Федеральное государственное бюджетное образовательное учреждение

Ульяновский государственный технический университет

На правах рукописи

Лапшов Юрий Александрович

СРЕДСТВА ПРОГРАММНО-КАРТОТЕЧНОГО УПРАВЛЕНИЯ ПОТОКАМИ РАБОТ В КОЛЛЕКТИВНОМ ПРОЕКТИРОВАНИИ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

Специальность: 05.13.12 –

Системы автоматизации проектирования (промышленность)

ДИССЕРТАЦИЯ

На соискание ученой степени кандидата технических наук

Научный руководитель: д.т.н., профессор

Соснин Петр Иванович

Ульяновск - 2015

Обозначения и сокращения

АС – Автоматизированная система

БД – База данных

БНФ – Бекус-Наурова форма

КСР – Корпоративная среда разработки

ПО – Программное обеспечение

ПКУ – Программно-картотечное управление

РБНФ – Расширенная Бекус-Наурова форма

УПР – Управление потоками работ

A – Answer (Ответ)

Q – Question (Вопрос)

QA – Question-Answer (Вопросно-ответный)

RUP – Rational Unified Process (Рациональный унифицированный процесс)

SIS – Software Intensive Systems (системы, интенсивно использующие программное обеспечение)

UML – Unified Modeling Language (Унифицированный язык моделирования)

Введение

Основные проблемы расширяющейся компьютеризации всех сфер человеческой деятельности проявляются в разработках систем, интенсивно использующих программное обеспечение (Software Intensive Systems, SIS). К числу этих проблем относится чрезвычайно низкая степень успешности разработки SIS, которая за последние 20 лет с «трудом» приблизилась к 35 % (всемирно признанные статистические исследования и отчёты корпорации Standish Group и других исследователей).

В отчётах, регистрирующих положение дел с успешностью разработок, фиксируются не только оценки успешности для разнообразных условий создания SIS, но и факторы, способствующие и препятствующие успеху разработок. В перечнях этих факторах (среди особо важных) устойчиво занимает «управление коллективной и персональной деятельностью проектировщиков», используемое в процессах проектирования SIS.

Необходимость переосмысления основ программной инженерии и внесения в них существенных изменений (а именно создание программных составляющих SIS, согласованных с остальными её компонентами приводит к проблемам с успешностью) привела к инициативе SEMAT[73] (Software Engineering Methods And Theory – Методы и Теория Программной Инженерии, доступно на сайте http://www.semat.org), в спецификации и осуществлении которой (начиная с 2009 года) приняли участие ведущие мировые учёные и специалисты в области программной инженерии (Ivar Jacobson,Watts S. Humphrey, Scott W. Ambler, Alistair Cockburn и другие).

В нормативных документах SEMAT «техника работы» (way-of-working), используемая командой проектировщиков, отмечена как очень важная сущность. В нормативах SEMAT «техника работы» определена как «адаптированный к специфике исполняемого проекта набор методов и инструментов, используемых командой проектировщиков в процессе

проектирования», причём, адаптация подразумевает не только настройку методов и инструментов, выбранных командой, исполняющих проект, но и создание членами команды новых методов, оказавшихся им необходимыми.

Особое отношение к техникам работы не может не затрагивать вопросы управления коллективной и персональной деятельностью проектировщиков SIS, а значит и автоматизированных систем (АС), которые относятся к классу SIS. Вышесказанное содержит достаточно оснований, чтобы считать создание инновационных методов и средств управления коллективной и персональной деятельностью проектировщиков АС актуальным. К этому направлению инноваций относится и предлагаемый в диссертации комплекс средств программно-картотечного управления процессами проектирования автоматизированных систем.

То, что «техники работы» имеют алгоритмическую природу, привело в диссертации к идее представлять «техники работы» и их связные совокупности (потоки работ) в виде программ на алгоритмическом языке, подобном естественному языку в его алгоритмическом употреблении для планирования персонального и коллективного поведения проектировщиков. Программное представление потоков работ и их составляющих открывает возможность для применения в их построении и использовании как методов и средств эмпирической программной инженерии, так и моделей опыта из других предметных областей, в первую очередь из арсенала Искусственного Интеллекта.

В документах SEMAT предлагается команде проектировщиков начинать разработку очередного проекта с адаптации к его специфике «ядра SEMAT», дополняя его «как пазлами» тем, что способствует реализации проекта. В диссертационном исследовании в число таких пазлов, дополнительных к программному управлению, было решено включить средства визуально-картотечного управления, нашедшего свое материальное воплощение в

различных версиях технологий KANBAN и SCRUM. Более того, интересы исследования было решено ограничить деятельностью проектировщиков в процессах концептуального проектировании АС, которые считаются наиболее проблематичными с позиций их влияния на успешность.

В диссертационной работе роль **области исследований** возложена на средства управление потоками работ в концептуальном проектировании АС.

Направление исследований в диссертации связано с инструментально-технологическими средствами управления потоками работ гибкой разработки программного обеспечения, которые специально введены в процесс разработки АС для координирования потоков проектных работ.

Функции **объекта исследований** в диссертации выполняют средства координирования потоков проектных работ, использующие гибкие методологии разработки.

Роль **предмета исследования** в диссертации возложена на методы и средства координирования потоков работ участника проектного процесса, такие, как управление очередями задач и прерываниями, а также - методы и средства координирования потоков работ команды проектировщиков, такие, как Kanban, Scrum.

Целью исследования является совершенствование процессов управления потоками работ, способствующее предотвращению ошибок, обусловленных человеческим фактором, и обеспечивающее сокращение непроизводительных затрат времени в оперативной индивидуальной и коллективной работе проектировщиков за счет включения в совокупность средств управления проектной деятельностью дополнительных программируемых составляющих

Задачи диссертационного исследования

1.Проанализировать возможность и обосновать целесообразность включения в состав средств управления персональной и коллективной

активностью проектировщиков автоматизированных систем (АС) дополнительных технологических средств программного управления работами проектировщиков, с позиций исполнения каждым из них роли «специализированного процессора».

2.Разработать специализированную систему псевдо-кодового программирования, ориентированную на создание и исполнение программ персональных и коллективных действий проектировщиков, включающих действия по оперативному использованию доступного опыта и его моделей, способствующих повышению эффективности управления процессами проектирования.

3.Разработать программно-картотечную систему оперативного распределения проектных задач, планирования их параллельного исполнения, картотечной визуализации состояния, оценки результативности работ и анализа их исполнения для оптимизации очередных шагов проектирования.

4. Разработать методы и средства для создания и использования библиотек запрограммированных прецедентов, упрощающих создание средств программного управления проектной деятельностью, учитывающих специфику конкретной разрабатываемой АС.

5.Разработать методологическое обеспечение и систему инструментально технологических средств, обслуживающих программно-картотечное управление персональными и коллективными действиями проектировщиков.

6.Разработать прототип технологии программно-картотечного управления и провести его испытания в лабораторных условиях.

На научную новизну претендуют:

1. Программно-картотечная модель гибкого управления потоками работ, ориентированная на использование механизмов Kanban и Scrum в проектировании АС, специфику которой определяет визуализация очередей задач, учитывающая их распределение в коллективе проектировщиков и во

времени и открывающая возможность эффективного распараллеливания решений проектных задач в условиях управления их прерываниями;

2. Подмножество концептуально-алгоритмического языка, к особенностям которого относится определение его данных и операторов над семантической памятью, обеспечивающее оперативное программное управление очередями задач за счет полезных метрик планирования работ и продуктивности групп проектировщиков.

3. Совокупность методик, включающих методики отображения потоков работ и их исполнителей на вопросно-ответную память, оперативного планирования, картотечной визуализации и гибкого управления по образцам Kanban и Scrum с учетом детализации очередей, позволяющей программно управлять шагами их исполнения.

4. Библиотека псевдокодовых программ гибкого управления и программных моделей шаблонов типовых потоков работ, настроенная на повторное использование и включающая дополнительные разделы процедур и функций, написанных на языках Microsoft .NET Framework.

В число **методов исследований** включены методы: объектно-ориентированного анализа, проектирования и программирования; теории и практики автоматизированного проектирования; когнитивного анализа задач и принятия решений; массового обслуживания.

Достоверность полученных результатов обеспечивается использованием достоверных знаний, методов и средств из логики, прикладной информатики и программной инженерии. Практический вклад в достоверность подтверждается разработкой комплекса средств, обеспечивающего программно-картотечное управление процессами персональной и коллективной проектной деятельности в создании автоматизированных систем.

Практическую ценность работы составляет разработанный набор

средств, обеспечивающий реализацию координирования потоков работ в проектировании АС, библиотека шаблонов потоков работ, представленный комплекс методик по координированию потоков работ в процессе проектирования АС.

Личный вклад соискателя ученой степени

Все результаты проведенных исследований и положения, выносимые на защиту, были получены автором самостоятельно. Соавторами совместных публикаций являются научный руководитель, д.т.н., проф. Соснин П.И., который принимал участие в выборе направления исследований, постановке задачи и обсуждении полученных результатов, а также - сотрудники, которые принимали участие в научно – исследовательских программах. В публикации с соавторами вклад соискателя определяется рамками представленных в диссертации результатов.

Реализация и внедрение результатов работы

Разработанные программные средства и комплекс методик их использования внедрены на ФНПЦ ОАО НПО «МАРС», проведен эксперимент по оценке эффективности использования предлагаемых средств.

Грант №15-07-04809 «Технологии и инструментарий ПКУ процессами в проектировании систем, интенсивно использующих ПО»

Свидетельства о регистрации программы для ЭВМ: №2015610593 «Компилятор псевдокодовых программ». №2015610236 «Программный инструментарий контроля поручений»

Апробация работы

Основные положения и результаты диссертационной работы докладывались и обсуждались на следующих конференциях:

8-th international conference «Interactive Systems: Problems of Human – Computer Interaction», региональная научно-техническая конференция «Информатика и вычислительная техника, 2009», всероссийская конференция

«Проведение научных исследований в области обработки, хранения, передачи и защиты информации, 2009», российская школа-семинар «Информатика, моделирование, автоматизация проектирования», 2009, российская конференция «Информатика и вычислительная техника», 2010, Научно-техническая конференция УлГТУ, 2010, российская школа-семинар «Информатика, моделирование, автоматизация проектирования», 2010, российская конференция «Информатика и вычислительная техника», 2011, российская школа-семинар «Информатика, моделирование, автоматизация проектирования, 2011», 9-th international conference «Interactive Systems: Problems of Human – Computer Interaction», российская конференция «Информатика и вычислительная техника», 2012, российская школа-семинар «Информатика, моделирование, автоматизация проектирования», 2012, International conference «Open Semantic Technologies for Intelligent Systems» (OSTIS-2013), 10-th international conference «Interactive Systems: Problems of Human – Computer Interaction», The 27th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, 2014.

В первой главе диссертационной работы раскрываются вопросы, связанные с особенностями коллективного проектирования АС, человеко-компьютерным взаимодействием и дизайном, ориентированным на пользователя, обосновывается необходимость управления потоками работ в проектировании АС; обосновывается необходимость учета особенностей коллективной и персональной деятельности во время реализации АС. Упоминаются вопросы профессиональной зрелости процессов разработки АС. Приводится обзор методов и средств координирования потоков проектных работ на ранних стадиях разработки АС; приводится общая схема области интересов человеко-компьютерного взаимодействия в привязке к проектированию АС; приводятся аргументы, доказывающие целесообразность координирования потоков проектных работ в

проектировании АС; формулируется задача исследований в обобщенной форме и детализированной в виде вопросно-ответного анализа; проводится мотивационно-целевая оценка диссертационной работы.

Во второй главе приводится формализация и спецификация процессов ПКУ. Эта работа была начата с построения архитектурной модели средств ПКУ в среде WIQA. В качестве инструмента формализации здесь и далее будет использоваться ***РБНФ***-грамматика.

В третьей главе диссертации проводится сценарная структуризация средств ПКУ, позволившая выделить ***акторов*** управления проектным процессом, а также - ***набор задач ПКУ***. В третьей главе также был рассмотрен ряд методик выполнения работ программно-картотечного управления, связанных с поручениями, управлением очередями потоков работ, параллелизмом в процессе выполнения задач сотрудником, а также с вычислением метрических показателей проектного процесса.

В четвертой главе раскрываются вопросы, связанные с реализацией комплекса средств ПКУ. Также проводятся экспериментальные исследования и нагрузочное тестирование программно-картотечного управления. В главе приведены особенности адаптации разработанного инструментария ПКУ к применяемой методологии проектирования.

В заключении раскрываются научные и практические результаты диссертационной работы с позиции их эффективности и новизны.

В приложении рассмотрено псевдо-кодовое программирование базовых шаблонов потоков работ.

Глава первая. Управление потоками работ в оперативном коллективном проектировании АС.

1.1. Место и роль управления потоками работ в оперативном коллективном проектировании автоматизированных систем.

Как отмечалось во введении, в настоящее время существует проблема успешной разработки систем[65], интенсивно использующих программное обеспечение. На наличие этой проблемы явно указывает статистика успешности, исследуемая корпорацией **Standish Group**и публикуемая в ее регулярных отчетах с 1994 года].

Интегральное представление статистики с 1994 по 2011 годы приведено в таблице 1.1, из которого следует, что положение дел, которое можно оценить, как крайне неудовлетворительное, изменяется в лучшую сторону, но процесс улучшений нельзя признать управляемым (провал, зарегистрированный в 2008-м году).

Таблица 1.1

Доля в процентах										Результат
16	27	26	28	34	29	35	32	37	39	Успех
31	40	28	23	15	18	19	24	21	18	Провал
53	33	46	49	51	53	48	44	42	43	Изменения
199	199	199	200	200	200	200	200	201	201	

Следует отметить, что разработка ***SIS*** в статистике относится к классу успешных, если в ней за запланированные время и стоимость качественно материализована затребованная функциональность. От оценок экспертов и потребителей статистика абстрагируется.

В отчетах **Standish Croup**[133] и других публикациях по вопросам успешности не только констатируется положение дел, но и приводятся факторы, оказывающие позитивное и негативное воздействие на успешность разработок ***SIS.***

Так, например, в таблице 1.2 приведены позитивы (и указаны их приоритеты), которые в отчетах за 1994, 2008 и 2012 годы отнесены к факторам, способствующим достижению успеха.

Таблица 1.2

Фактор	1994	2008	2012
Вовлеченность пользователей	1	1	2
Поддержка руководителей высшего звена	2	2	1
Понятные формулировки требований	3		
Подходящее планирование	4		
Реалистичные ожидания	5		
Частые контрольные точки	6		
Компетентный персонал	7	8	4
Права собственности	8		
Ясное видение и цели	9	3	7
Напряженная работа/Нацеленный персонал	10		
Управление проектом		**7**	**5**
Стандартный инструментарий и инфраструктура		10	10
Гибкость работ с требованиями		6	6
Оптимизация масштаба/Оптимизация		**5**	**3**
Эмоциональная зрелость		4	8
Нормативное исполнение		**9**	**9**

Отсутствие единой позиции по факторам, способствующим успеху и препятствующим его достижению, объясняется, в основном, разнообразием типологии проектов и условий, в которых они разрабатываются. Единым в оценках положения дел остается то, что проблема успешности разработок ***SIS*** сохраняется, несмотря на многое, что для ее решения предложено и сделано.

