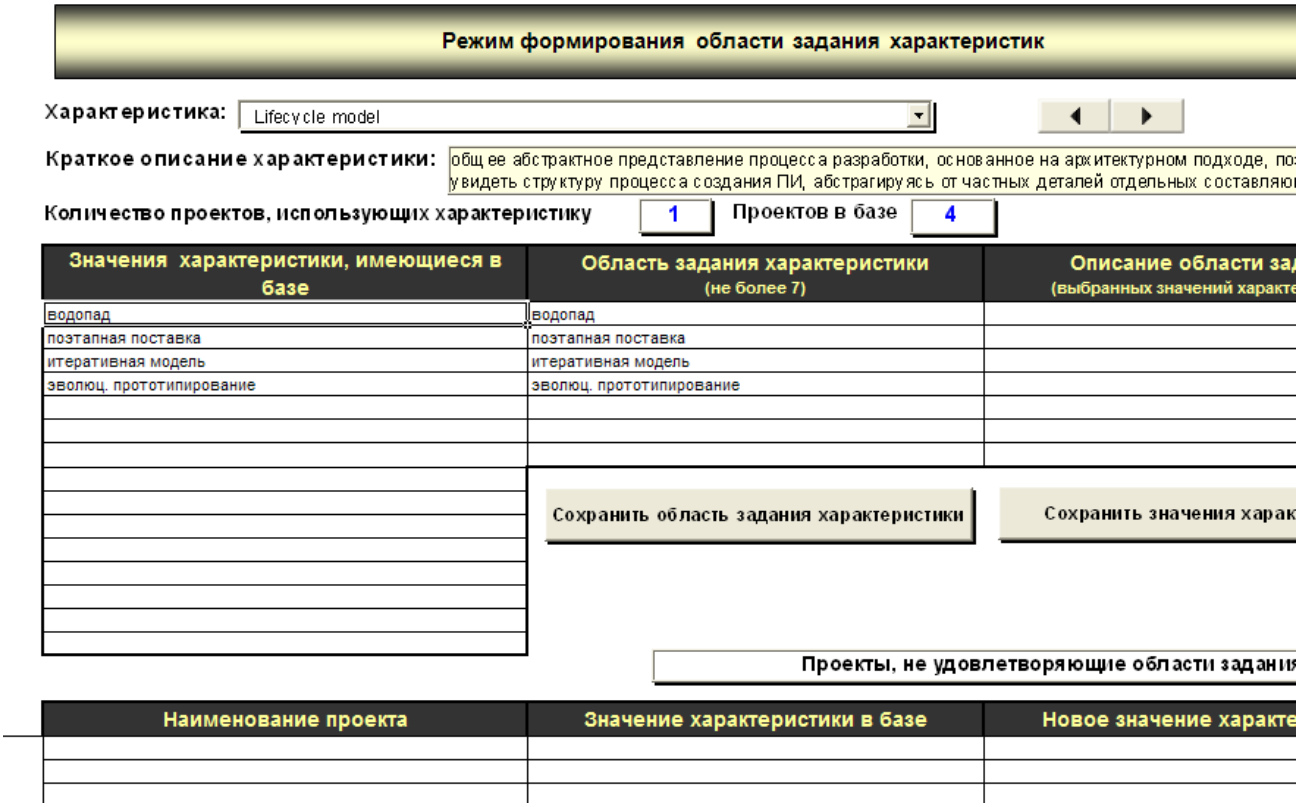


Рис.4.5. Режим формирования области задания характеристик

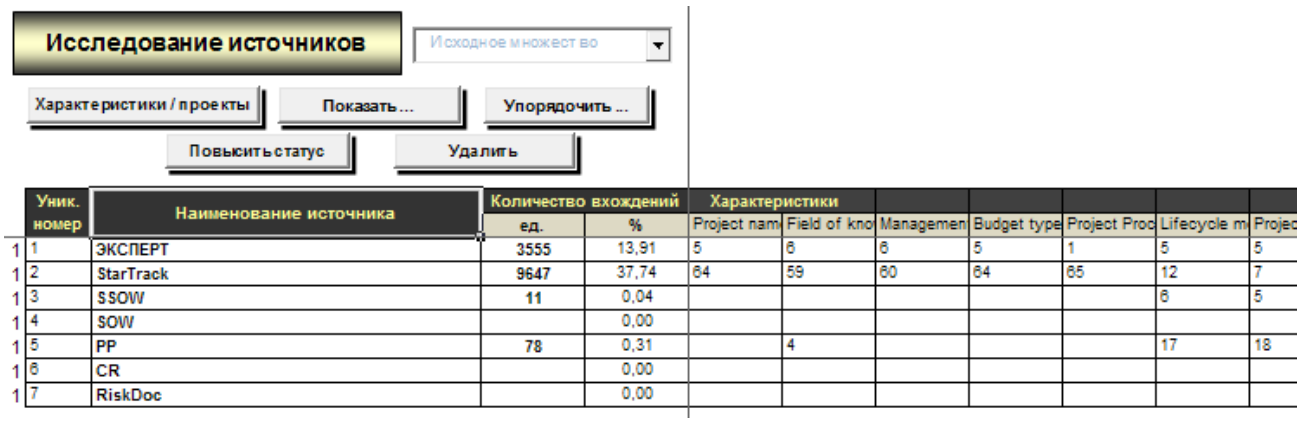
Выбор характеристики может быть произведен как непосредственно из выпадающего списка, так и путем его листания. При этом появляется краткое описание выбранной характеристики, а также количество проектов, в которых ее значение уже занесено. Система позволяет проанализировать уникальные значения характеристики, имеющиеся в базе, сформировать новую область задания и ее описание.

Чтобы сохранить отредактированную область задания необходимо нажать кнопку «Сохранить область задания характеристики». После чего система проверит значения характеристики, имеющиеся в базе на соответствие новой области. При наличии расхождений система выдает проекты, имеющие значения, не удовлетворяющие области задания и сами значения. В случае если область задания признана правильной, меняются значения характеристик в проектах (столбец «Новое значение характеристики») и нажимается кнопка «Сохранить значения характеристик»).

4.3.2 Режим исследования базы выполненных проектов

Исследование источников. В данном режиме можно провести исследование всех множеств источников (исходного, уточненного и рекомендуемого), узнать количество вхождений (используемость) в процентном и абсолютном значении в целом, а также по каждой характеристике или проекту в частности. Проводить сортировку множества по уникальным номерам, увеличению или уменьшению количества вхождений, а также просмотреть карту вхождений. Помимо этого можно управлять составом множеств, то есть переносить источники из исходного в уточненное и т.д.

На рисунке 4.6 представлена форма исследования источников исходного множества.



	Уник. номер	Наименование источника	Количество вхождений		Характеристики						
			ед.	%	Project nam	Field of kno	Managemen	Budget type	Project Proc	Lifecycle m	Projec
1	1	ЭКСПЕРТ	3555	13,91	5	6	6	5	1	5	5
1	2	StarTrack	9647	37,74	64	59	60	64	65	12	7
1	3	SSOW	11	0,04						6	5
1	4	SOW		0,00							
1	5	PP	78	0,31		4				17	18
1	6	CR		0,00							
1	7	RiskDoc		0,00							

Рисунок 4.6. Режим исследования источников

Исследование характеристик. Процедуры исследования множества характеристик аналогичны вышеописанным процедурам исследования множества источников.

Исследование проектов. Процедуры исследования множества проектов аналогичны вышеописанным процедурам исследования множества источников.

4.3.3 Режим использования базы выполненных проектов

Поиск проекта-аналога (ручной). Режим позволят искать проекты-аналоги при помощи встроенных средств Excel (фильтры). До начала процедуры поиска

необходимо сформировать множество характеристик, описывающих исследуемый проект (см. рис 4.7).