Развитие теории и практики *разработок сложных* ***SIS не доведено*** *до состояний, позволяющих создателям* ***SIS достаточно верно оценивать будущую успешность и управляемо продвигаться к успеху*** *и это определяет* ***сущность проблемы успешности****, на негативы которой указывают отчеты* ***Standish Group****.*

Недопустимые для человечества потери, обусловленные проблемой успешности, требуют, как можно быстрее найти выход из положения дел, сложившегося в разработках ***SIS.*** Это особенно важно и потому, что именно это направление деятельности определяет глобальную информатизацию человеческой жизни.

Отметим, что результативность любых попыток, нацеленных на решение проблемы успешности, в существенной мере зависит от того, насколько адекватно и измеримо определены понятия «**успеха**» и «**успешности разработки *SIS»*** и насколько конструктивны механизмы **оценивания успешности** и **управления разработкой *SIS*** на основе **оцениваний**.

Необходимым, но недостаточным условием «успеха» проектной организации считается постоянное совершенствование процессов, коллектива и продуктов. Более конкретные спецификации «успеха» за счет совершенствования указаны в стандарте[83] **ISO/MEK 9004–2009,** обобщённое представление содержания которого образно отражено на рисунке 1.1.

«Устойчивый успех организации достигается за счет ее способности отвечать потребностям и ожиданиям своих потребителей и других заинтересованных сторон на долговременной основе и сбалансированным образом. Устойчивого успеха можно добиться посредством эффективного менеджмента организации, путем осознания организацией среды своего существования, за счет обучения и должного применения улучшений и (или) инноваций»[83].

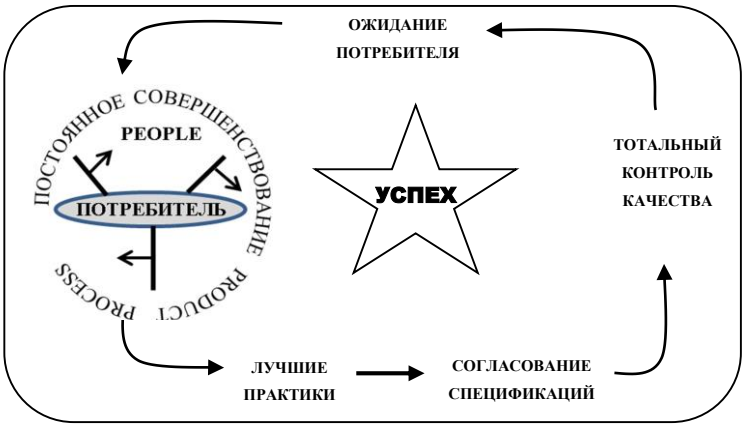


Рис. 1.1. Схема совершенствование производственной деятельности

Стандарт ориентирован на широкий круг организаций, включая проектные. Но из-за своей широты он раскрывает пути достижения успеха обобщенно и в рекомендательном плане, что не умаляет его значимости для практики.

Как уже отмечалось, для обеспечения в разработках ***SIS*** управляемой нацеленности на успех необходимо определиться с понятием «успех» конструктивно.

> *Слово «**успех**» принято употреблять в качестве **оценочной характеристики**, которую приписывают **результату деятельности** по **степени** его **востребованности «потребителями»**. Такую характеристику принято переносить и на **исполнителя** деятельности, приведшей к **успешному результату**, и на саму эту **деятельность**.*

В статистических исследованиях успешности разработок ***SIS*** ***конкретную*** разработку считают «успешной», если она завершена в соответствии с запланированными требованиями в рамках оговоренных финансовых затрат и времени. Реалии востребованности ***SIS потребителями*** в такой оценке учитываются опосредованно – через соответствие того, что вложено в ***SIS*** согласно запланированным (при заключении контракта на разработку) функциональными нефункциональным требованиям. Будет реальный успех ***SIS*** на рынке или нет, в такой версии оценок не присутствует и не прогнозируется.

Возможны и другие версии приписывания «успеху» значений, например, через характеристики проектной деятельности и её исполнителей, в первую очередь с позиций их профессиональной зрелости.

Одним из направлений повышения успешности разработок программных систем и систем, интенсивно использующих программное обеспечение, является совершенствование профессиональной зрелости[66][69] осуществляемых процессов.

В таком совершенствовании различают «уровни профессиональной зрелости»[71], с каждым из которых связывают систему «практик», в которой определенный набор «практик» доведен до определенной «степени совершенства». Под «практиками» понимают определенные виды технологических работ, которые в разработке следует выполнять обязательно, действуя в соответствии с подходящими методиками, а «степень совершенства» связывают с использованием «лучших практик» (best practices).

В оценках профессиональной зрелости[115][116] конкретной системы процессов абстрагируются от исполнителей работ, полагая, что их компетентность достаточна.

*Для **характеристики и оценки профессиональной зрелости исполнителей** принято использовать их «**компетентность**», уровням которой приписывают определенную «**квалификацию**».*

Системы процессов разработки осуществляются, и их профессиональная зрелость[30] повышается в определенных коллективах при определенных условиях, например, в проектных организациях, что позволяет говорить о «профессиональной зрелости проектной организации» и оценивать ее в соответствии с достигнутым уровнем «профессиональной зрелости процессов»[64].

***Уровень зрелости** производственного процесса – это степень, до которой тот или иной процесс **определен, управляем, измеряем, контролируем и эффективен**.*

Уровень зрелости является интегральной характеристикой, значения которой приписываются в результате экспертной оценки, в которой учитываются экспертные оценки степени определенности, степени управляемости, степени измеряемости, степени контролируемости степени эффективности[66].

Так, например, в степени определенности принято выделять два значения:

- Первое значение («повторяемый процесс») отражает сам факт повторяемости совокупности действий разработчиков от одного проекта к другому;
- Второе значение («определяемый процесс») указывает на то, доведена ли повторяемая совокупность действий до технологии с соответствующим инструментарием.

В степени управления процессом также принято выделять два значения «управляемый процесс» и «количественно управляемый процесс», вкладывая во второе значение **оперативную управляемость** по **количественно измеряемым отклонениям** характеристик процесса от плановых значений.

В экспертных оценках профессиональной зрелости[30] встает вопрос об **эталонах** для приписывания значений. Функции таких эталонов принято возлагать на «определенные совокупности работ, которые выполняются нормативно», то есть по определенным технологическим методикам. Каждая такая методика интерпретируется как «измеряемая» метрика (эталон исполнения работы)[102].

Отметим, что ничто не мешает расширить содержание «профессиональной зрелости проектной организации», включив в него, например, составляющую, которая интегрально представляет «компетенции» ее коллектива, и характеристику «полезности инструментария», используемого в организации.

1.2. Обзор родственных исследований и разработок

Исследования в данной диссертационной работе выполнены в соответствии с несколькими тематическими направлениями, связанными моделированием и программно-контролируемым исполнением потоков работ разработки SIS.

1.2.1. Особенности проектной деятельности

Феномен «управление» был создан природой в процессе эволюции жизни на Земле для решения вполне определённых задач, в число основных из которых входят задачи саморегулирования (в первую очередь, самосохранения) и самоорганизации (в том числе саморазвития) живой «материи». Феномен проявляет себя через взаимодействие «живого организма» и/или их совокупности с окружающей средой, причём, в условиях происходящих в среде изменений.

В реальных задачах управления информационные потоки структурируются в переменных, характеризующих воздействия, взаимодействия и состояния объекта управления и управляющего субъекта, что образно отражено на рисунке 1.2, причём с каждым набором таких переменных принято связывать их векторные интерпретации. Так, например, составляющие вектора A – это набор основных целевых ориентиров (характеристик) управления.

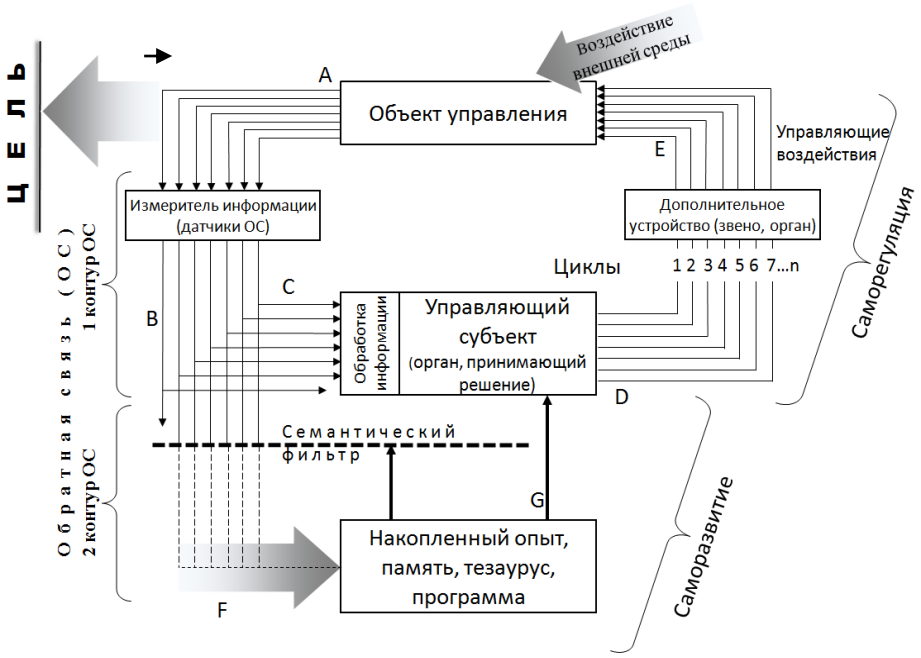


Рис. 1.2. Обобщённая схема управления

В принципе, ничто не мешает применять обобщённую схему врешении задач управления проектами, нацеленных на успешность разработок. В таких применениях принципиальна «ЦЕЛЬ», связанная с повышением степени успешности создания проектов за счёт совершенствования профессиональной зрелости[30] процессов проектирования и профессиональной зрелости лиц, вовлечённых в такие процессы[23].

Особенности проектной деятельности (в общем смысле) обусловлены тем содержанием, которое вкладывается в то, что называют проектами. Приведём ряд определений проекта:

1. ***Проект*** – это **временное предприятие**, предназначенное для создания уникальных продуктов, услуг или результатов (PMBOK до пятой версии).

2. ***Проект*** – это **комплекс взаимосвязанных мероприятий,** направленный на создание уникального продукту или услуги в условиях временных или ресурсных ограничений[5] (ГОСТ Р 54869–2011).

3. ***Проект*** – это **уникальная совокупность процессов**, состоящая из контролируемых и управляемых видов деятельностей с датами начала и

завершения, предназначенная для достижения определенных целей[82] (ISO 21500:2012).

Ориентация на «цель» позволяет связать его принципиальную особенность «уникальность» с уникальностью «совокупности процессов, состоящей из контролируемых и управляемых видов деятельностей». Другими словами, стандарт ISO 21500 указывает, что «проектное управление» можно и даже рекомендуется осуществлять на основе средств, в которых используется «процессное управление». Но динамика проектной деятельности обязательно приведёт к уникальной версии комплексирования деятельностных единиц, управляемых процессно. Схема типовой процессной единицы приведена на рисунке 1.3.

.

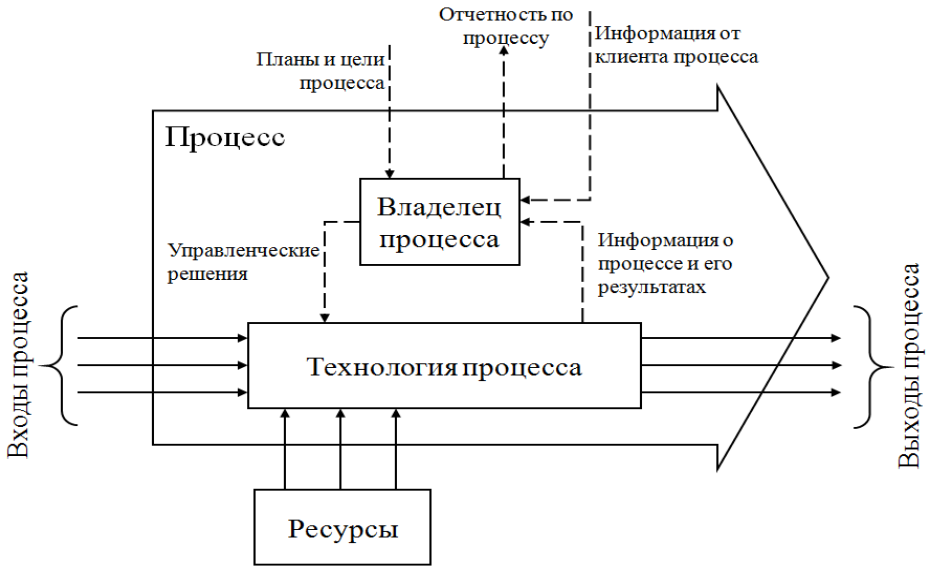


Рис. 1.3. Обобщённая схема процесса

Таким образом, полезно понимать, что:

За осуществлением проекта стоит построение и реализация уникального процесса, приводящего к достижению определённой цели или целей.

А значит, для проектирования следует обязательно использовать технологию и подходящий инструментарий, в основу которых положены

«лучшие практики», в число которых входят и практики, управляемые процессно. Замечательным примером такого подхода является технология и инструментарий «Rational Unified Process» (RUP), созданные корпорацией IBM [8].

Отметим, что на схеме процесса приведён один контур управления, но это следует понимать как обобщение. На самом деле, типичной формой управления процессами является двухконтурное управление, в котором управляющий субъект, формируя управляющее воздействие, действует по определённой программе (алгоритму), которую он «не имеет права изменять». В общем случае двухконтурного управления, субъект адаптирует или совершенствует «программу» своих действий по формированию управляющих воздействий на объект управления.

Напомним, что в стандарте ISO 21500 в определении «проекта»[82] отсутствует акцент на «временное предприятие». Но два последних рисунка подсказывают, что в организациях, деятельность которых структурируется в «программах» и «портфелях», например, в организациях осуществляющих разработки семейств ***SIS***, для каждого очередного проекта приходится внутри организаций создавать аналог «временного предприятия». Приходится на время, запланированное под каждый проект, создавать команду разработчиков проекта, выделять необходимые ресурсы и осуществлять организационное управление работой проектировщиков в условиях определённых ограничений.

Завершая пункт, отметим, что исполнение программ и выполнение обязательств по портфелям открывают дополнительные направления по управлению проектной деятельностью, которые находят своё нормативное выражения в ряде стандартов[5][82], например, в стандартах ISO 21500 и ГОСТ Р 54869-2011.

1.2.2. Структура проектного управления

В соответствии с любым из определений «проекта» он имеет

определённую структуру, схематическое изображение которой зависит отточки зрения на проект как систему. Другими словами, для представления проекта можно использовать различные «block and line» модели, выражающие тот или иной аспект его системности[103].

Любая системная точка зрения на проект должна отражать его окружение. Подробную концептуальную информацию об окружении «проекта» даёт схема[5] из ГОСТ Р 54869-2011, приведённая на рисунке 1.4.

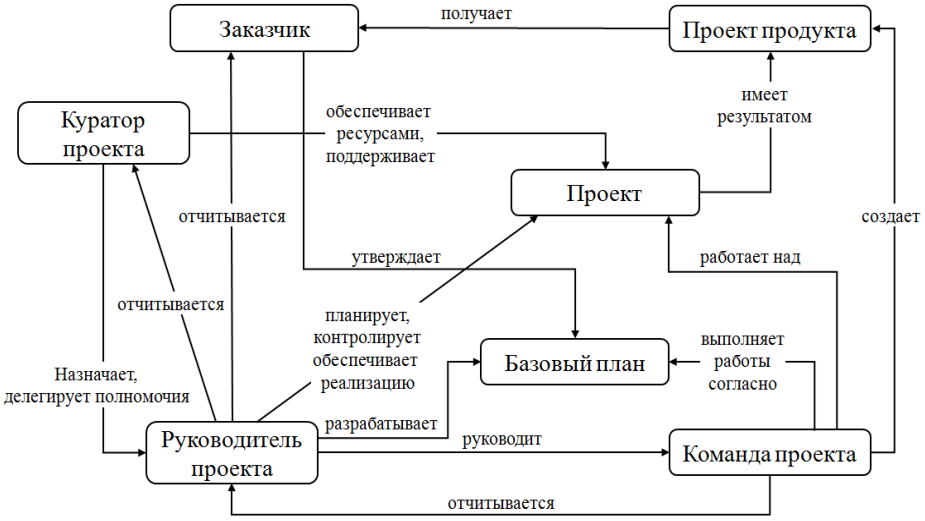


Рис. 1.4. Окружение проекта

Эта схема согласована с определением «проекта» как «**комплекса взаимосвязанных мероприятий**, направленных на создание уникального продукту или услуги»

Для обобщённой демонстрации видов групп процессов), раскрывающих деятельностную структуру проекта, принято использовать схему, приведённую на рисунке 1.5, на которой только частично отражена динамика комплексирования указанных групп.

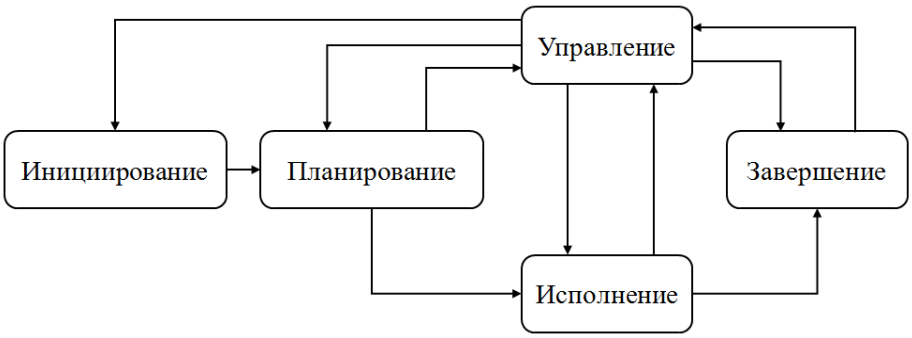


Рис. 1.5. Взаимодействие групп процессов

Обобщённое представления динамики комплексирования составляющих проектной деятельности отражает диаграмма, приведённая на рисунке 1.6.

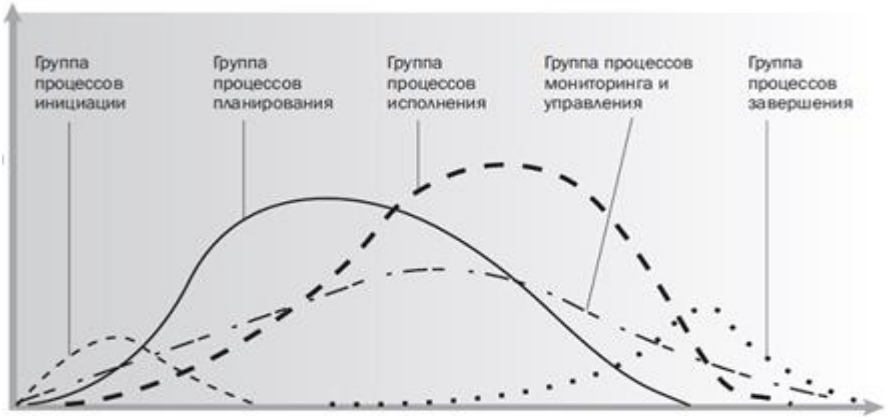


Рис. 1.6.Динамика групп процессов проекта

Диаграмма отражает тот факт, что в реальном проектировании процессы групп приходится распараллеливать[104].

1.2.3. Стандартизация знаний о проектном управлении

Самым известным стандартом проектного управления считается стандарт PMBOK, который изначально и во всех своих версий (до PMBOK 5:2012) создавался как «свод» знаний о проектном управлении, или, как часто говорят, как «энциклопедия» об этой предметной области. Содержимое этой «энциклопедии» структурировано в виде, представленном в таблице 1.3.

Таблица 1.3.

Области знаний	Группы процессов				
	Инициация	Планирование	Исполнение	Мониторинги контроль	Завершение
Интеграция					
Содержание					
Сроки					
Стоимость					
Качество					
Персонал					
Коммуникации					
Риски					
Покупки					
Заинтересованные стороны					

Для каждого процесса в документах PMBOK указаны обобщённые спецификации его входных данных (но не шаблоны данных), спецификации их обработки (связанные с методами, показавшими свою эффективность) и результатов обработки (спецификации выходных данных).

Энциклопедичность PMBOK[74] используется разработчиками прикладных технологий и других стандартов как источник полезных решений, потенциальных процессов и практик или для сравнений выбранных решений с PMBOK (например, в виде «Наша технология и PMBOK»).

1.2.4. Потоки работ

При решении задач, связанных с бизнес-процессами, за которыми стоят определенные деятельностные образования, бизнес-процессы моделируют, применяя различные полезные средства, чаще всего диаграммные модели и соответствующие им формальные языки диаграмматики. К числу таких средств, например, относятся диаграммные средства структурного анализа

SADT[31] и унифицированного моделирования UML.

Наиболее полезным и популярным переходом от бизнес-процессов реальности к их представлению в модельной действительности является моделирование бизнес-процессов и их совокупностей в форме потоков работ (workflows)[40].

Определяя потоки работ и строя их как модели, используют следующие точки зрения[76]:

- функциональную, фокусирующую внимание на «исполняемых активностях» как на сущностях (работах), которые «перетекают» от одной единицы к другой по определенной схеме;
- поведенческую, акцентирующую внимание на том, когда и в каких условиях деятельностные единицы исполняются (особое внимание сосредотачивается на управлении процессами);
- структурную, раскрывающую моделируемый бизнес-процесс с системной точки зрения, причем, учитывая как объекты деятельности, так и деятельностные процессы.

Для определения специфики потоков работ в проектной деятельности вернемся к проблеме успешности разработок ***SIS*** с позиций стандарта **ISO/MEK 9004–2009** «Менеджмент для достижения устойчивого успеха организации: Подход на основе менеджмента качества» в его части, требующей явно осуществлять и совершенствовать «**Согласование спецификаций**» для используемых проектировщиками «лучших» практик (рисунок 1.2).

В проектной деятельности, нацеленной на разработку ***SIS***, принципиальное место занимает **фронт работ** с **требованиями**[40] к проектируемой ***SIS***. В общем случае требования поступают в проект из различных источников. Они, например, предъявляются заказчиками ***SIS***,

выявляются проектировщиками или диктуются условиями эксплуатации будущей системы.

В технологии, например, типа RUP, за этот фронт работ отвечает группа нормативных практик[86], применение которых в определенные моменты времени приводит к необходимости, **согласованной спецификации требований.**

Проведем анализ того, **кому** с **кем**, **что** с **чем, каким образом** и в **каких формах** приходится согласовывать для достижения согласованной спецификации системы требований к разрабатываемой ***SIS***.[19][41]

Так, например, в концептуальном проектировании[100] $\boldsymbol{D^{CD}(SIS)}$, применяя задачи технологии $\boldsymbol{\{Z^N_j\}}$, необходимо заново или повторно (с учетом полезных модификаций) решить каждую из предметных задач $\boldsymbol{Z^D_i}$ любого бизнес-процесса ***БП*** будущей ***SIS*** .

В такой работе бизнес-процессы разрабатываемой ***SIS*** и бизнес-процессы технологии ***T*** представляют с помощью моделей типа «**поток работ**», так что от проектировщиков требуется согласованно решить группы задач $\boldsymbol{\{Z^S_{mi}\}}$ **для каждого из потоков работ** $\boldsymbol{\{W^m\}}$ создаваемой ***SIS***, исполняя при этом группы задач $\boldsymbol{\{Z^N_{nj}\}}$, объединенные в потоки работ $\boldsymbol{\{W^n\}}$ технологии $\boldsymbol{T(\{W^n\})}$. Более того, обычно потоки работ ***SIS*** **образуют группы**, а потоки работ технологии **объединены в группы** $\boldsymbol{G^r(\{W^n\})}$ для обслуживания этапов проектирования.

Так как исполнение потоков работ и их групп предполагает согласованное решение групп задач, с каждым потоком работ $\mathbf{W^n}$ и с каждой группой $\mathbf{G^r(\{W^n\})}$ потоков работ можно связать **составную задачу** $\mathbf{Z^W_n}$ и $\mathbf{Z^G_r}$, приписав ей тип **W** или **G**. Следовательно, в наиболее общем плане, приходится согласовывать решения задач из множеств $\mathbf{\{Z^S_{im}\}}$ и $\mathbf{\{Z^N_{in}\}}$, а также решения задач из множеств $\mathbf{\{Z^W_m\}}$, $\mathbf{\{Z^W_n\}}$ и $\mathbf{\{Z^G_r(\{Z^W_n\})\}}$.

Фронт работ «согласований», за которым стоят задачи типа $\mathbf{Z^C}$, разнородный, сложный и очень важный. По этой причине «лучшие практики»

согласований включают в систему практик технологий. К числу таких практик относится, например, комплекс практик построения **архитектуры *SIS***, обслуживающий согласование требований и их спецификаций на стратегическом уровне разработки.

Однако нормативно предусмотреть все, что приходится делать для согласования, невозможно[110]. Так, например, невозможно предусмотреть «что придется делать» для согласования каждой конкретной из предметных **задач типа Z^S с задачами из множества $\{Z^N_{nj}\}$.**

Таким образом, в работе с очередной предметной задачей $\mathbf{Z^S_{mi}}$, то есть решая ее, проектировщик должен перенести ее содержание в используемые им нормативные задачи технологии, формулируя и решая подходящие задачи адаптации Z^A_q. Заметим, что задачи адаптации $\{Z^A_q\}$ образуют подкласс Z^A класса задач согласования Z^C.

Заметим, что отсутствие в технологии практик, ответственных за задачи адаптации, не является следствием того, что такую активность проектировщиков нельзя автоматизировать.

Для того чтобы **адаптация**[40] к задаче $\mathbf{Z^S_{mi}}$ оказалась успешной, проектировщик, в первую очередь, должен **понять ее содержание**. Поэтому **достижение адекватного понимания** является основной составляющей адаптации, которая при переносе понятого проектировщиком в нормативные задачи **закрепляется и регистрируется в нормативных задачах**. В результате **понимание,** вложенное в используемые задачи технологии, становится доступным для проверок и использования другими проектировщиками, а также лицами, заинтересованными в успешности разработки ***SIS***.

Ориентируясь на достижение согласованного понимания, можно обобщенно ответить на вопросы «**кто?**» и «**с кем?**» осуществляет согласование требований и их спецификаций.

Во-первых, согласование осуществляет **проектировщик сам с собой**, а вернее, проектировщик **M**, решающий назначенную ему задачу $\mathbf{Z^{S}_{mi}}$со своим **персональным опытом**[29] **и его моделями**, если они имеются. В таком согласовании проектировщик выступает в двух ролях – «**специалист, решающий задачу**» и «**субъект, являющийся носителем определенного опыта**».

Во-вторых, проектировщик осуществляет согласование с другими членами команды проектировщиков и доступными им моделями опыта, разумеется, если в этом имеется необходимость.

В-третьих, – проектировщик осуществляет согласование с пользователями и другими лицами, заинтересованными в успешности проекта.

Отметим, что во всех версиях **персонификации согласований**, причем как для процесса согласований, так и для проверки его результатов, используются **доступный опыт** и **понимание**.

1.2.5. Гибкое проектное управление

Технология RUP относится к классу «тяжеловесных» технологий, которые представляют их пользователям большое разнообразие практик и их композиций, что приводит к необходимости настройки технологии как на специфику семейства проектов, так и на специфику проектной организации.

Для «тяжеловесной» технологии характерна сложная структура потоков работ, в осуществление которых вовлечены многие тысячи экземпляров из сотен типовых практик.

Одним из технологических направлений в разработках программного обеспечения или систем типа SIS, которое активно и конструктивно развивается последние годы является создание и использование «гибких технологий»[139], в основе которых лежат Agile-подходы, а также методы и средства их материализации (Agile– эджайл – подвижный, проворный, быстрый, верткий, живой)[67][93].

Одно из определений гибкой методологии и технологии разработки говорит, что это «серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля».

Названное существенно отличается от систему требований к дисциплинированным проектным технологиям с проектным управлением, о котором говорилось в п. 1.2 и 1.3.Следует отметить, что ничто не мешает использовать (где и когда это полезно) или дисциплинированные технологии, или гибкие технологии, или их композиции.

На текущий момент времени разработан ряд гибких технологий[93][139], доля каждой из которых в применениях отражена на рисунке 1.7. Рисунок приведён, в основном, для того чтобы указать на доминирующее использование на практике средств Scrum и их модификаций. За одной из модификаций стоит Agile-подход, получивший название Kanban.