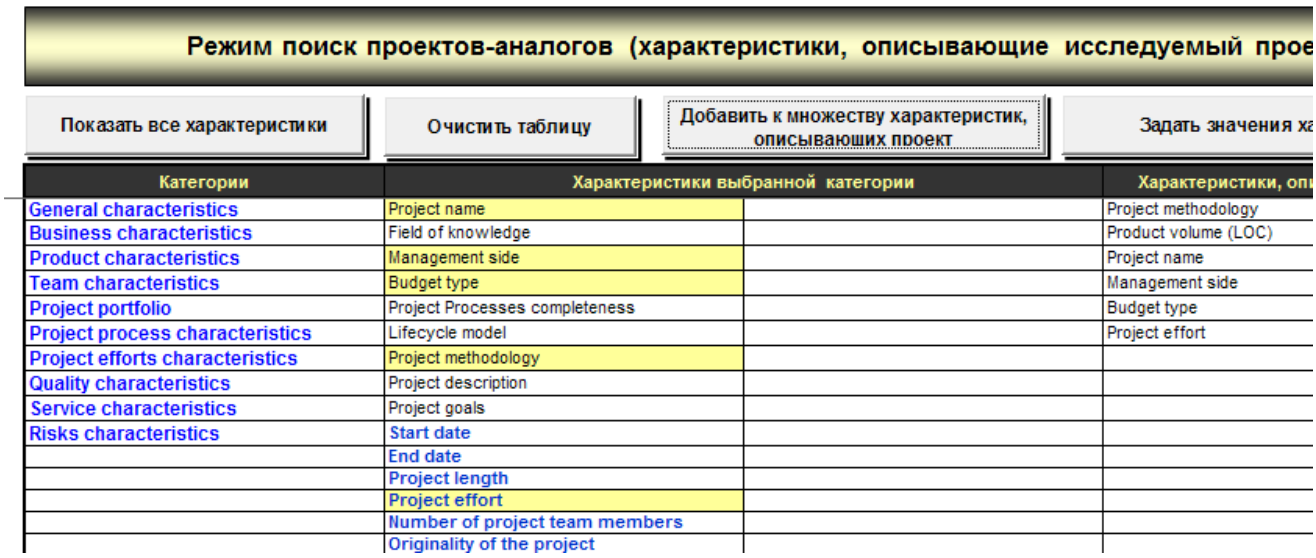


Рисунок 4.7. Выбор характеристик, описывающих проект

Процедура поиска характеристик аналогична процедуре, описанной в разделе 4.3.1. По умолчанию активируется ячейка «General characteristics». Все характеристики таблицы, имеющие потомков, выделены жирным шрифтом и окрашены в синий цвет (интегральные характеристики). Если требуется посмотреть характеристики, входящие в состав интегральных, активируется соответствующая ячейка и осуществляется щелчок правой кнопкой мыши. Все характеристики, которые, по мнению пользователя, отражают какие-либо свойства анализируемого проекта выделяются желтым цветом (двойной щелчок левой кнопкой мыши). Когда множество характеристик категории сформировано нажимается кнопка «Добавить к множеству характеристик, описывающих проект». В результате чего множество выделенных цветом характеристик помещается в столбец «характеристики, описывающие проект». Имеется возможность перехода к выбору характеристик другой категории и так до тех пор, пока не будет сформирован требуемый список.

Список характеристик, описывающих проект, может быть еще раз отредактирован: двойной щелчок левой кнопкой мыши убирает характеристику из списка.

Нажатие кнопки «Задать значение характеристик» позволяет перейти к следующей фазе подрежима: заданию значений характеристик и выбору проектов-аналогов.

Очистить таблицу можно нажав кнопку «Очистить таблицу» (будут уничтожены все введенные характеристики за исключением характеристик-категорий).

Кнопка «Показать все характеристики» выводит во второй столбец таблицы все характеристики базы. Формирование множества характеристик, описывающих проект, в этом случае происходит путем подсвечивания нужных характеристик (двойной щелчок) и перенесения подсвеченных характеристик в искомое множество после нажатия кнопки «Добавить к множеству характеристик, описывающих проект». Операции, связанные с декомпозицией интегральных характеристик (щелчок правой кнопкой мыши), в этом случае блокированы.

Нажатие кнопки «Задать значение характеристик» осуществляет переход к заданию значений характеристик и выбору проектов-аналогов. Как видно из рисунка 4.8, в таблице в качестве столбцов выступает множество характеристик, описывающих анализируемый проект, тогда как проекты, содержащиеся в основной базе и значения их характеристик, предстают в виде строк.

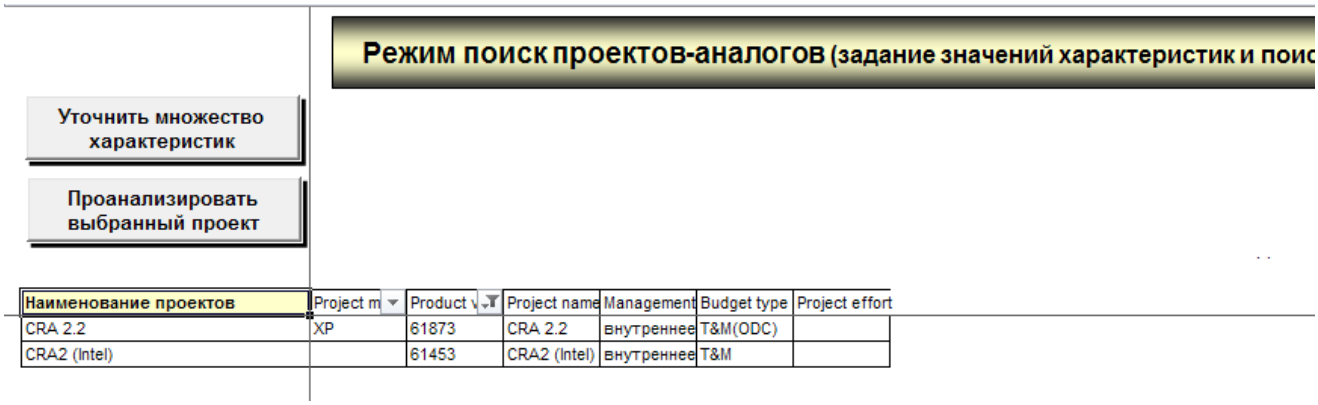


Рисунок 4.8. Режим ручного поиска проекта-аналога

Поиск проектов-аналогов происходит штатными средствами Excel путем задания соответствующих значений в автофильтрах, помещенных в заголовке таблицы.

Нажатие кнопки «Уточнить множество характеристик» позволяет вернуться к предыдущему окну и изменить множество характеристик, описывающих исследуемый проект.

Выбор проекта, полученного в результате применения автофильтра и нажатие кнопки «Проанализировать выбранный проект» позволяет перейти к заключительному этапу подрежима – этапу анализа выбранного проекта-аналога.

Поиск проекта-аналога (автоматизированный). Процедура автоматизированного поиска проекта-аналога аналога аналогична процедуре ручного поиска за исключением последнего этапа – система сама ищет множество проектов-аналогов.

4.4 Результаты применения системы

Результаты работы над созданием БВП ООО «Эксиджен Сервисес» приведены в таблице 4.4.

Таблица 4.4. Исходные, уточненные и рекомендуемые множества «Эксиджен Севисес»

Тип множества	**Вид множества**		
	Источники	Характеристики	Проекты
Исходное	7	360	342
Уточненное	6	168	71
Рекомендуемое	6	137	66

Для проведения испытаний системы были выбраны 4 представительных проекта из БВП; проекты, их характеристики представлены в таблице 4.5.