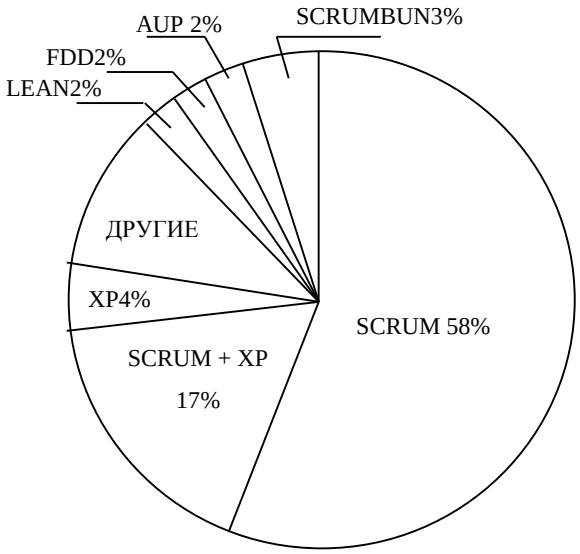


Рис. 1.7. Востребованность гибких технологий

Разработанные авторами средства программно-картотечного управления потоками работ построены с использованием Kanban и Scrum[57][117]-

подходов. По этой причине ниже приведено введение в два этих подхода.

Изначально методология и комплекс средств Kanban создавались как система организации непрерывного потока производства без создания запасов[57]. Это так называемая система “точно в срок”[136], в результате работы которой продукты производства поставляются небольшими партиями прямо к необходимым пунктам процесса производства, не попадая на склад. При этом готовая продукция сразу перенаправляется потребителям.

Потом эта система была адаптирована к решению задач управления в разработках программного обеспечения. На текущий момент времени накоплен богатейший опыт применения системы Kanban для управления деятельностью в различных областях, в том числе и для управления проектной деятельностью в проектировании SIS и их семейств. Системы управления по образцу Kanban относятся к Agile-средствам, которые существенным образом отличаются от традиционных средств проектного управления.

Scrum – это наиболее востребованный на практике Agile-подход[117]. Изначально он разрабатывался для создания программного обеспечения, в том числе и для разработок SIS. В то же время этот подход применим для инструментальной поддержки коллективной разработки сложных продуктов.

Как и в Kanban-подходе команде проектировщиков, применяющих Scrum, доступен фронт работ[117], например, в форме потока задач или в форме пользовательских историй (Бэклог проекта), из которого они отбирают определенную совокупность единиц работ (Бэклог спринта) для их согласованного исполнения (Спринт) на определённом промежутке времени.

Работы в рамках спринтов проводят в соответствии со следующим набором правил[117]:

• правила, по которым должен планироваться и управляться список требований к продукту;

• правила планирования итераций;

• основные правила взаимодействия участников команды;

• правила анализа и корректировки процесса разработки.

Принципиальное место в Scrum-процессах занимает экспериментирование, по результатам которого выявляются производительность команды и другие характеристики коллективной работы.

Одной из особенностей Scrum-процесса являют регулярные (зачастую ежедневные) совещания, во время которых идет обсуждение результатов предыдущих спринтов (итераций) и ставятся задачи на следующие. Это позволяет команде держать под контролем процесс разработки и, при необходимости, направляя его в нужное русло.

Scrum-процессы принято визуализировать по образцу визуальной доски Kanban. Более того, эти два подхода принято интегрировать в единую систему ScrumBan[118].

1.2.6. Прерывания в деятельности человека

В современных автоматизированных системах обычно поддерживается многозадачность, в рамках которой общая работа распределяется между человеком либо группой лиц и компьютерными программами. В определенных условиях человек может выполнять несколько действий одновременно, однако человеческие возможности ограничены[63], и попытки одновременного выполнения нескольких задач могут привести к ошибкам. Эти ограничения также существенно влияют на эффективность работы пользователя[55][59]. Если пользовательский интерфейс спроектирован с учетом особенностей человеческой деятельности, то пользователь может в определенный промежуток времени успешно выполнить несколько задач, переключаясь между ними. Разработке таких интерфейсов препятствует отсутствие единой теории о природе прерываний в процессе человеко-компьютерного взаимодействия.

Когда же пользовательский интерфейс спроектирован без учета этих

человеческих ограничений, попытки одновременного выполнения нескольких задач могут привести к значительному снижению производительности работы пользователя и появлению ошибок в решении критически важных задач[9].

При разработке системы обработки прерываний необходимо четко представлять, что же такое прерывание[11]. Можно дать несколько определений термина «прерывания»[15].

Во-первых, МакФарлайн дал общее определение человеческого прерывания[98] следующее: «Человеческое прерывание есть процесс координации быстрой перемены человеческой деятельности».

Из этого определения следует, что прерывание – это:

- процесс
- координация
- внезапность
- изменение деятельности человека.

Во-первых, прерывание – это процесс. Поэтому любое определение прерывания должно ссылаться на весь процесс прерывания[95].

Во-вторых, прерывание – это координация. Человеческое прерывание – это сложный процесс, состоящий из многих поддействий, связанных между собой связями, и этими связями необходимо управлять[97].

В-третьих, прерывание является неожиданным.

В-четвертых, прерывание – это изменение деятельности. Человек обычно выполняет какую-то другую задачу перед прерыванием.

В зависимости от источника прерываний, прерывания могут быть

следующих видов[17][60]:

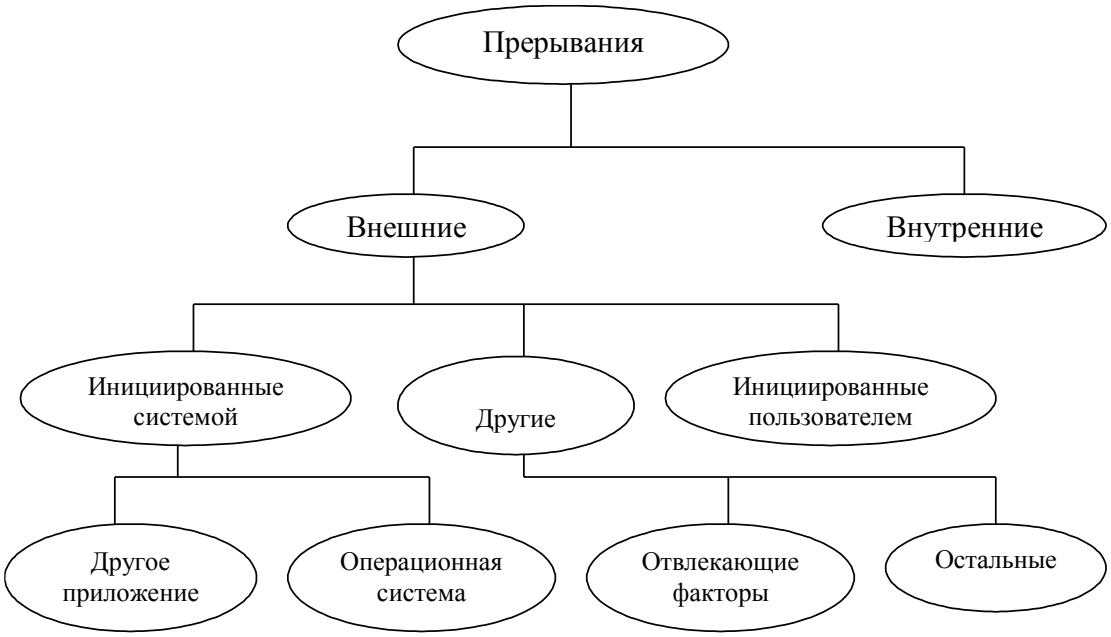


Рис. 1.8. Виды прерываний в зависимости от источника

Управление потоком прерываний обеспечивает более эффективную работу пользователя. Для того, чтобы эффективность работы не снизилась, необходимо предусмотреть восстановление контекста прерванной задачи. Следует разделить обработку прерываний на три этапа[97]:

1. Во время фазы пре-прерывания пользователь готовится перейти от текущей задачи к прерывающей задаче.
2. Вторая фаза обычно включает переход к прерывающей задаче, а также характеризуется тем, что пользователь прилагает усилия, направленные на прерванную задачу, работающую в фоновом режиме, работая при этом с задачей, которая инициировала прерывание.
3. Пост-прерывание включает в себя возвращение к прерванной задаче и соответствующее переключение контекстов.

Пре-прерывание

Перед тем, как происходит прерывание, пользовательский интерфейс должен обеспечить поддержку пользователя для быстрого запоминания текущей задачи перед переключением контекста, которое является частью

обработки прерывания. Джиль и Броадбент пришли к выводу, что припоминание может с успехом применяться в пользовательских интерфейсах для обработки прерываний. Детвейлер описывает два эксперимента, изучающих эффект от раннего предупреждения о прерываниях[97]. Эксперименты показали, что предупреждения крайне полезны в том случае, если прерывающая задача требует малой загрузки памяти и является непохожей на прерываемую задачу.

Средняя фаза прерывания (Mid-Interruption)

При наступлении прерывания интерфейс должен обеспечить поддержку пользователя при переключении контекстов, а также поддерживать внимание пользователя для наблюдения фоновых задач. Такое переключение контекстов может принимать различные формы. Макфарлейн провел эксперимент[97] для сравнения четырех основных подходов к проблеме координации человеческих прерываний в человеко-компьютерных интерфейсах. Четыре метода прерываний:

1) прервать немедленно и сразу переключить;
2) обеспечить поддержку предупреждения, чтобы пользователь мог контролировать время и другие параметры между переключением задач;
3) передать управление прерываниями от пользователя к интеллектуальному агенту, который бы решал, прерывать пользователя или нет в зависимости от его поведения (прерывание через посредника;
4) прерывание по расписанию, то есть прерывание может случиться только в определенные промежутки времени.

Пост-прерывание

После того, как прерывание произошло и пользователь завершил работу с задачей, которая инициировала прерывание, происходит возвращение к

исходной, прерванной задаче. Пользовательский интерфейс должен обеспечивать поддержку восстановления. То есть должны существовать механизмы, которые помогали бы пользователю восстановить контекст прерванной задачи, сокращая тем самым время, необходимое для восстановления. Мэйлин говорит о том, что пользовательский интерфейс должен быть спроектирован таким образом, чтобы пользователь быстро мог переориентироваться на работу с прерванным приложением, когда он попытается восстановить его. В его работе для этих целей предлагалось ведение журнала последних совершенных действий.

Из четырех методов, предложенных Макфарлейном[95][96][97], прерывание с предупреждением было признано лучшим способом для всех уровней производительности пользователя, за исключением случаев, где задержки в обработке прерываний являются критическими для всей работы. Вообще выбор того или иного метода прерывания зависит от того, какие цели преследует проектировщик интерфейса. Макфарлейн для различных целей проектирования интерфейса отвечает на вопрос, какой метод в данной ситуации является лучшим, а какой худшим:

Таблица 1.4. Методы прерываний для различных целей проектирования

Цель проектирования интерфейса	**Лучший метод**	**Худший метод**
Безошибочное выполнение текущей задачи	С предупреждением	По расписанию
Высокая производительность текущей задачи	С предупреждением / Через посредника	Мгновенное / По расписанию
Меньшее количество переключения задач	По расписанию	Мгновенное
Безошибочное выполнение прерывающей задачи	Не мгновенное	Мгновенное
Завершенность выполнения прерывающей задачи	Мгновенное / Через посредника	По расписанию / С предупреждением
Обязательное переключение на прерывающую задачу	Мгновенное	По расписанию / С предупреждением
Высокая производительность прерывающей задачи	С предупреждением / По расписанию	Немедленное
Высокая производительность всех задач	С предупреждением / По расписанию	Немедленное
Обработка прерываний по выбору пользователя	С предупреждением / Через посредника	Немедленное / По расписанию
Контроль пользователем своего	Не немедленное	Не «с предупреждением»

внимания для текущей задачи		
Пользователь не ожидает прерывания	С предупреждением / Через посредника	Немедленное / По расписанию
Пользователь знает об ожидаемом прерывании	По расписанию / С предупреждением	Немедленное / Через посредника
Пользователь контролирует и осознает сложность текущей задачи при получении прерывания	С предупреждением / Через посредника	Немедленное / По расписанию

Макфарлейном разработана стратегия прерываний, с помощью которой выбирается соответствующий метод прерывания пользователя, основываясь на значении приоритета текущего приложения и приоритета прерывающего приложения. Данная стратегия для трехзначной системы приоритетов может быть отображена в виде следующей таблицы[97]:

Таблица 1.5. Стратегия для трехзначной системы приоритетов задач.

Приоритет прерывающей задачи	**Приоритет текущей задачи**		
	Высокий	**Средний**	**Низкий**
Высокий	С предупреждением, по умолчанию поместить в очередь	С предупреждением, по умолчанию прервать	Немедленное прерывание
Средний	С предупреждением, по умолчанию поместить прерывание в очередь	С предупреждением, по умолчанию поместить в очередь	С предупреждением, по умолчанию прервать
Низкий	Отложить прерывание до переключения задач	С предупреждением, по умолчанию поместить прерывание в очередь	С предупреждением, по умолчанию поместить прерывание в очередь

При реализации системы обработки прерываний автором данной магистерской работы использовался следующий подход к обработке прерываний. При прерывании текущей технологической задачи данные о ней заносятся в очередь прерываний. Если пользователем установлено, что он занят, то ранее прерванные задачи не восстанавливаются, а пользователь сам решает, какие действия ему совершать далее. Если же пользователь свободен, то из очереди по одной начиная со второй восстанавливаться прерванные ранее задачи. Восстановление заканчивается последней прерванной задачей.

При наступлении какого-то внешнего события (телефонный звонок, в

дверь вошел посетитель) пользователь прерывает свою работу на компьютере, то есть происходит внешнее прерывание. Соответственно пользователь просто прерывает текущую задачу. При этом прерванная задача заносится в стек прерванных задач.

Закончив с внешним прерыванием, пользователь может восстановить прерванную задачу, выбрав ее в списке прерванных задач[49][50]. При этом пользователь столкнется с рядом проблем, аналогичных проблемам, возникающим при внутреннем прерывании. Главной из этих проблем, как уже ранее оговаривалось, является то, что при возвращении к прерванной задаче человек не всегда помнит те действия, которые он производил с ней[112]. Система обработки прерываний обязана учесть этот момент[114].

1.3. Платформа для средств ПКУ потоками работ

1.3.1. Опыт разработки АС и комплекс WIQA

В коллективе исследуются и разрабатываются подходы, методы и средства, в основе которых лежит оперативное использование в проектной деятельности опыта, доступного проектировщикам. В такой работе коллектив ориентируется на вопросно-ответные процессы формирования и применения доступного опыта, в состав которого входят личный и коллективный опыт проектировщиков, а также модели опыта, в том числе вложенного в технологии и инструменты[44].

В качестве платформы для разработки средств ПКУ потокам работ был выбран комплекс ***WIQA***. Как уже было отмечено, изначально комплекс ***WIQA*** разрабатывался для инструментально-технологического сопровождения процессов решения задач в концептуальном проектировании ***АС***[45]. Этот этап разработки особо критичен к дорогостоящим ошибкам и промахам, причём, основной причиной их появления является проблемы с пониманием, которое в любых своих проявлениях, строится с использованием

взаимодействий с доступным опытом.

Решая любую задачу, проектировщик вкладывает в свою работу и в её результаты определённое понимание[43], конструктивное представление которого должно выводить на те единицы опыта, которые привели к решению. А значит, если решение задачи, которая назначена проектировщику, будет им конструктивно представляться и интерпретироваться в единицах опыта, связанных с решением, то это будет способствовать пониманию решения самим проектировщиком и коллегами по совместной работе, и, следовательно, будет способствовать предотвращению дорогостоящих семантических ошибок и их обнаружению[120][121].

Исходя из такой установки был задуман комплекс ***WIQA*** и разработан ряд его версий, последними из которых являются комплекс ***WIQA.Net*** для коллективной работы и комплекс ***OwnWIQA*** для индивидуальной работы. У комплексов единая концептуально-методологическая основа, что позволяет представить их архитектурно просто как комплекс ***WIQA***. Ниже, там, где не избежать особенностей реализации и/или особенностей эти версии комплекса ***WIQA*** будут называться собственными именами.

1.3.2. Особенности комплекса WIQA

Наиболее общим видом задач, с которым связаны интересы данной диссертационной работы, является задача **Z*** разработки очередной ***AC*** из их семейства. За этой задачей стоят все остальные задачи проекта (как технологические, таки предметные), отношения подчиненности между которыми фиксируются деревом задач.

Особое место в комплексе ***WIQA*** занимает та его часть, которая предназначена для отображения процесса решения задачи **Z*** и его результата, то есть проекта ***AC*** на семантическую память вопросно-ответного типа, или другими словами, на ***QA-память***. К особенностям QA-памяти относится то, что она специфицирована и материализована для обслуживания

взаимодействий проектировщиков с доступным опытом[127].

За учёт особенностей проектирования ***AC*** в отображениях ***Z**** на QA-память отвечает вопросно-ответная модель этой задачи, которая построена на основе анализа типологии потоков работ и задач в потоках технологии RUP, а также анализа моделей задач из других источников.

В построениях QA-модели задачи ***Z**** были использованы рекомендации стандарта IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Этот стандарт рекомендует использовать в создании ***SIS*** (а значит и ***AC***) интегральную совокупность архитектурных видов. Так как решение задачи ***Z**** представляет соответствующий проект **AC**, то применение рекомендаций стандарта **IEEE 1471** правомерно[124].

Применение рекомендаций стандарта **IEEE 1471** к информационной базе, полученной в результате анализа ***RUP*** и других источников привело к архитектурной модели задачи ***Z****, приведённой на рисунке 1.n.

Отметим, что центральное место в архитектуре и в модели занимают «Задачный вид» и «Логико-лингвистический вид»[123]. «Задачный вид» введен в модель для того чтобы представить иерархию подчиненных задач как для задачи ***Z****, так и для любой другой задачи Z_i из дерева задач $T(Z^*, t) = T(\{Z_i\}, t)$ в его состоянии на текущий момент времени **t** процесса разработки ***AC***.

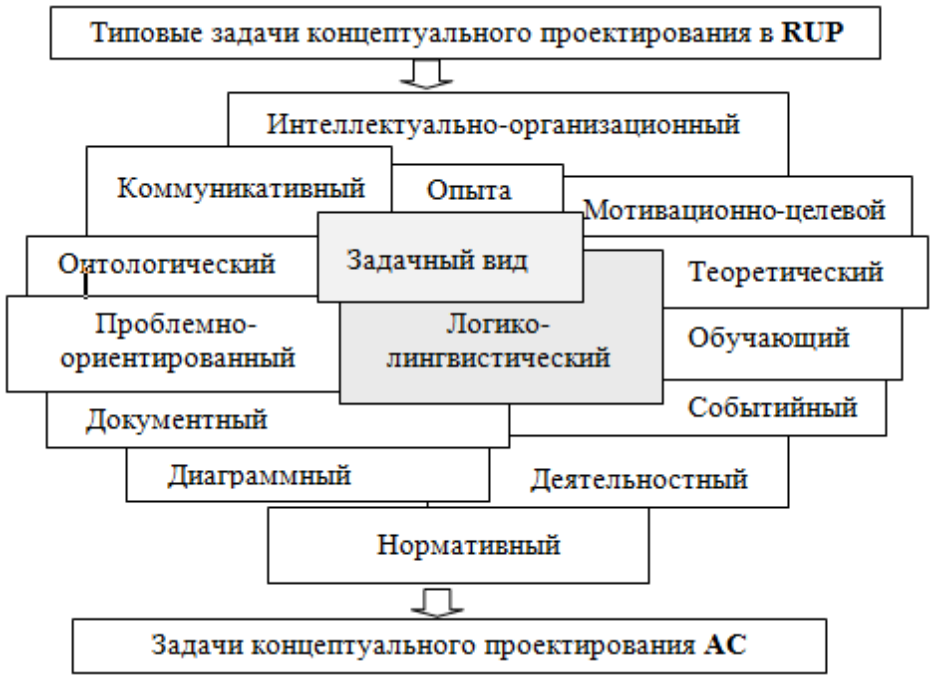


Рис. 1.9. Архитектура типовой **QA**-модели

«Логико-лингвистический вид» фиксирует (в вопросно-ответной форме) рассуждения проектировщика, которые привели его к построению концептуального решения задачи $\mathbf{Z_i}$. Задачный и логико-лингвистический виды являются «родственниками», поскольку то, что называют «задачами» является определённым видом вопросов как феноменов человеческого существования. Одной из версий ответов на «вопрос-задачу» является совокупность действий по ее решению.

«Задачный» и **«логико-лингвистический»** виды определяют «**ядро**» интегральной системы видов $\mathbf{QA(Z_i)}$[127], из-за чего такое образование и названо **QA**-моделью. Остальные виды развивают модель, настраиваясь над их содержанием и открывая новые направления работ по построению решения и его повторному использованию:

- «интеллектуально-организационный вид» раскрывает назначение ответственных за решение задачи $\mathbf{Z_i}$ и ее подзадач $\mathbf{\{Z_{ij}\}}$;
- «деятельностный вид» фиксирует динамические отношения процессов решения совокупности$\mathbf{Z_i \cup \{Z_{ij}\}}$ с использованием «потоков работ»;

- «коммуникативный вид» выделяет коммуникативные задачи, которые пришлось решать в работе с задачей $\mathbf{Z_i}$;
- остальные виды, как и коммуникативный вид, выводят на соответствующие группы задач, решение которых способствует построению и использованию решения задачи $\mathbf{Z_i}$.

Инструментарий ***WIQA*** обеспечивает их информационно-процедурное овеществление в форме, интерактивно доступной проектировщикам для использования модели $\mathbf{QA(Z_i)}$ по назначению.

В схеме архитектуры отражен тот факт, что модель построена на базе задач технологии **RUP**, вернее, технология **RUP** использовалась как важнейший источник требований к реализации **QA**-моделей. Отметим, что типовая **QA**-модель, в первую очередь, представляет собой связанную совокупность образцов (артефактов, моделей, процедур), которая для каждой задачи и любой из ее подзадач шаг за шагом (пошаговая детализации + итерации) наполняется информационным содержанием, извлекаемым из реальных рассуждений разработчиков **AC**. Специфика типа задачи учитывается в настройке архитектурных видов экземпляра модели.

Как было отмечено выше, за формирование и использование QA-модели задачи $\boldsymbol{Z^*_i}$ конкретного проекта $\boldsymbol{AC_i}$ отвечает только часть комплекса $\boldsymbol{WIQA^K}$– его ядро (**kernel**), которое можно виртуально объединить с$\mathbf{QA(Z^*_i)}$. Образованный таким образом конструкт ($\boldsymbol{WIQA^K}$, $\mathbf{QA(Z^*_i)}$) представляет собой специализированную $\boldsymbol{AC^{WIQA}}$, обеспечивающую создание концептуального проекта для $\boldsymbol{AC_i}$.

Так что типовая модель[7][10], приведённая на рисунке 1.9, определяет требования к комплексу ядра ***WIQA***, в соответствии с которыми он был построен в первых своих версиях и развит до его текущих версий. Для версии ***WIQA.Net*** её компонентный состав приведён на рисунке 1.10.

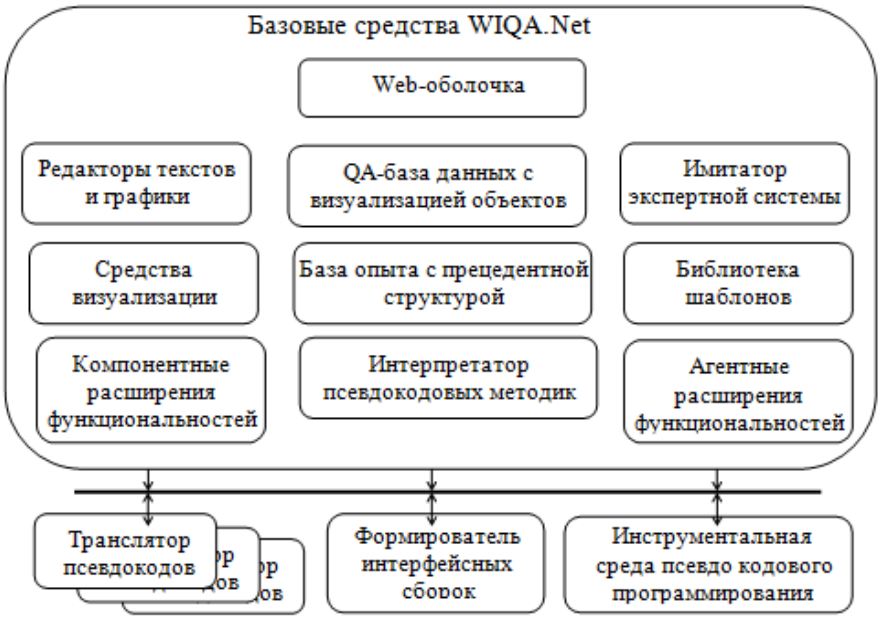


Рис. 1.10. Компонентный состав WIQA.Net

Возможны две версии размещения компонентов – с сохранением клиент-серверной структуры и без. Для однопользовательской версии ***OwnWIQA*** все функциональности сервера и клиента размещены на одном компьютере и оптимизированы. ***Web***-оболочка, разработанная только для версии ***WIQA.Net***, позволяет строить его комплектацию, в которой клиенты связаны с сервером через интранет. Такая возможность предусмотрена для подключения к сети ***WIQA.Net*** рабочих мест, управляемых операционными системами отличными от ***Microsoft Windows***.

Одной из специфик комплексов ***WIQA.Net*** и ***OwnWIQA*** является поддержка псевдокодового программирования[7][12][90][91], ориентированного на прецеденты. Такая возможность, во-первых, позволяет расширять функционал комплекса ***WIQA***, а, во-вторых, позволяет строить на основе этого комплекса приложения.

1.3.3. Место средств ПКУ потоками проектных работ в WIQA

Инструментарий ***WIQA*** предполагает решение ряда задач проектирования, представленных на рисунке 1.11. Комплекс средств ПКУ

потоками проектных работ отвечает за решение задачи гибкого проектного управления.

Таблица 1.2

Фактор	1994	2008	2012
Вовлеченность пользователей	1	1	2
Поддержка руководителей высшего звена	2	2	1
Понятные формулировки требований	3		
Подходящее планирование	4		
Реалистичные ожидания	5		
Частые контрольные точки	6		
Компетентный персонал	7	8	4
Права собственности	8		
Ясное видение и цели	9	5	7
Напряженная работа/Нацеленный персонал	10		
Управление проектом		7	5
Стандартный инструментарий и инфраструктура		10	10
Гибкость работ с требованиями		6	6
Оптимизация масштаба/Оптимизация		**5**	**3**
Эмоциональная зрелость		4	8
Нормативное исполнение		**9**	**9**

Рис. 1.11. Задачи проектирования WIQA

Инструментарий ***WIQA*** включает в себя, как уже было сказано выше, проекты, представленные в виде наборов ***QA***-моделей проектных задач, а также представляет коллектив проектировщиков в виде организационной структуры. ***QA***-модель задачи детализирует её содержание, позволяя говорить о ней как о потоке работ по решению этой задачи. В процессе формирования набора задач этапа проектной работы происходит ассоциация между проектировщиками и решаемыми ими проектными задачами. При этом происходит формирование фронта работ коллектива. Каждый участник проектного процесса получает для решения в определенный период времени заданный набор задач, которые формируют очередь задач для решения.

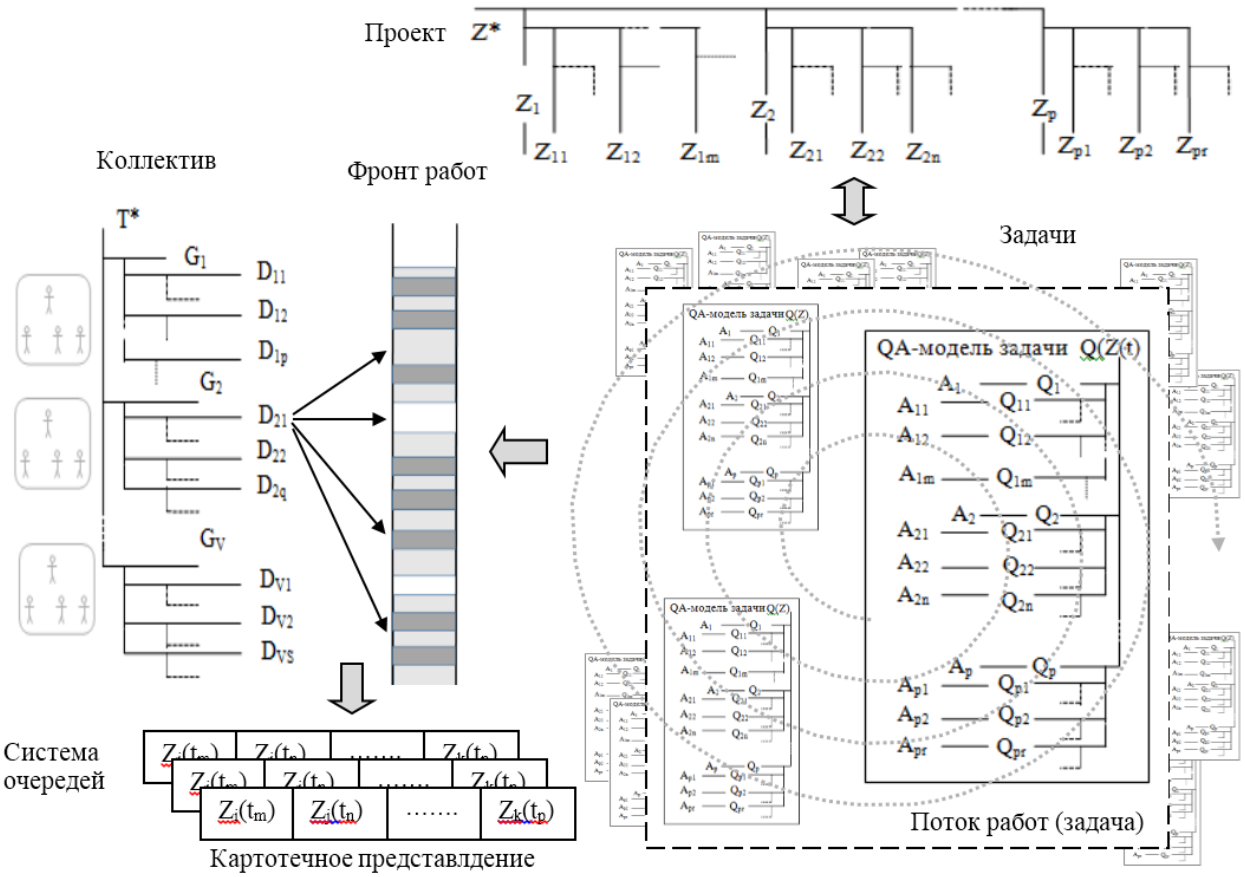


Рис. 1.12. Средства ПКУ потоками работ в WIQA

Одним из вариантов представления фронта работ и очередей задач является картотечное представление, в качестве образца для которого выступает доска Kanban[81][94].