Таблица 4.5. Фактические данные из метрической базы данных ООО «Эксиджен Сервисес»

Наименование проекта	**KLOC**	**Реальная трудоемкость, человеко-месяцы**	**Число членов команды**	**Время разработки, месяцы**
Paddaso	118	269,64	22	13
LEDW	12,5	28,8	4	8,3
MIZM	4,4	9,9	6	2,5
UHPSM	8,6	24,6	6	4,7

В таблице 4.6 приведены отношения в процентном и абсолютном значении трудоёмкости проекта и времени разработки для модели COCOMO со стандартами коэффициентами.

Таблица 4.6. Сравнение плановых и фактических показателей (COCOMO со стандартными коэффициентами)

Наименование проекта	**Расхождение в трудоемкости, человеко-месяцы**	**Расхождение в трудоемкости, %**	**Расхождение во времени разработки, месяцы**	**Расхождение во времени разработки, %**
Paddaso	89,85	33,3%	2,32	17,8%
LEDW	5,24	18,2%	-1,78	-21,4%
MIZM	1,47	14,8%	1,88	75,2%
UHPSM	-1,62	-6,6%	0,96	20,4%
Среднее:		**18,2%**		**33,7%**

В таблице 4.7 приведены отношения в процентном и абсолютном значении трудоёмкости проекта и времени разработки для модели COCOMO с усреднёнными показателями.

Таблица 4.7. Сравнение плановых и фактических показателей (COCOMO с усредненными коэффициентами)

Наименование проекта	**Расхождение в трудоемкости, человеко-месяцы**	**Расхождение в трудоемкости, %**	**Расхождение во времени разработки, месяцы**	**Расхождение во времени разработки, %**
Paddaso	46,41	17,2%	0,54	4,2%
LEDW	1,12	3,9%	-2,53	-30,5%
MIZM	0,09	0,9%	1,38	55,2%
UHPSM	-4,39	-17,8%	0,3	6,4%
Среднее		**10,0%**		**24,1%**

В таблице 4.8 приведены отношения в процентном и абсолютном значении трудоёмкости проекта и времени разработки для модели COCOMO II со стандартными показателями.

Таблица 4.8. Сравнение плановых и фактических показателей (COCOMO II со стандартными коэффициентами)

Наименование проекта	**Расхождение в трудоемкости, человеко-месяцы**	**Расхождение в трудоемкости, %**	**Расхождение во времени разработки, месяцы**	**Расхождение во времени разработки, %**
Paddaso	288,56	107,0%	16,6	127,7%
LEDW	14,9	51,7%	4,8	57,8%
MIZM	5,1	51,5%	6,5	260,0%
UHPSM	6,7	27,2%	6,7	142,6%
Среднее		**59,4%**		**147,0%**

В таблице 4.9 приведены отношения в процентном и абсолютном значении трудоёмкости проекта и времени разработки, полученные для модели COCOMO II с усредненными показателями.

Таблица 4.9. Сравнение плановых и фактических показателей (COCOMO II с усреднёнными коэффициентами)

Наименование проекта	**Расхождение в трудоемкости, человеко-месяцы**	**Расхождение в трудоемкости, %**	**Расхождение во времени разработки, месяцы**	**Расхождение во времени разработки, %**
Paddaso	-8,44	-3,1%	10	76,9%
LEDW	-0,9	-3,1%	2,7	32,5%
MIZM	-0,7	-7,1%	5,1	204,0%
UHPSM	0,8	3,3%	6	127,7%
Среднее		**4,1%**		**110,3%**

В таблице 4.10 приведены отношения в процентном и абсолютном значении трудоёмкости проекта и времени разработки, полученные при помощи системы САМПО+.

Таблица 4.10. Сравнение плановых и фактических показателей (САМПО+)

Наименование проекта	**Расхождение в трудоемкости, человеко-месяцы**	**Расхождение в трудоемкости, %**	**Расхождение во времени разработки, месяцы**	**Расхождение во времени разработки, %**
Paddaso	*Проект-аналог не найден*			
LEDW	*Значения трудоемкости отсутствуют*		-0,415	-5,0%
MIZM	0,7	7,0%	-0,15	-6,0%
UHPSM	-1,48	-6,0%	-0,19	-4,0%
Среднее		**6,5%**		**5%**

В таблице 4.11 сведены расхождения по оценкам трудоёмкости и времени разработки оцененные с помощью моделей COCOMO и COCOMO II со стандартными и усредненными коэффициентами, а также системы САМПО+

Таблица 4.11. Сводная таблица по точности оценки

Метод оценки	Расхождение по трудоемкости, %	Расхождение по времени разработки, %
COCOMO со стандартными коэффициентами	18,2%	33,7%
COCOMO со усредненными коэффициентами	10%	24,1%
COCOMO II со стандартными коэффициентами	59,4%	147%
COCOMO II со усредненными коэффициентами	4,1%	110,3%
САМПО+	**6,5%**	**5%**

4.5 Выводы

В рамках реализации метода формирования пространства характеристик для оценки необходимых программному проекту ресурсов была разработана система САМПО+ для поддержки создания баз выполненных проектов компаний, разрабатывающих программное обеспечение. Система позволяет проводить оценку выполнимости проектов, инициируемых в компании, для которой проводилось исследование, и реализована на базе предложенного метода гибких оценок с использованием формализма алгоритмических сетей.

В системе реализованы основные режимы поддержки создания БВП: формирование и исследование множеств источников, характеристик и проектов, а также процедуры автоматизированного и ручного поиска проекта-аналога.

Проведен эксперимент с представительными данными по выполненным проектам (данные компании "Эксиджен Сервисис" по более чем 60 проектам в течение 2-х лет). В отличие от традиционных методов COCOMO и COCOMO II со средним расхождением между оценками и фактическими показателями 15%, расхождение между оценками системы САМПО+ и фактическими величинами трудоемкости и времени выполнения разработки для представительных 4-х проектов, участвовавших в эксперименте, не превысило 7%.

Заключение

Полученные в диссертационном исследовании результаты представляют собой решение актуальной задачи повышения точности и оперативности оценки ресурсов, необходимых программному проекту.

В ходе исследования были получены следующие основные результаты:

1) разработана модель формирования базы выполненных проектов для поиска проектов-аналогов с учетом неопределенного характера информация о значениях характеристик, описывающих проекты, неполноты исходных данных, слабо формализуемого опыта и высокой квалификация экспертов, проводящих оценку.
2) разработан метод формирования пространства характеристик для оценки ресурсов, необходимых для выполнения проектов разработки программных изделий на основе алгоритмических сетей (метод гибких оценок). Особенность разработанного метода заключается в специфике использования формализма алгоритмических сетей для формирования пространства характеристик, используемого для оценки ресурсов, необходимых для выполнения проектов разработки программных изделий (метод гибких оценок). Данный формализм используется для привлечения опыта экспертов, плохо поддающегося формализации, но при этом не требующего от самих экспертов специальных знаний в области программирования. Применение данного метода позволило уточнить оценки необходимых ресурсов – расхождение между плановыми и фактическими величинами не превышает 7%.
3) разработана модель программной системы для автоматизированного поиска ближайших проектов-аналогов по базе выполненных проектов для оценки необходимых ресурсов по принципу подобия. Также для использования в системе было собрано и классифицировано множество, содержащее более 100 метрик, характеризующих программный проект, создаваемый продукт и процесс разработки, учет которых позволяет

производить обоснованную оценку ресурсов, необходимых программному проекту.

Результаты диссертационного исследования внедрены в ООО «Эксиджен Сервисес» (в конце 2009 года компания претерпела изменения и стала именоваться «Ф-Лайн Софтвер») и НП «Объединение подземных строителей». Акты внедрения представлены в приложениях 1 и 2 соответственно.

Они также могут быть использованы в перспективных НИР, при разработке ОКР по созданию ПО и в учебном процессе при подготовке специалистов в области ПО.