1.4. Задача диссертационного исследования

1.4.1. Обобщенная постановка задачи

Переходя к формулировке задаче исследований диссертационной работы, начнём с её обобщённой постановки следующего вида:

Z*. 1. Разработать комплекс методов и средств управления потоками работ, способствующий предотвращению ошибок, обусловленных человеческим фактором, и обеспечивающий сокращение

непроизводительных затрат времени в оперативной коллективной работе за счет включения в совокупность средств управления проектной деятельностью дополнительных программируемых составляющих

2. Механизм дополнительных программируемых составляющих, предлагаемых комплексом средств следует ориентировать на участие в их реализации проектировщиков, выполняющих порученную им работу в соответствии с планами, представленными на языке, близком к ЕЯ, в его алгоритмических условиях

3. Для снижения затрат на разработку комплекса средств управления потоками проектных работ связать его информационно по входу и выходу с вопросно-ответной инструментальной средой WIQA.Net

1.4.2. Вопросно-ответный анализ

Формулировка задачи построена в виде, с которого должно начинаться применение QA-подхода, первые шаги которого требуют извлечь из формулировки вложенные в неё вопросы.

В наиболее общем плане постановка задачи ориентирует на управление потоками работ с помощью дополнительных программных составляющих в оперативной коллективной работе и нацеливает на ряд положительных эффектов, вызванных этим управлением.

Следовательно, в число важнейших исходных вопросов должны быть включены следующие:

Q1. Какова специфика потоков работ в проектировании АС?

Q2. Какие особенности человеческого фактора вызывают негативные эффекты?

Q3. Как возникают непроизводительные затраты времени в оперативной коллективной работе?

Q4. Каково место дополнительных программируемых составляющих в процессе управления проектной деятельностью?

Q5. Каким образом дополнительные программируемые составляющие позволяют достичь указанных положительных эффектов?

Q6. Что представляет собой язык, близкий к ЕЯ?

Q7. Какова цель представления планов работ на языке, близком к ЕЯ?

Q8. Каким образом интеграция с вопросно-ответной инструментальной средой WIQA.Net позволит снизить затраты, связанные с разработкой программных средств?

Q9. Каким образом должно осуществляться взаимодействие разрабатываемых программных систем с инструментальной средой WIQA.Net?

В процессе проектирования коллектив разработчиков осуществляет человеко-компьютерную активность, направленную на решение проектной задачи. Объём этой активности ограничивается множеством проектных задач. Каждая задача имеет ассоциацию с актором, решающим данную задачу.

Таким образом, из анализа вопроса Q1 можно выделить следующие вопросы:

Q1.1. Какие акторы выполняют проектные задачи?

Q1.2. Каким образом различается активность в зависимости от актора?

Q1.3. Каким образом осуществляется координирование выполнения множества проектных задач?

Руководит проектом и группой разработчиков непосредственный

руководитель. Он является актором. Также актором является группа разработчиков, выполняющих проектный процесс. Разработчики в группе, взаимодействуя между собой и с руководителем, также являются акторами. В некоторых случаях в роли актора может выступать также автоматизированная система, например, экспертная система. Таким образом, список акторов может расширяться.

Активность группы разработчиков является коллективной, поскольку состоит из активностей элементарных акторов, а разработчика или руководителя – персональной, выполняемой одним, неделимым актором.

Из вопроса ***Q1.2*** возникает ещё один вопрос:

Q1.2.1. Чем отличается коллективная активность от персональной?

В отличие от индивидуальной активности, коллективная исполняется сразу несколькими элементарными акторами – членами коллектива, соответственно, она предполагает параллелизм. В персональной же активности для достижения параллелизма приходится прибегать к псевдо параллелизму, связанному с необходимостью переключения между задачами.

Поскольку коллективная активность отличается от персональной реализацией параллелизма, существует различие между координированием множества проектных задач, выполняемых в процессе коллективной активности и координированием множества проектных задач, выполняемых в процессе персональной активности. Поэтому из вопроса Q1.3. возникают следующие вопросы:

Q1.3.1. Каким образом осуществляется координирование выполнения множества проектных задач в процессе коллективной активности?

Q1.3.2. Каким образом осуществляется координирование выполнения множества проектных задач в процессе персональной

активности?

С ответами на эти вопросы связывается следующее содержание:

A1.1. Член группы проектировщиков, группа проектировщиков, руководитель группы, АС

A1.2. Активность группы проектировщиков является коллективной, а активность члена группы проектировщиков – персональной

A1.2.1. Коллективная активность предполагает наличие параллелизма в процессе решения задач, в отличие от индивидуальной, реализация мультизадачности которой сопряжена с необходимостью переключения между задачами.

В процессе коллективной активности затрагиваются связи между акторами, совокупная активность которых составляет коллективную активность. Руководитель коллективной активности управляет распределением задачной нагрузки между членами коллектива. В отличие от координирования коллективной активности, координирование персональной активности заключается в поддержке механизма псевдопараллелизма.

Таким образом, с этими вопросами связывается следующее содержание ответов:

A1.3.1. Координирование осуществляется путем распределения задачной нагрузки между членами коллектива, а также за счет учета логических межзадачных связей.

A1.3.2. Координирование осуществляется путем поддержки механизма псевдопараллелизма.

A1.3. Координирование выполнения множества проектных задач осуществляется за счёт учёта акторами логических связей между задачами.

Из обобщения всех ответов на подвопросы вопроса ***Q1***, ответом на этот вопрос будет следующий текст:

> ***A1. В общем случае поток работ в проектировании АС представляет собой определенный объём человеко-компьютерной активности, включающей координируемое выполнение множества проектных задач, выполняемых различными акторами.***

Говоря об особенностях человеческого фактора, приводящих к негативным эффектам в процессе коллективного проектирования, следует упомянуть, что эти особенности различаются для процессов коллективной и персональной активности ввиду различий между механизмами их координирования. Таким образом, вопрос ***Q2*** разделяется на два следующих вопроса:

> ***Q2.1. Какие особенности человеческого фактора приводят к негативным эффектам в процессе коллективной активности?***
>
> ***Q2.2. Какие особенности человеческого фактора приводят к негативным эффектам в процессе персональной активности?***

В процессе персональной активности к негативным эффектам приводят такие особенности человеческого фактора, как непроизводительные затраты времени, связанные с особенностями псевдопараллелизма, а также – ошибки, возникающие в процессе псевдопараллельного выполнения задач. Чтобы наиболее полно ответить на этот вопрос, следует также ответить на следующий:

> ***Q2.2.1. Каким образом осуществляется псевдопараллельное выполнение задач?***

Попытки одновременного выполнения человеком нескольких задач могут привести к ошибкам. Псевдопараллелизм предполагает последовательное переключение фокуса внимания с одной задачи на другую. В один момент времени выполняется только одна задача. После переключения фокуса

внимания происходит восстановление в памяти человека оперативного контекста задачи, на которую происходит переключение. На вопрос ***Q2.1.1*** можно ответить следующим образом:

> ***A2.2.1. Псевдопараллельное выполнение задач осуществляется последовательным переключением фокуса внимания между задачами с восстановлением оперативного контекста задачи после переключения***

На восстановление оперативного контекста задачи после переключения внимания требуется определенное время. Эти временные затраты являются непроизводительными, т.к. пока оперативный контекст не восстановлен полностью, человек не может приступить к решению задачи без риска возникновения ошибок. Ошибки могут возникать вследствие неполного восстановления оперативного контекста или его искажения.

Таким образом, на вопрос ***Q2.2*** можно ответить следующим текстом:

> ***A2.2. К негативным эффектам приводят такие особенности человеческого фактора, как временные затраты, связанные с необходимостью восстановления оперативного контекста задачи при переключении между задачами, а также вероятность неполного или ошибочного восстановления оперативного контекста задачи.***

Данные негативные эффекты позволяют минимизировать дополнительные программные составляющие, которые можно включить в комплекс методов и средств управления потоками работ.

Ответом на вопрос ***Q2.1***. является следующий текст:

> ***A2.1. К негативным эффектам в процессе коллективной активности приводят такие особенности, как сложность в оценке оптимальной задачной нагрузки***

Ответ на вопрос Q2 образуется в результате объединения ответов на

вопросы Q2.1 и Q2.2:

A2. К негативным эффектам в процессе коллективной активности приводят такие особенности, как временные затраты, связанные с необходимостью переключения оперативного контекста при переключении между задачами, риск возникновения ошибок при его переключении, а также - сложность в оценке оптимальной задачной нагрузки.

В процессе выполнения коллективной работы происходит выполнение ряда задач проекта, а также – ряда задач управления проектным процессом. Решая задачи управления проектным процессом, коллектив не выполняет задачи проекта. Таким образом, возникают непроизводительные затраты времени, связанные с выполнением этих управленческих задач. Другим источником непроизводительных затрат времени является нерациональное оперативное планирование проектных работ, в ходе выполнения которых возникают ситуации простоя участников проектного процесса. Кроме этого, непроизводительные затраты времени возникают в связи с особенностями человеческого фактора, рассмотренными в рассуждениях, относящихся к вопросу ***Q2***.

Таким образом:

A3. Непроизводительные затраты времени в процессе коллективного проектирования возникают в результате необходимости выполнения задач управления проектным процессом, в результате нерационального оперативного планирования, а также в связи с влиянием человеческого фактора.

Возникают следующие вопросы, связанные с сокращением непроизводительных затрат времени:

Q3.1. Каким образом можно минимизировать временные затраты, связанные с необходимостью выполнения задач управления проектным

процессом?

Q3.2. Каким образом можно минимизировать временные затраты, связанные с нерациональным планированием проектного процесса?

Q3.3. Каким образом можно минимизировать временные затраты, связанные с человеческим фактором?

Задачи управления проектным процессом являются автоматизируемыми путем включения программных средств, позволяющих более рационально осуществлять управленческие процессы. Данные средства должны включать в себя инструментарий планирования проектного процесса, позволяющий учитывать межзадачные зависимости и другие факторы, которые могут привести к простоям в работе проектировщиков. Также с целью рационализации оперативного планирования этапа проектного процесса можно использовать метрические характеристики процесса выполнения предыдущих этапов проектного процесса в соответствии с Agile-методологиями проектирования.

Таким образом, на с вопросами Q3.1 и Q3.2 связаны следующие ответы:

A3.1. Минимизация временных затрат, связанных с необходимостью выполнения задач управления проектным процессом осуществляется путем включения программных средств автоматизации решения задач проектного процесса.

A3.2. Минимизация временных затрат, связанных с нерациональным планированием проектного процесса осуществляется путем учета факторов, вызывающих простои в работе проектировщиков, а также путем использования в оперативном планировании метрических характеристик процесса выполнения предыдущих этапов проектирования.

Возникающие в процессе выполнения проектных работ временные затраты, связанные с особенностями человеческого фактора можно

минимизировать путем включения программных средств управления прерываниями, которые облегчают процесс переключения проектировщика между задачами и помогают ему восстановить оперативный контекст прерванной задачи после прерывания. Таким образом, ответ на вопрос ***A3.3*** связан со средствами управления прерываниями:

> ***A3.3. Минимизация временных затрат, связанных с человеческим фактором достигается путем включения программных средств управления прерываниями.***

В процессе проектирования потоки работ являются логически связанными цепочками задач, предполагающих их параллельное выполнение. Следовательно, поток работ является алгоритмически программируемым и это программирование потока работ является одной из программируемых составляющих в проектном процессе. Другой программируемой составляющей в проектном процессе является псевдопараллелизм в выполнении задач проекта сотрудниками. Так как задачи могут иметь различные приоритеты и требовать различное количество времени на из выполнение, организация их очереди и переключение между задачами должно выполняться в соответствии с определенным алгоритмом, который зависит от набора решаемых задач[13].

Таким образом, с ответом на вопрос ***Q4*** связано следующее содержание:

> ***A4. Программируемыми составляющими в проектном процессе является программирование параллельного выполнения задач потока работ, и программирование очередей и псевдопараллельного выполнения задач проектировщиком.***

Если же говорить о положительных эффектах, достигаемых путем включения дополнительных программных составляющих, то вопрос ***Q5*** подразделяется на два вопроса, связанных с различными видами программируемых задач в проектном процессе:

Q5.1. Какие положительные эффекты достигаются в результате программирования параллельного выполнения потока проектных работ?

Q5.2. Какие положительные эффекты достигаются в результате программирования псевдопараллельного выполнения задач проектировщиком?

Программирование параллельного выполнения потока проектных работ позволяет построить и запрограммировать алгоритм выбора задач участниками проектного процесса, учитывающий связность задач таким образом, чтобы избежать или минимизировать простои в работе проектировщиков над задачами. Также запрограммированный алгоритм можно использовать для отбора задач для оперативного планирования, и тем самым, ускорить этот процесс. Программирование псевдопараллельного выполнения задач проектировщиком позволяет построить алгоритм переключения между задачами с учетом, во-первых, управляемых прерываний, а во-вторых – организовать псевдопараллельное выполнение задач проектировщиком с учетом их приоритета и трудоемкости, и тем самым, снизить негативное влияние человеческого фактора.

Таким образом, с этими вопросами связано следующее содержание:

A5.1. В результате программирования параллельного выполнения потока проектных работ достигается задача минимизации простоев в работе проектировщиков, а также – задача рационализации процесса оперативного планирования.

A5.2. В результате программирования псевдопараллельного выполнения задач проектировщиком происходит снижение негативного влияния человеческого фактора.

Для описания межзадачных взаимосвязей требуется особый язык. Наиболее подходящим языком для их описания является естественный язык,

понятный всем разработчикам. Но у естественного языка есть существенный недостаток, такой, как сложность в его машинной трансляции. Для преодоления этого недостатка можно использовать псевдо-кодовый язык программирования. Однако этот язык должен быть расширен таким образом, чтобы получить возможность описывать алгоритмические структуры, специфические для потоков работ. Возникают следующие вопросы:

Q6.1. Какие структуры, не входящие в базовый набор структур алгоритмических языков программирования, требуются для описания потоков работ?