Дальнейшее развитие научных исследований в области повышения точности и оперативности оценок может быть проведено в следующих направлениях:

1) уточнение универсального множества характеристик, используемых для описания проектов.
2) добавление в процедуры поиска проекта-аналога весов характеристик для более точного ранжирования проектов при выдаче результата пользователю.
3) разработка дополнительных режимов системы: формирование области задания характеристики (анализ значений характеристики и выработка шкалы, либо области задания), поиск зависимостей между характеристиками (определение возможной корреляционной зависимости между характеристиками), прогноз значений характеристик (определение значений дополнительных характеристик инициируемого проекта).

Список сокращений и условных обозначений

АВО – Алгоритм вычисления оценок

БВП – База выполненных проектов

БД – База данных

ЖЦ – Жизненный цикл

ИТ – Информационные технологии

ЛПР – Лицо, принимающее решение

ПИ – Программное изделие

ПО – Программное обеспечение

ПП – Проектный процесс (производственный процесс проекта)

САМПО+ – Система автоматизированного поиска проекта-аналога

СППО – Стандартный производственный процесс организации

СУБД – Система управления базами данных

Список литературы

1. Гласс Р. Факты и заблуждения профессионального программирования. — Пер. с англ.— СПб.: Символ-Плюс, 2007. — 240 с.

2. Brooks F. The Mythical Man-Month. Addison-Wesley. ISBN 0-201-00650-2: *Русский перевод*: Брукс Ф. Мифический человеко-месяц или как создаются программные системы. – Пер. с англ. – СПб.: Символ-Плюс, 1999. – 304 с.

3. Boehm B. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

4. Putnam, Lawrence H. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering, vol. SE-4, No. 4, pp 345-361.

5. Mazel B. The Role of Computer Simulation in Corporate Management: An Overview, Page 8, December 1975

6. Million Lines of Code | Information Is Beautiful [Электронный ресурс]. – Режим доступа: http://www.informationisbeautiful.net/visualizations/million-lines-of-code, свободный. – Загл. с экрана (дата обращения 04.09.2015).

7. Albrecht A.J., Gaffney J.E. Software function, source lines of codes, and development effort prediction: a software science validation. – IEEE Trans Software Eng. – 1983. – 4 p.

8. Parkinson S.N. Parkinson's Law and Other Studies in Administration. – Houghton-Miffin. – 1957.– 148 p

9. Coates J. Technological Forecasting and Social Change. – Elsevier Science Inc. – 1999. – 235 p

10. Shepperd M., C. Schofield Estimating software project effort using analogy. – IEEE Trans Software Eng. – 1997. – P. 736 – 743.

11. Иванищев В.В. Об одной модели региона // Алгоритмы и системы автоматизации исследований и проектирования. М.:Наука,1980. С.13-16.

12. Иванищев В.В. Алгоритмический базис для описания механизмов экономики // Алгоритмические модели в автоматизации исследований. М.:Наука, 1980. С.37-42.

13. Иванищев В.В., Марлей В.Е., Морозов В.П. Язык алгоритмических сетей: препринт № 63 ЛНИВЦ АН СССР. – Л.,1984. – 37 с

14. Михайлов В.В., Марлей В.Е. Алгоритмические сети и их применение. Санкт-Петербург. Изд-во СПбГУАП, 2004. 80 с.

15. Морозов В.П. Методология и системы моделирования на основе алгоритмических сетей // X Санкт-Петербургская международная конф. «Региональная информатика-2006» («РИ-2006»), Санкт-Петербург, 24–26 октября 2006 г.: труды конференции. – СПб.: СПОИСУ, 2007. – С. 112–118.

16. Марлей В.Е., Михайлов В.В. Королев О.Ф. Алгоритмические сети и их применение. Издание 2-е, дополненное. – СПб.: ГУАП, 2012. 132 с.

17. Иванищев В. В. Технология множественного моделирования на основе формализма алгоритмических сетей // Сб. научных трудов КИИ-96. Т. III, Казань. 1996. — С.466–468.

18. Иванищев В.В. Базы фрагментарных моделей в задачах автоматизации моделирования // Труды СПИИРАН. 2002. Вып. 1. С. 125-131.

19. Иванищев В.В., Михайлов В.В. Автоматизация моделирования экологических систем. СПб.: Изд-во СПбГТУ, 2000. — 172 с.

20. Иванищев В.В. Модель поведения поставщиков и потребителей на распределенном рынке // Труды СПИИРАН. 2004. Вып. 2. С. 47-56.

21. Марлей В. Е. Алгоритмические сети и параллельные вычисления // Труды СПИИРАН, Вып. 1, т. 2. — СПб: СПИИРАН, 2002.

22. Морозов В. П., Калугина Е. А., Тележкин А. М. Система поддержки создания баз исторических данных компаний, разрабатывающих программное обеспечение / Четвертая Всероссийская научно-практическая Конференция «Имитационное моделирование. Теория и практика» (ИММОД-2009), Санкт-Петербург, 21-23 октября 2009 года

23. Баранов С.Н. Тележкин А.М. Метрическое обеспечение программных разработок / Труды СПИИРАН. - 2014. № 5(36). – С. 5-27

24. Боэм Б.У. Инженерное проектирование программного обеспечения: Пер. с англ. - М.: Радио и связь, 1985. – 512 с.

25. ISO/IEC 20926:2009 - Software and systems engineering – Software measurement – IFPUG functional size measurement method 2009 [Электронный ресурс]. – Режим доступа: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51717, свободный. – Загл. с экрана (дата обращения: 04.09.2015).
26. AFPVs. 1.0 [Электронный ресурс]. – Режим доступа: http://www.omg.org/spec/AFP/1.0, свободный. – Загл. с экрана (дата обращения 04.09.2015).
27. About the Object Management Group [Электронный ресурс]. – Режим доступа: http://www.omg.org/gettingstarted/gettingstartedindex.htm, свободный. – Загл. с экрана (дата обращения: 04.09.2015).
28. COCOMO® II – Constructive Cost Model [Электронный ресурс]. – Режим доступа: http://csse.usc.edu/tools/COCOMOII.php, свободный. – Загл. с экрана (дата обращения: 04.09.2015).
29. НОУ ИНТУИТ | Лекция | Управление разработкой ПО [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/64/64/lecture/1896?page=6, свободный. – Загл. с экрана (дата обращения 04.09.2015).
30. Models for Competing on Schedule, Cost, and Quality [Электронный ресурс]. – Режим доступа: http://www.slideshare.net/Sixsigmacentral/models-for-competing-on-schedule-cost-and-quality, свободный. – Загл. с экрана (дата обращения 04.09.2015).
31. COCOMO II modeling manual [Электронный ресурс]. – Режим доступа: http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf, свободный. – Загл. с экрана (дата обращения: 04.09.2015).
32. Function Point Programming Languages Table [Электронный ресурс]. – Режим доступа: http://www.qsm.com/resources/function-point-languages-table, свободный. – Загл. с экрана (дата обращения: 04.09.2015).
33. Oracle Unified Method | Область знаний | Oracle Partner Network [Электронный ресурс]. – Режим доступа:

http://www.oracle.com/partners/ru/products/applications/oracle-unified-method/get-started/index.html, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

34. Ковалев В.В., Волкова О.Н. Анализ хозяйственной деятельности предприятия. Методы экспертных оценок. М.: ТК Велби, 2002. — 424 с.