Q6.2. Каким образом можно расширить псевдо-кодовый язык?

Описывая на языке псевдо-кода планы потоков проектных работ, разработчики получают возможность трансляции этих потоков работ с целью автоматизированной конвертации их в форматы данных, которые используются в программных средствах управления потоками работ.

Таким образом, на вопросы Q4-Q6 можно ответить следующим образом:

A6.1. Для описания потоков работ требуются методы имитационного моделирования

A6.2. Псевдо-кодовый язык можно расширить, дополнив его рядом операторов из языка имитационного моделирования

A6. Язык, близкий к ЕЯ, представляет собой язык псевдо-кода, дополненный операторами языка имитационного моделирования.

A7. Представление планов потоков работ на языке, близком к ЕЯ, позволяет осуществить их трансляцию с целью автоматизированного заполнения структур данных в соответствии с заданным планом в средствах управления потоками работ.

Для расширения языка псевдокода требуется открытая реализация этого

языка. Такая реализация существует в среде вопросно-ответного процессора WIQA.Net, разрабатываемой на кафедре «Вычислительная техника». Кроме этого, данная среда представляет возможность объединения различных средств с помощью общей вопросно-ответной базы данных, доступной из всех плагинов этой среды. Кроме этого, данная среда содержит ряд инструментов, таких, как средство управления организационными структурами, средство протоколирования действий пользователя, которые могут потребоваться при реализации средств управления потоками работ. Расширение языка L^{WIQA} и использование среды вопросно-ответного процессора WIQA.Net позволяет сократить временные затраты на реализацию комплекса средств управления потоками работ за счат использования уже реализованных в ней инструментов и базы данных.

A8. Интеграция со средой WIQA.Net позволяет снизить затраты, связанные с реализацией средств управления потоками работ за счет использования реализации интерпретатора языка псевдокода, предполагающего возможность расширения, а также за счет уже реализованных в ней средств и вопросно-ответной базы данных, представляющей собой единое для всех реализуемых средств пространство данных.

A9. Интеграция со средой WIQA.Net обеспечивается за счет расширения этой среды разрабатываемыми программными средствами.

1.4.3. Мотивационно-целевые установки

К числу принципиальных результатов любой диссертационной работы относятся те положительные результаты, которые можно получить от

внедрения в практику. В данном случае – в практику коллективного проектирования сложных АС. Рациональным является предварительное оформление ожиданий в виде мотивационно-целевых установок, а затем, шаг за шагом, доводить через спецификации до мотивов и целей. Для определения мотивационно-целевых установок вернемся к обобщенной постановке задачи диссертационного исследования, к анализу её первого предложения, сформулировав на его основе основной мотив диссертационной работы:

М0. Включение в совокупность средств управления проектной деятельностью дополнительных программных составляющих в совокупность средств управления проектной деятельностью, должно привести к повышению степени успешности оперативной коллективной работы в разработке сложных АС.

Оперативная коллективная работа включает в себя как коллективную активность, то есть активность группы разработчиков, так и персональную активность, которая характеризует активность каждого разработчика по отдельности. Эти активности существенно различаются между собой и требуют различных подходов к их оптимизации. Таким образом, средства управления проектной деятельностью можно разделить на две группы:

- Средства управления коллективной проектной деятельностью
- Средства управления персональной проектной деятельностью

Следовательно, мотив М0 включает в себя содержание двух подмотивов:

М0.1. Повысить степень успешности оперативной коллективной работы в разработке сложных АС за счет включения в совокупность средств управления коллективной проектной деятельностью дополнительных программных составляющих.

М0.2. Повысить степень успешности оперативной коллективной работы в разработке сложных АС за счет включения в совокупность

средств управления персональной проектной деятельностью дополнительных программных составляющих.

С мотивом М0 связано влияние человеческого фактора на крайне низкую степень успешности проектирования SIS. Снизить его влияние можно, сделав программное управление проектным процессом более естественным. Эту естественность можно достичь путем построения алгоритмического псевдо-кодового языка, который позволил бы использовать его для взаимодействия с доступным для него опытом. Наиболее удобным для использования человеком является естественный язык, а естественно-языковой доступ к опыту имеет диалоговую природу. Следовательно, разрабатываемый псевдо-кодовый язык должен учитывать особенности, которые существенны в отношениях между естественным языком и человеческим опытом. Таким образом, одна из целей формулируется следующим образом:

Ц0. Построить алгоритмический псевдокодовый язык ориентированный на его употребление проектировщиками для взаимодействия с доступным опытом и его моделями.

Достижение этой цели приводит к следующим эффектам:

- Появляется мощный инструмент для работы с проектным опытом;
- Использование опыта проектной работы приводит к расширению возможностей в повторном использовании успешных практик проектной деятельности.

Из мотива ***М0.1*** вытекает необходимость разработки программного инструментария, позволяющего управлять проектной деятельностью. Поскольку проектная деятельность разделяется на коллективную и персональную, выделим в отдельную цель включение инструментария управления коллективной проектной активностью в средства управления проектной активностью. Соответственно, формируется цель ***Ц1***, связанная с управлением коллективной проектной активностью:

Ц1. Дополнить инструментарий управления проектной деятельностью средствами управления коллективной проектной активностью.

Достижение этой цели осуществляется за счет включения инструментария управления коллективной активностью и приводит к следующим эффектам:

- Снижение влияния человеческого фактора на эффективность коллективной проектной активностью;
- Обеспечивается обратная связь в управлении коллективной проектной деятельностью, позволяющая с каждым следующим этапом проектной деятельности осуществлять ее совершенствование.

Одним из направлений управления коллективной проектной активностью является планирование проектных работ. Работы эти связаны между собой и образуют потоки работ. Псевдокодовый язык программирования, позволяющий работать с опытом проектных работ, также позволяет моделировать и потоки работ, подлежащих выполнению. Таким образом, включение данного языка программирования позволяет осуществить планирование потоков проектных работ, которое является подцелью цели ***Ц1***:

Таким образом, формируется цель ***Ц1.0***, являющаяся подцелью цели ***Ц1***:

Ц1.0. Включить в средства управления коллективной проектной активностью инструментарий планирования потоков работ.

Достижение этой цели также приводит к эффекту построения обратной связи в управлении потоками работ, реализующейся в виде более рационального планирования и выполняющийся на базе вычисления метрических характеристик выполненных работ.

В процессе планирования потока работ необходимо также обеспечить контроль его целостности, заключающийся в обеспечении проверки условий зависимости задач друг от друга. Достигается этот контроль также путем

применения псевдокодового языка, позволяющего, с одной стороны, описать эти зависимости, а с другой – программы, написанные на этом языке, являются выполнимыми, что позволяет добиться некоторой автоматизации этого процесса. Соответственно, можно говорить о том, что в комплекс средств управления проектной деятельностью включаются инструменты, позволяющие контролировать связность потока работ. Включение этих инструментов является ещё одной подцелью цели ***Ц1***:

Ц1.1. Включить в средства управления коллективной проектной активностью инструменты, позволяющие осуществить контроль за связностью потока работ.

Эффектом от достижения данной цели является контроль корректности планирования потока работ за счет включения инструментария контроля поручений.

В процессе выполнения проектной деятельности коллектив, выполняя работы, должен осуществлять выполнение задач в запланированном порядке. Кроме того, в процессе выполнения проектных работ в план может вноситься корректировка. Кроме этого, процесс выполнения потока работ характеризуется рядом метрических характеристик. Соответственно, следующая подцель цели ***Ц1*** характеризует контроль хода проектной работы:

Ц1.2. Включить в средства управления коллективной проектной активностью инструменты контроля за ходом выполнения проектной деятельности.

Эффекты, получаемые в результате достижения этой цели за счет включения следующие:

- Метрические характеристики также способствуют обеспечению обратной связи в процессе выполнения проектного процесса.
- Включаемые инструментальные средства позволяют обеспечить выполнение проектных работ в соответствии с планами;

Кроме этого, после выполнения этапа проектных работ появляется возможность рассчитать метрические характеристики выполненного потока работ, позволяющие в дальнейшем скорректировать планирование следующих этапов. Таким образом, следующая подцель цели ***Ц1*** формулируется как включение инструментов анализа выполненной проектной работы на завершенном её этапе:

Ц1.3. Включить в средства управления коллективной проектной активностью инструменты анализа выполнения этапа проектной деятельности.

Эффекты от достижения этой цели следующие:

- Также её достижение обеспечивает формирование обратной связи в процессе коллективного проектирования;
- Результаты анализа выполнения этапа проектной деятельности могут быть представлены заинтересованным в них стейкхолдерам в целях отчетности.

Как было сказано выше, проектная активность подразделяется на коллективную и персональную. Следующая цель связана с минимизацией негативного воздействия человеческого фактора на персональную активность. Цель ***Ц2*** формулируется следующим образом:

Ц2. Дополнить инструментарий управления проектной деятельностью средствами управления персональной активностью.

Эффектом от достижения данной цели является минимизация негативного воздействия человеческого фактора на персональную проектную деятельность.

Поскольку параллельное выполнение задач одним человеком затруднено, в инструментарий управления персональной активностью включаются средства организации мультизадачности:

Ц2.0. Включить в инструментарий управления персональной

активностью средства организации распараллеливания выполнения задач;

Эффекты от включения данных средств следующие:

- Рациональное формирование и управление очередями задач проектировщика;
- Организация псевдопараллельного выполнения проектных задач одним проектировщиком.

Поскольку в инструментарии управления персональной активностью появляются средства формирования и управления очередями задач, то возникает необходимость в визуальном контроле очередей, соответственно, появляется следующая цель:

Ц2.1. Включить в инструментарий управления персональной активностью средства визуального контроля очереди персональных проектных задач.

Эффектом достижения данной цели является облегчение выбора следующей работы из потока проектных работ проектировщиком.

На основании сформулированных целей и мотивов можно построить мотивационно-целевую диаграмму данной работы. Диаграмма представлена на следующем рисунке:

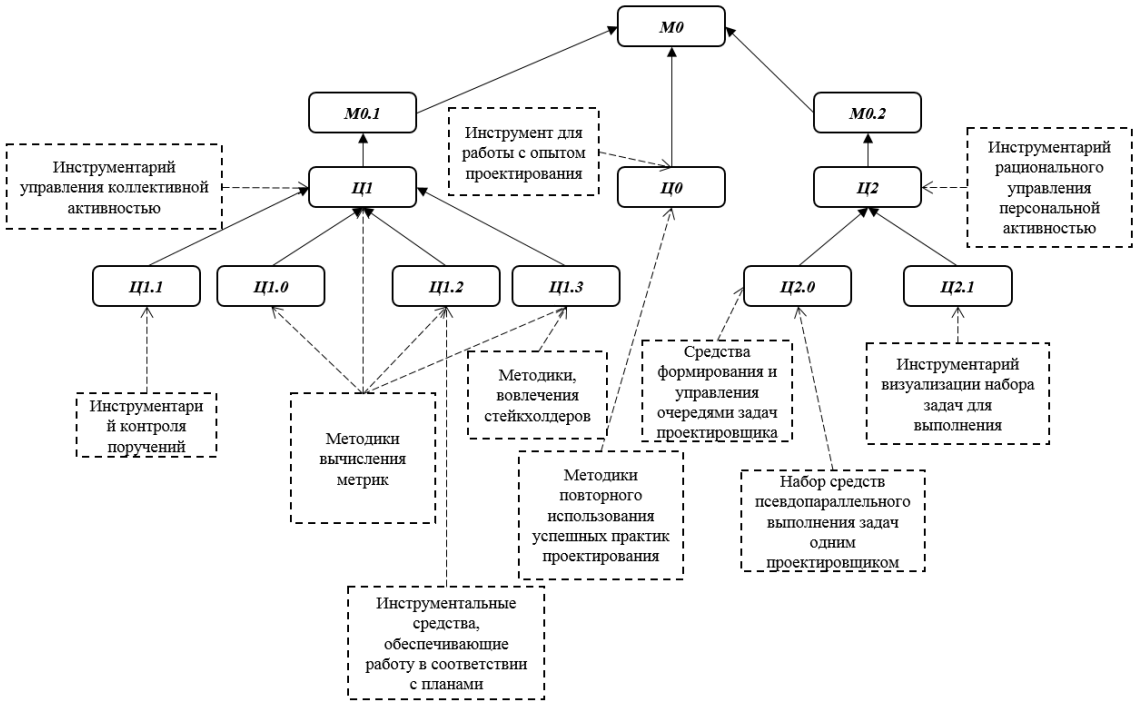


Рис. 1.13. Мотивационно-целевая диаграмма ПКУ потоками работ.

Выводы по первой главе

1. Разработка систем, интенсивно использующих программное обеспечение, считается одним из наиболее проблемных видов деятельности, степень успешности которой недопустимо низка. За чрезвычайно низким процентом успеха в разработках систем SIS стоят определенные причины, обусловленные, в основном тем, что освоение (человечеством) такого вида деятельности еще не достигло уровня гарантийной предсказуемости ее результатов;
2. Уровень зрелости производственного процесса – это степень, до которой тот или иной процесс определен, управляем, измеряем, контролируем и эффективен;
3. Включение комплекса средств ПКУ позволяет ввести в проектную деятельность не только измерения её эффективности, но также и контроль, тем самым приблизиться к максимальному – эффективному уровню зрелости процесса проектирования;
4. Наиболее полезным переходом от бизнес-процессов реальности к их

представлению в модельной действительности является моделирование бизнес-процессов и их совокупностей в форме потоков работ;

5. Существует ряд технологий проектного управления, представляющих наборы знаний, инструкций и инструментальных средств, обеспечивающих выполнение процесса проектирования с учетом богатого опыта проектирования, накопленного в ходе многолетней практики проектирования в различных организациях по всему миру;
6. В отдельный класс можно выделить гибкие методологии разработки, ориентированные на Agile-подход, предполагающий итеративность разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Ряд этих методологий не исключает их совместного применения, а также применения в сочетании с не-Agile дисциплинированными технологиями;
7. В состав комплекса средств ПКУ должен быть включен инструментарий параллелизма и псевдопараллелизма в выполнении задач проектировщиками;
8. Псевдопараллельное выполнение задач проектировщиком ведет к возникновению прерываний, приводящих к непроизводительным затратам времени;
9. Проектирование инструментальных средств, учитывающих особенности человеческих прерываний, позволяет минимизировать данные непроизводительные временные затраты;
10. В качестве платформы для реализации средств ПКУ наиболее рациональным является использование среды WIQA, в задачи которой входит гибкое проектное управление.

Глава вторая. Формализация и специализация процессов ПКУ.