35. Enterprise Program Management Software | OpenPlan | Deltek [Электронный ресурс]. – Режим доступа: http://www.deltek.com/products/ppm/schedule/openplan, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

36. Capabilities [Электронный ресурс]. – Режим доступа: http://www.deltek.com/products/ppm/schedule/openplan/capabilities, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

37. Project Managementand Earned Value Management Solution | ArtemisViews | Artemis [Электронныйресурс]. – Режим доступа: http://www.aisc.com/views, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

38. Характеристики Спайдер Проджект [Электронный ресурс]. – Режим доступа: http://www.spiderproject.ru/booklet_r.php?p=2|6, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

39. Спайдер Проджект: Управление Проектами | Project Management |консалтинг | обучение | Spider Project [Электронный ресурс]. – Режим доступа: http://www.spiderproject.ru, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

40. Microsoft Project – программа для управления проектами [Электронный ресурс]. – Режим доступа: https://products.office.com/ru-ru/project/project-and-portfolio-management-software, свободное. – Загл. с экрана (дата обращения: 04.09.2015).

41. Basecamp is everyone’s favorite project management app. [Электронный ресурс]. – Режим доступа: https://basecamp.com, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

42. Two big announcements. [Электронный ресурс]. – Режим доступа: https://37signals.com, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

43. JIRA - Issue & Project Tracking Software | Atlassian [Электронныйресурс]. Режим доступа: https://www.atlassian.com/software/jira, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

44. Development and Collaboration Software Company | About Atlassian [Электронный ресурс]. – Режим доступа: https://www.atlassian.com/company, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

45. Supported Platforms - Atlassian Documentation [Электронный ресурс]. – Режим доступа: https://confluence.atlassian.com/jira/supported-platforms-207488170.html, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

46. Integrating JIRA with Code Development Tools – Atlassian Documentation [Электронный ресурс]. – Режим доступа: https://confluence.atlassian.com/jira/integrating-jira-with-code-development-tools-395707042.html, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

47. Wrike – Система для управления проектами и совместной работы онлайн [Электронный ресурс]. – Режим доступа: https://www.wrike.com/ru, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

48. Interview with Wrike – Folksonomy [Электронный ресурс]. – Режим доступа: http://web.archive.org/web/20071117093656/http://www.folksonomy.org/2007/03/interview_with_wrike, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

49. Worksection – система управления проектами, управление бизнесом, планирование и отчеты [Электронный ресурс]. – Режим доступа: http://worksection.com, свободный. – Загл. с экрана (дата обращения: 04.09.2015).

50. Обзор системы управления задачами Worksection – Cloudrethink.ru [Электронный ресурс]. – Режим доступа: http://cloudrethink.ru/worksection-test-review, свободный. Загл. с экрана (дата обращения: 04.09.2015).

51. Поспелов Г.С. Искусственный интеллект – основа новой информационной технологии. М.: Наука, 1988. 279 с.

52. Глушков В.М. Основы безбумажной информатики. М.: Наука, 1982. 168 с.

53. Рыжко А.Л., Лобанова Н.М., Рыжко Н.А., Кучинская Е.О. Экономика информационных систем: учебное пособие. – М.: Финансовый университет, 2014. – С. 41.

54. Егоров А.Г., Морозов В.П., Тележкин А.М., Тубольцева В.В. Система автоматизации поиска проектов-аналогов для оценивания выполнимости проектов разработки программных изделий / Региональная информатика-2008 (РИ-2008) XI Санкт-Петербургская Международная Конференция, Санкт-Петербург, 22-24 октября 2008 года: Материалы конференции \ СПОИСУ. - СПб, 2008. С. 32 - 33.

55. Тележкин А.М. Создание исторических баз данных при помощи системы САМПО+ // Региональная информатика (РИ-2012). Санкт-Петербург, 24-26 октября 2012 г. Труды конференции. – СПб., 2013. –84-90.

56. Морозов В.П., Тележкин А.М. Формирование пространства характеристик для определения ресурсов, обеспечивающих успешное завершение проекта, в системе САМПО+ / Региональная информатика-2010 (РИ-2010) XII Санкт-Петербургская Международная Конференция, Санкт-Петербург, 20-22 октября 2010 года: Материалы конференции \ СПОИСУ. - СПб, 2010. С. 199 - 200.

57. Тележкин А.М. Система САМПО+ для создания и анализа исторической базы данных / Приборостроение. – 2014. № 11. – С. 58-62.

58. Иванищев В.В. Моделирование без посредника // Изв. РАН. Теория и системы управления, №5, 1997.

59. Иванищев В.В., Марлей В.Е. Введение в теорию алгоритмических сетей. Санкт-Петербург. Изд-во СПбГТУ, 2000. 179 с.

60. Морозов В.П. Программные системы, ориентированные на привлечение знаний экспертов в процессе решения задач. Программные продукты и системы, №1, 2001. С.33-35.

61. Морозов В.П. Системы управления проектами на основе прозрачных технологий IX Санкт-Петербургская Международная Конференция «Региональная информатика-2004» («РИ-2004»), Санкт-Петербург, 22-24 июня 2004 года: Труды. — СПб.: Наука, 2005. — С.255 - 261.

62. Гарнаев А. Самоучитель VBA. — СПб.: БХВ-Петербург, 2004. — 560 с. — 2е изд.

63. Морозов В. П. Поддержка принятия решений, ориентированная на знания эксперта // Тр. XII Санкт-Петербург. Международная конференция. «Региональная информатика (РИ-2010)», 20—22 окт. 2010 г. СПб: СПОИСУ, 2011. С. 69—73.

64. Паулк М., Куртис Б., Хриссис М. Б., Вебер Ч. В., Гарсия С. М., Буш М. Модель зрелости процессов разработки программного обеспечения – Capability Maturity Model for Software (CMM): Пер. с англ. – М.: Богородский печатник, 2002. – 256 с.

65. Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504-CMM). Пер. с англ. А.С. Агапов, Н.Э. Михайловский, А.А. Мкртумян. – М.: Книга и бизнес, 2001. – 348 с.

66. Volodin M.A. Estimating Procedure. Version 2.5 — SPb.: Exigen Services. StarSoft, 2007. — 8 p.

67. Баранов С.Н., Домарацкий А.Н., Ласточкин Н.К., Морозов В.П. Процесс разработки программных изделий. М: Наука, Физматлит, 2000. — 176 с.

68. Пунтиков Н.И. Метод выбора модели процесса разработки программного изделия на основе формализации описания пространства характеристик проектов. Диссертация на соискание степени кандидата технических наук. Спб.,2007. — 161 с.

69. Баркалов С.А., Воропаев В.И., Секлетова Г.И. и др. Математические основы управления проектами: Учебн. пособие / Под ред. Буркова В.Н. — М.: Высш. шк., 2005. — 423 с.

70. Одинцов И.О. Профессиональное программирование. Системный подход. — СПб.: БХВ-Петербург, 2004. — 624 с.

71. Орлов С.А. Технология разработки программного обеспечения. Учебник для вузов — СПб.: Питер, 2004. — 527 с.

72. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению

73. Липаев В.В. Анализ качества баз данных [Электронный ресурс]. В.В. Липаев // Открытые системы, — 2002. — №03. Режим доступа: http://www.osp.ru/os/2002/03/181272.

74. Горелик А.Л., Скрипкин В.А. Методы распознавания: Учеб.пособие для вузов М.: Высш. шк., 2004. — 261 с.

75. Емельянов С.В., Ларичев О.И. Многокритериальные методы принятия решений — М.: Знание, 1985. — 32 с.

76. Холстед М.Х. Начала науки о программах. М.: Фин. истатистика, 1981. – 128 с.

77. Watson A.H., McCabe, Th.J., Dolores, R. Structured Testing: a Testing Methodology Using the Cyclomatic Complexity Metric // National Institute of Standards and Technology Special Publication 500-235. – September 1996. – 123 p.

78. Capers J. Software Assessments, Benchmarks, and Best Practice. Addison-Wesley, 2000. –339.

Приложения

Приложение 1 Акт об использовании результатов работы в ООО «Ф-Лайн Софтвер»

FIRST LINE SOFTWARE
Think Results.

ООО « Ф-Лайн Софтвер»
197101, ул. Б. Монетная, 16, кор.1 лит.В
Санкт-Петербург, Россия
тел. +7 (812) 336 5533
www.firstlinesoftware.com

Санкт-Петербургский институт информатики и
автоматизации Российской академии наук
В диссертационный совет Д 002.199.01
199178,Санкт-Петербург, 14 линия, 39

Акт внедрения

Комиссия в составе: председателя комиссии Позднякова А.Л., членов комиссии Юсупова В.Р. и Блаера И.Ю. составила настоящий акт о том, что результаты диссертационного исследования Тележкина А.М на тему: «Метод формирования пространства характеристик для оценки ресурсов, обеспечивающих успешное завершение проектов разработки программных изделий», а именно система поддержки создания баз исторических данных «САМПО+», представляют практический интерес и были использованы компанией ООО «Ф-Лайн Софтвер» («First Line Software») для поисков проектов-аналогов, а также в рамках работы по анализу зависимости метрик, собираемых в процессе разработки программного обеспечения.

Применение результатов позволило сформировать пространство характеристик необходимых для поиска проектов-аналогов и оценки на их основе ресурсов потребных для успешного завершения инициируемых проектов, а также снизить риски неудачного их завершения.

Генеральный директор
Председатель комиссии А.Л. Поздняков

Члены комиссии:

Заместитель генерального директора, к.т.н., с.н.с. В.Р. Юсупов

Директор по производству ПО И.Ю.Блаер

Исполнительный директор Д.К.Краснов

"27" августа 2014 г.

Приложение 2 Акт об использовании результатов работы в НП «Объединение подземных строителей»

Саморегулируемая организация
Некоммерческое партнерство

«Объединение строителей подземных сооружений, промышленных и гражданских объектов»
(НП «Объединение подземных строителей»)

СРО-С-064-09112009
ОГРН: 1087800002234 ИНН/КПП: 7810330641/781601001

192102, Санкт-Петербург, ул. Фучика, д. 4, лит.К, пом. 16Н
Тел.:(812)325-05-64,325-05-65, факс: (812)325-05-66
Эл. почта: info@metrotunnel.ru Сайт: www.metrotunnel.ru

19.09.2011 исх. № 357
На № _____ от _____

Санкт-Петербургский институт информатики и автоматизации Российской академии наук
В диссертационный совет Д 002.199.01
199178,Санкт-Петербург, 14 линия, 39

АКТ
о практическом применении полученных результатов диссертационного исследования Тележкина А.М. на тему «Метод формирования пространства характеристик для оценки ресурсов, обеспечивающих успешное завершение проектов разработки программных изделий»

Комиссия в составе:

председателя – генерального директора НП «Объединение подземных строителей» Алпатова С.Н.,

членов комиссии: финансового директора Синаковой С.Н., инженера по АСОИУ Цветкова И.Г.,

составила настоящий акт в том, что в НП «Объединение подземных строителей» изучены результаты диссертационного исследования Тележкина А.М., как практического, так и теоритического характера.

НП «Объединение подземных строителей» является саморегулируемой организацией, основанной на членстве юридических лиц, осуществляющих все виды строительных работ, в том числе строительство особо опасных, технически сложных и уникальных объектов. Сфера саморегулирования и подземного строительства не является темой диссертационного исследования, но результаты проведенного исследования на базе НП «Объединение подземных строителей» признаны актуальными и представляющими практический интерес для дальнейшей деятельности Партнерства.

В частности, метод формирования пространства характеристик был использован при разработке и анализе концепции программной системы для автоматизации ведения реестра членов саморегулируемых организаций «Эскулап». А подмножество характеристик из набора «Метрическое обеспечение программных разработок» было использовано в процессе внедрения системы, что позволило завершить работы в установленные сроки.

Генеральный директор
(должность подписанта)

_____ / Алпатов Сергей Николаевич
(подпись) (расшифровка подписи)

Финансовый директор
(должность подписанта)

/ Синакова Светлана Николаевна

Инженер по АСОИУ
(должность подписанта)

/ Цветков Иван Геннадиевич

М.П.

"19" сентября 2017 г.

Приложение 3 Список метрик, характеризующих программный проект, создаваемый продукт и процесс разработки

Продуктовые метрики характеризуют непосредственно измеряемые свойства самого программного продукта (изделия).

1. Исходные требования – Source requirements. Число исходных требований, общее и по типам (функциональные, интерфейсные, по производительности, системные и т.д.).

2. Размер онтологии – Ontology size. Число разных онтологических сущностей, задействованных в описании исходных требований.

3. Изменчивость требований – Volatility of requirements. Число изменений, добавлений и удалений в формулировках исходных требований, совершенных по инициативе заказчика, абсолютное и в отношении к общему числу требований.

4. Полнота требований – Completeness of requirements. Число требований, добавленных по инициативе разработчиков, абсолютное и в отношении к общему числу требований.

5. Противоречивость требований – Inconsistency of requirements. Число изменений и удалений требований, совершенных по инициативе разработчиков, абсолютное и в отношении к общему числу.

6. Завершенность требований – Finalization of requirements. Число незакрытых вопросов по уточнению исходных требований, общее и по типам требований.

7. Системные компоненты – System components. Число и наименования различных системных компонентов в продукте. Определяется по высокоуровневому проекту и структуре разбиения работ.

8. Программные модули – Software modules. Число и наименования различных программных модулей в продукте. Определяется по компонентам высокоуровневого проекта и структуре разбиения работ.

9. Используемые интерфейсы – Interfaces used. Число и наименования различных интерфейсов, используемых в продукте. Определяется по высокоуровневому проекту и структуре разбиения работ.

10. Входные и выходные каналы – I/O channels. Число и наименования различных входных и выходных каналов в продукте. Соотносится со сложностью по Холстеду [76] высокоуровневого проекта.

11. Используемые языки программирования –Programming languages used. Число и названия языков программирования.

12. Используемые базы данных – Databases used.

13. Используемые серверы приложений – Application servers used. Число и названия таких серверов.

14. Используемые операционные системы – Operating systems used. Число и названия таких операционных систем.

15. Используемые технологии – Technologies used.

16. Размер кода – Code size. Число строк кода на входном языке программирования, не считая пустые строки и строки, состоящие только из комментариев; единица измерения – KLOC или KAELOC.

17. Размер кода для модульного тестирования – Code size for unit testing. Число строк кода на входном языке программирования для модульного тестирования; единица измерения – KLOC или KAELOC.

18. Ветвления в коде – Code branches. Число всех операторов ветвления и цикла в исходном коде.

19. Параллелизм кода – Code parallelism. Число параллельно исполняемых ветвей в исходном коде.

20. Сложность кода по Холстеду – Halstead complexity. Четыре основных

показателя: число различных/всех (η_1/N_1) операторов и число различных/всех (η_2/N_2) операндов в программе и пять производных, определяемых через них: N – длина программы, V – объем программы, L* – оценка длины реализации программы, λ – уровень языка программирования, I – интеллектуальное содержание программы [76].

21. Цикломатическая сложность кода (сложность по Мак-Кейбу) – Cyclomatic complexity (McCabe complexity). Цикломатическое число $\lambda(G) = m – n + 2\times p$, где m – число вершин, n – число дуг и p – число компонентов связности в графе G передач управления в программе[77].

22. Качество кода – Code quality. Оценка числа остаточных дефектов в коде (как известных, так и еще не выявленных), абсолютное и в отношении к общему размеру кода; единица измерения – число/KLOC или число/KAELOC, либо в виде Nσ.

23. Повторное использование кода – Code reuse. Процент повторно использованного кода в общем коде поставки.

24. Размер кода для повторного использования – Code for reuse. Число строк вновь разрабатываемого кода с дополнительной целью его дальнейшего повторного использования; единица измерения – KLOC (KAELOC).

25. Покрытие тестами требований – Requirement coverage with tests. Процент числа требований, покрытых тестами, от общего числа требований, суммарный и по типам требований.

26. Пост-релизные дефекты – Post-release faults. Число дефектов, обнаруженных в продукте после поставки заказчику в течение определенного срока (обычно 6 или 12 месяцев), абсолютное и в отношении к размеру всего поставляемого кода; единица измерения – число/KLOC или число/KAELOC, либо в виде Nσ.

27. Покрытие тестами кода – Code coverage with tests. Процент кода,

покрытого тестами, от всего кода.

28. Покрытие тестами ветвлений в коде – Branch coverage with tests. Процент числа ветвлений, покрытых тестами, от числа ветвлений во всем поставляемом коде.

29. Размер документации – Documentation size. Число страниц в разработанной документации, поставляемой заказчику.

30. Повторное использование документации – Documentation reuse. Процент повторно использованной документации в общем объеме документации, поставляемой заказчику (по числу страниц).

31. Использованные патенты – Patents used. Число и наименования запатентованных решений, использованных в данном продукте.

32. Инновационность продукта – Product innovativeness. Число признанных инновационных решений, использованных в данном продукте, сделанных участниками проекта и реализованных в нем.

33. Пакет поставки – Delivery package. Метрические данные по видам составляющих компонентов пакета поставки, включая исходных код, выполняемый код, документацию, отчеты о дефектах, материалы тестирования, спецификации, планы, процедуры, технологии оценивания, предложения по инновациям и т.д.

34. Обратная связь по продукту – Product feedback. Число предложений, отзывов, рекламаций и т.д., полученных в течение определенного периода (обычно 6 или 12 месяцев) после поставки продукта.

Проектные метрики измеряемым образом характеризуют сам проект по созданию и сопровождению программного продукта.

1. Трудоемкость – Effort. Суммарные трудозатраты, общие и по фазам проекта, по видам работ и категориям исполнителей, по плану и по факту на

данный момент; единица измерения – число человеко-месяцев, человеко-дней или человеко-часов.

2. Производительность труда – Performance. Отношение размера разработанного в данном проекте кода в законченном продукте к общей трудоемкости на его создание, плановое и по факту; единица измерения – KLOC (KAELOC) на человеко-день (человеко-месяц).

3. Длительность проекта – Project duration. Число календарных дней или месяцев, отведенных на разработку, плановое и по факту.

4. Экономия от автоматизации разработки кода – Cost saving due to code development automation. Экономия затрат на разработку за счет автоматизации разработки кода, плановая и по факту; единица измерения – руб.

5. Автоматизация разработки кода – Code development automation. Процент автоматически созданного кода в общем размере кода, плановый и по факту.

6. Автоматизация разработки тестового кода – Test development automation. Процент автоматически созданного тестового кода в общем размере тестового кода (код тестов плюс код тестового окружения), плановый и по факту.

7. Автоматизация прогона тестов – Test run automation. Процент тестов, исполняемых без участия тестировщика, в общем числе тестов в эталонном тестовом наборе, плановый и по факту.

8. Экономия от автоматизации тестирования – Cost saving due to test automation. Экономия затрат на тестирование за счет автоматизации разработки и прогона тестов, плановая и по факту; единица измерения – руб.

9. Методы и инструменты – Methods and tools. Число и наименования различных методов и инструментов, используемых в разработке, включая лицензии на ПО.

10. Стоимость проекта – Project cost. Суммарная текущая стоимость

проекта в денежном выражении, плановая и по факту, общая и по видам затрат, которые включают затраты на заработную плату, оборудование, ПО, специальные компоненты, обучение и переподготовку, книги и журналы, командировки, аренду, услуги сторонних организаций, расходы на юридическое сопровождение, накладные и представительские расходы, расходы на обеспечение секретности, стоимость исправления дефектов и т.д.; единица измерения – руб.

11. Стоимость разработчика – Cost of software developer. Стоимость программной разработки в суммарных затратах на одного участника в месяц, плановая и по факту, включая расходы на оборудование и накладные расходы; единица измерения – руб./месяц.

12. Предел стоимости – NTE (Not to Exceed). Суммарная величина всех затрат на проект, которую нельзя превысить ни при каких обстоятельствах; превышение означает немедленное прекращение проекта; единица измерения – руб.

13. Стоимость строки кода – Cost of a line of code. Отношение суммарных затрат на проект к размеру кода в LOC или AELOC в законченном программном продукте; единица измерения – руб.

14. Частота совершения ошибок – Error rate. Число совершаемых ошибок на единицу размера кода и документации; единица измерения – число/KLOC (KAELOC) для кода и число/стр. для документации.

15. Стоимость обеспечения качества – Cost of quality. Процент суммарных трудозатрат на деятельности по обеспечению качества продукта, включая обзоры, тестирование, повышение квалификации и т.п., в суммарных трудозатратах на проект, плановый и по факту.

16. Стоимость переделок – Cost of poor quality. Процент суммарных трудозатрат на нахождение, исправление и повторную проверку исправления дефектов с учетом трудозатрат на регрессионное тестирование, вызванное исправлениями, в суммарных трудозатратах на проект, плановый и по факту.

17. Разработчики – Developers. Число лиц, создающих код продукта и сопроводительную документацию, плановое и по факту.

18. Распределенность разработки – Distributed (multi-site) development. Число отдельных групп (часто разнесенных по разным площадкам), работающих над проектом, если оно больше единицы.

19. Тестировщики – Testers. Число инженеров, тестирующих продукт, плановое и по факту.

20. Штат проекта – Staffing. Число ставок, включая руководство, разработчиков, тестировщиков и поддержку, плановое и по факту.

21. Женщины – Women. Число женщин, участвующих в проекте, абсолютное и как % от общего числа участников проекта.

22. Запланированные поставки – Planned deliverables. Число и наименования поставок заказчику, включенных в план разработки.

23. Точность поставок – On-time delivery. Процент поставок заказчику, совершенных вовремя, в общем числе всех совершенных поставок, плановый и по факту.

24. Точность планирования – Schedule accuracy. Процент отклонений фактических показателей хода проекта от запланированных в графике проекта в общем числе запланированных показателей.

25. Точность оценивания – Estimation accuracy. Процент отклонений фактических значений параметров проекта от их первоначальных оценок, по каждому параметру в отдельности и усредненный по всем учтенным параметрам.

26. Завершенность разработки – Development completion. Процент объема выполненных работ (по числу завершенных работ в структуре разбиения работ, по трудоемкости, по затратам и т.д.) от запланированного на данный момент, плановый и по факту.

27. Завершенность тестирования – Testing completion. Процент выполненных работ по тестированию (по числу созданных тестов, совершенных тестовых прогонов, успешно прошедших тестов и т.д.) от запланированного на данный момент, плановый и по факту.

28. Перепланирования проекта – Project replannings. Число утвержденных изменений плана в процессе работы, плановое и по факту.

29. Отчеты об ошибках – Defect reports. Число полученных отчетов об ошибках за данный период времени.

30. Плотность выявленных дефектов – Defect density. Отношение числа выявленных дефектов в коде и документе к его размеру; единица измерения – число/KLOC или число/KAELOC для кода и число/стр. для документов, либо в виде Nσ.

31. Эффективность сдерживания дефектов – Defect containment effectiveness. Процент числа дефектов, выявленных и устраненных до сдачи продукта, от сделанной оценки общего числа дефектов.

32. Эффективность сдерживания серьезных (уровень 3 и выше по 5-бальной шкале) дефектов – MFC (Major Fault Containment). Процент серьезных ошибок, не перешедших в дефекты, от общего числа всех выявленных серьезных дефектов, плановое и по факту.

33. Эффективность сдерживания дефектов по фазам проекта – Defect containment effectiveness on project phases. Процент дефектов, выявленных и устраненных на данной фазе, и относящихся к ней, от оценки общего числа дефектов, относящихся к данной фазе.

34. Удовлетворенность заказчика – Customer satisfaction. Число баллов (обычно по 10-бальной шкале, где 10 – высшая оценка), поставленных заказчиком при анкетировании, плановое и по факту.

35. Размер тестового набора – Test suite size. Число тестов в эталонном

тестовом наборе для проверки готовности продукта к выпуску.

36. Процент прохождения тестов – Tests passed. Процент прошедших тестов в общем числе тестов в эталонном тестовом наборе, плановый и по факту.

37. Циклы тестирования – Test cycles. Число полных циклов системного тестирования, совершенных до выпуска продукта, плановое и по факту.

38. Опыт совместной работы данного коллектива – Project team continuity. Среднее время работы одного человека в данном коллективе; единица измерения – число месяцев или лет.

39. Результативность (эффективность) взаимодействия внутри коллектива – Effectiveness of team interaction. Процент числа проблем, разрешенных внутри коллектива, в общем числе проблем, поднятых в ходе исполнения проекта.

40. Рациональность (экономичность) взаимодействия внутри коллектива – Efficiency of team interaction. Средние затраты на разрешение одной проблемы внутри коллектива разработчиков (число взаимодействий, длительность процесса разрешения, понесенные трудозатраты и т.д.); единица измерения – в зависимости от способа.

41. Опыт работы с данной платформой – Platform experience. Срок работы участников проекта с данной платформой, суммарный по всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

42. Опыт работы в данной предметной области – Subject domain experience. Срок работы участников проекта в данной предметной области, суммарный по всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

43. Опыт работы в данной методологии и с данными инструментальными средствами – Method and tools experience. Срок работы участников проекта в данной методологии и с данными инструментальными средствами, суммарный по

всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

44. Время жизни дефекта – Defect longevity. Среднее и максимальное время от момента обнаружения дефекта до сообщения о его закрытии по всем выявленным и закрытым дефектам, плановое и по факту; единица измерения – число дней.

45. Процент незакрытых дефектов – Still open defects. Процент числа еще не закрытых дефектов в общем числе обнаруженных дефектов на данный момент времени.

46. Результативность (эффективность) обзоров – Review effectiveness. Процент числа дефектов, выявленных на обзорах, в общем числе выявленных дефектов на данный момент времени.

47. Рациональность (экономичность) обзоров – Review efficiency. Отношение суммарных трудозатрат на подготовку и проведение обзоров к общему числу найденных на обзорах дефектов – т.е., средняя стоимость нахождения одного дефекта путем обзора; единица измерения – человеко-час.

48. Инновационность проекта – Project innovativeness. Число рацпредложений и заявок на изобретение, поступивших от участников проекта по его тематике в ходе выполнения проекта и в течение некоторого срока (обычно до 6 месяцев) после его завершения.

49. Публикационность проекта – Project publicity. Число публикаций, включая защиты диссертаций, сделанных участниками проекта по его тематике в ходе выполнения проекта и в течение некоторого срока (обычно до 2 лет) после его завершения. Учитываются как открытые публикации, так и публикации для служебного пользования.

50. Загруженность оборудования – Equipment usage (load). Процент времени использования оборудования в суммарном времени, в течение которого

оно находится в распоряжении участников проекта (по видам оборудования и лицензиям на ПО), плановый и по факту.

51. Использование уникального оборудования – Unique equipment usage. Суммарное время, потраченное на настройку и подготовку уникального оборудования организации или заказчика и прогон на нем создаваемого продукта, плановое и по факту; единица измерения – число человеко-месяцев, человеко-дней или человеко-часов.

52. Проектные риски – Project risks. Общее число различных проектных рисков, включенных в план управления рисками.

53. Сработавшие риски – Triggered risks. Число актуализировавшихся проектных рисков, абсолютное и в отношении к общему числу проектных рисков, включенных в план управления рисками.

54. Переоценка рисков – Risk reassessment. Число пересмотров плана управления рисками (сколько раз производился пересмотр рисков по их перечню и характеристикам), плановое и по факту.

55. Смягчение последствий рисков – Risk mitigation. Трудозатраты на предотвращение и смягчение последствий рисков, плановые и по факту.

56. Воздействие рисков – Risk impact. Убытки в денежном выражении, которые проект может понести при срабатывании всех рисков и каждого в отдельности, плановые и по факту.

57. Декомпозиция проекта на продукты – Project decomposition to products. Число самостоятельных продуктов, появляющихся в результате данного проекта, если оно больше единицы.

58. Декомпозиция проекта на функциональные подгруппы – Project decomposition to functional subgroups. Число функциональных групп, на которое в итоге был разбит проект.

59. Компьютеры – Computers. Число компьютеров (настольных и др., включая тестовые и сервера), выделенных для данной группы.

60. Достаточность ресурсов – Sufficiency of resources. Процент необходимых по плану ресурсов в фактическом их объеме по видам ресурсов и усредненный по всем видам ресурсов.

61. Постоянство разработчиков – Staff continuity. Процент лиц, работавших от начала до конца проекта, в общем числе разработчиков.

62. Интенсивность взаимодействия внутри организации – Intensity of interaction within the organization. Число человеко-часов потраченных на общение внутри организации. Подсчитывается взаимодействие между разработчиками, между разработчиками и тестировщиками, между руководством проекта и командой.

63. Интенсивность взаимодействия с заказчиком – Intensity of interaction with customer. Число человеко-часов потраченных на общение с заказчиком.

64. Результативность (эффективность) тестирования – Effectiveness of testing. Число дефектов, выявленных путем тестирования, абсолютное и в отношении к общему числу выявленных дефектов.

65. Рациональность (экономичность) тестирования – Efficiency of testing. Трудозатраты на нахождение дефектов путем тестирования, абсолютные и в отношении к общему числу выявленных дефектов.

66. Метрики – Metrics. Число и наименования используемых в проекте метрик.

Процессные метрики характеризуют применяемый в создании продукта технологический процесс разработки и, соответственно, уровень зрелости организации, в которой этот процесс установлен.

1. Уровень зрелости разработчиков – Maturity level. Определяется путем

официального оценивания или самооценивания; единица измерения – номер уровня по модели CMMI.

2. Повышение квалификации – Training. Среднее число часов для одного сотрудника на обучение и переподготовку в течение года, плановое и фактическое.

3. Выполненные разработки (проекты) – Completed projects. Число и наименования разработок (проектов), выполненных в организации за последние несколько лет. Определяется по БВП организации.

4. База заказчиков – Customer base. Число и наименования разных заказчиков, для которых выполнялись разработки за последние несколько лет. Определяется по БВП организации.

5. Портфель патентов – Patent portfolio. Число и наименования патентов, полученных данной организацией за пять несколько лет.

6. Текучесть кадров – Attrition rate. Процент сотрудников, покинувших организацию, в общем числе сотрудников в расчете на год.

7. Нарушения процесса – Process violations. Число зафиксированных нарушений процесса разработки за период по их категориям.

8. Улучшения процесса – Process improvements. Число поданных предложений по улучшению процесса за период.

9. Выработка ресурса – Burn-down chart. Число человеко-часов, остающихся на выполнение данной задачи; в методике Scrum наглядно отображает "сгорание" выделенного ресурса на выполнение задач по всему проекту в целом и по каждому спринту в отдельности.

10. Процессные активы – Project assets. Число шаблонов документов, моделей жизненного цикла, программных и тестовых наработок и других процессных активов, накопленных в организации.

11. Используемость процессных активов – Project assets usage. Процент повторно использованных процессных активов из базы процессных активов организации в общем числе процессных активов по отдельным активам и их группам.

12. Опытность команды – Staff experience. Процент ведущих опытных разработчиков в общем числе разработчиков.