2.1. Архитектурная модель системы ПКУ

Определимся с совокупностью средств программно-картотечного управления (ПКУ) учитывая, что она является дополнением к традиционному управлению в разработках ***AC***.

Будем учитывать и то, что для реализации ПКУ принято решение об использовании среды **WIQA**. Это решение подсказывает необходимость отображения ПКУ, а вернее, её структуры, процессов построения и использования ПКУ на семантическую память этой среды.

Для осуществления такого отображения начнем со структуры ПКУ, представив её в виде архитектурной модели. Обобщенная версия такой модели с позиций управления оперативной работы проектировщика приведена на рисунке 3.1.

Перейдем к обобщенным спецификациям компонентов системы ПКУ.

В верхней части рисунка 3.1 представлено схематичное изображение потоков работ. Поток работ состоит из наборов разнотипных проектных работ (на рисунке 3.1 обозначенных, как $\boldsymbol{Z^X}$), где каждой работе соответствует набор элементарных задач, совокупность которых составляет поток работ. На рисунке 3.1 выделен один из потоков работ, соответствующей выделенной задаче. Элементарные задачи $\boldsymbol{Z_{ij}}$ этого потока работ связаны между собой. Какие-то из них допускают параллельное выполнение, какие-то другие задачи требуют для работы над ними результатов выполнения предыдущей задачи.

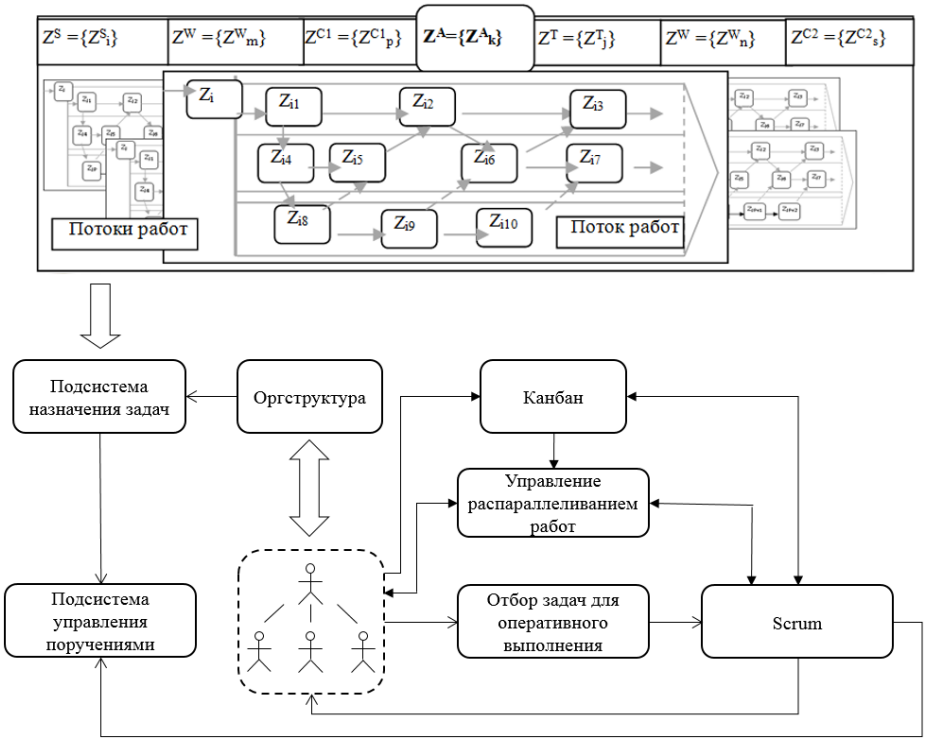


Рис. 2.1. Архитектурная модель ПКУ

Подсистема оргструктуры ***WIQA*** реализует инвариантную интерактивную модель группы проектировщиков, которая в корпоративной среде обеспечивает оперативную адаптацию, рациональное использование и развитие индивидуального и коллективного опыта. Эта подсистема позволяет установить связи между сотрудниками и задачами проекта, а также – между сотрудниками и ролями, которые они выполняют в проектном процессе. Наборы ролей могут быть адаптированы под конкретный коллектив и конкретный проект.

Роль – это потенциальная возможность проектировщика решить определенный набор задач. Любая роль актуализируется только после того, как проектировщику, исполняющему роль, будет назначена хотя бы одна из задач роли.

После назначения задач в организационной структуре требуется указать для каждой из них примерные сроки выполнения этой задачи и сформировать поручение на выполнение этой задачи. Для этих целей служит **подсистема**

контроля поручений.

Существует множество инструментов, обеспечивающих возможность планирования рабочего процесса, включая такие программные системы, как ***MS Project***. Такой тип систем отлично зарекомендовал себя в планировании и контроле выполнения проектной работы. Эти инструменты позволяют сформировать график работы над проектами и их задачами, обеспечить временные резервы на случай непредвиденных временных затрат, отслеживать прогресс и анализировать объем работ. MS Project позволяет визуализировать цепочку работ в виде диаграмм Ганта. Обеспечение интеграции программных пакетов планирования с другими компонентами системы ПКУ на уровне доступа к данным для автоматизации процесса управления проектной деятельностью является достаточно сложной задачей. Поэтому в диссертационном исследовании было принято решение упростить все вопросы, связанные с привязкой выполнения работ по назначенным задачам ко времени, а более конкретно, принято решение осуществлять привязку по образцу систем контроля поручений. Система контроля поручений позволяет обеспечить привязку поручений к задачам, хранящимся в памяти ***WIQA***, а также обеспечить программный доступ к данным поручений, используя отображение поручений на ***QA***-память.

Для осуществления привязки в системе контроля поручений необходимо:

- Провести ассоциацию между сотрудником или группой сотрудников и задачей совместно со связанной с ней ролью;
- Назначить ориентировочную дату выполнения задачи;
- Задать значения набору дополнительных атрибутов поручения, таких, как ответственные лица, контролеры, приоритет задачи;
- Проинформировать сотрудника о новом поручении.

В средствах, поддерживающих такие виды управления, потенциально доступный набор задач для их решения их называют бэклогом.

В процессе выполнения потока работ перед каждым сотрудником в течение заданного периода времени возникает необходимость параллельного выполнения работ. Попытки одновременного выполнения нескольких задач могут приводить к ошибкам, и поэтому человеку приходится осуществлять псевдопараллельное выполнение задач посредством прерываний, по аналогии с реализацией мультизадачности в ЭВМ. В реализации прерываний центрального процессора используются такие механизмы, как приоритетизация задач, очередь прерываний, запрет прерывания. В той или иной степени, эти механизмы можно реализовать и на интеллектуальном процессоре человека. За это отвечает изображенная на схеме 2.2 **подсистема управления распараллеливанием.**

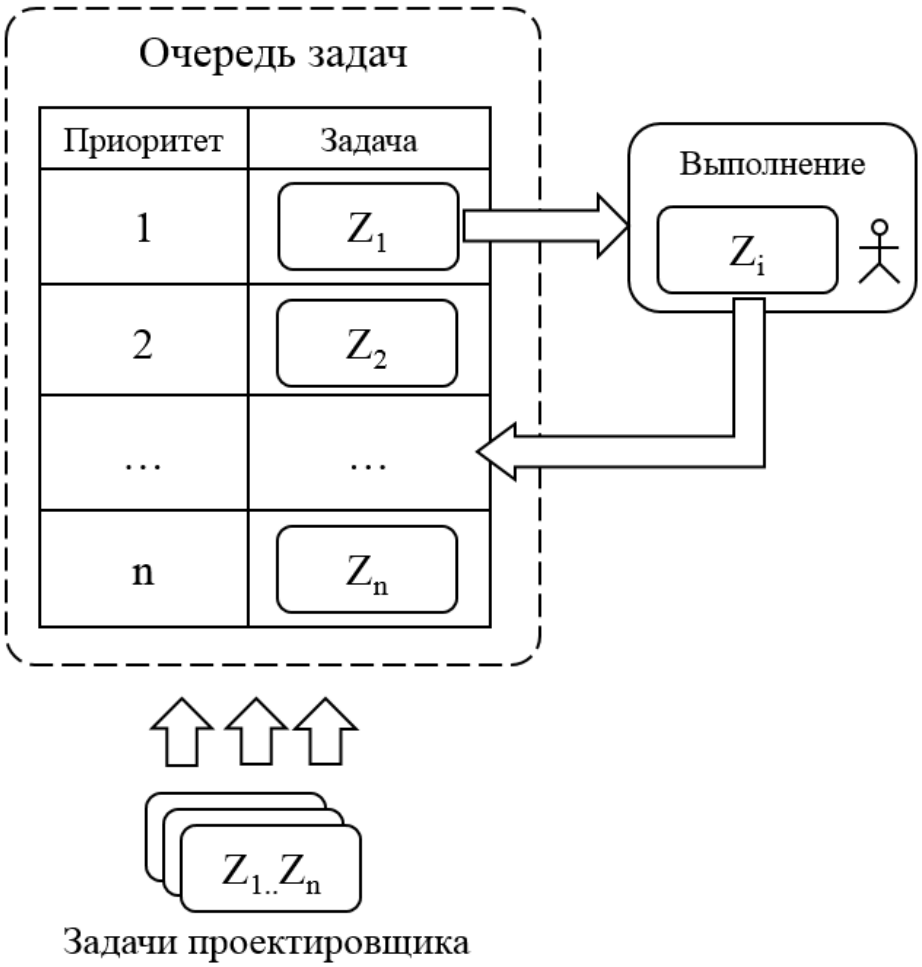


Рис. 2.2. Псевдопараллельное выполнение работ человеком

Эта подсистема обеспечивает следующую функциональность:

- Очередь задач;
- Прерывание задачи и помещение ее в очередь с рассчитанным или заданным приоритетом;
- Сохранение оперативного контекста прерванной задачи;
- Переход к задаче, хранящейся в очереди;
- Восстановление оперативного контекста задачи в том случае, если она была прервана.

2.2. Отображение средств ПКУ на память WIQA

2.2.1. Отображение среды ПКУ на память WIQA

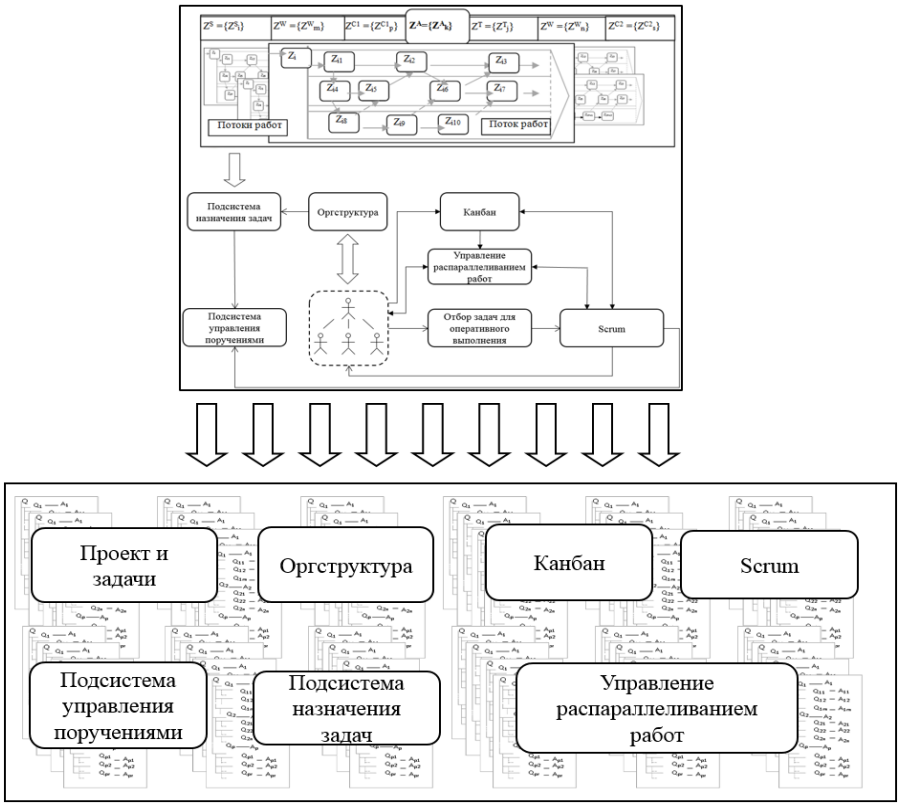


Рис. 2.3. Отображение среды ПКУ на память WIQA

В среде ***WIQA*** данные, с которыми работает среда, подразделяются на два основных типа – данные, хранимые в реляционной БД на сервере ***WIQA***, и данные, хранимые в виде файлов на локальной файловой системе.

Данные WIQA = БД, файлы;

Данные, хранимые в файлах на локальной файловой системе могут быть произвольными и их формат зависит только от решаемой задачи. В свою очередь, данные, которые хранятся в ***БД***, подразделяются на два основных типа – ***QA***-память ***WIQA*** и табличные данные, не укладывающиеся в формат вопросно-ответных структур. Таким образом,

БД = QA-память, табличные данные;

В QA-памяти, применительно к системе ПКУ, хранятся данные проекта, проектных работ, поручения, Kanban, Scrum, псевдокодовые программы. В виде табличных данных представляется организационная структура.

В следующих пунктах рассмотрим представление изображенных на рисунке 2.3 компонентов ПКУ в памяти ***WIQA***.

2.2.2. Отображение проекта и задач на память WIQA

Отображение проектов и задач на семантическую память ***WIQA*** было освоено и успешно применено на предприятии НПО "Марс".

Перед тем, как построить отображение проекта на память ***WIQA*** сформулируем в виде РБНФ саму память. Вопросно-ответная память ***WIQA*** представляет собой иерархически организованное хранилище ***QA***-объектов:

QA-память = {QA-объект};

Причем сам QA-объект представляет собой пару, состоящую из вопроса и ответа. Причем ответ может быть пустым или отсутствовать:

QA-объект = Вопрос, "←", Ответ;

Здесь и далее нетерминал ***"←"*** означает вхождение (соответствие), а нетерминал ***"↓"*** – иерархическое подчинение. Мы можем ввести эти нетерминалы, поскольку сама память ***WIQA*** предполагает такие отношения.

Как вопрос может содержать в себе подвопросы, так и ответ может содержать иерархически подчиненные ему ответы:

Вопрос = Q | (Q, "↓", {Q});

```
Ответ = A | (A, "↓", {A});
```

Как вопрос, так и ответ, являются узлами вопросно-ответного дерева. Каждый узел дерева имеет определенные базовые атрибуты *g*, такие, как, дата создания, текстовое описание, индекс, статус, и некоторые другие. Кроме основных, каждый узел этого дерева может содержать в себе произвольный набор дополнительных атрибутов *ag*. Кроме основных и дополнительных атрибутов каждый узел дерева может содержать ряд прикрепленных к нему файлов:

```
Q    = ({g}, {[ag]}, {[файл]});
A    = ({g}, {[ag]}, {[файл]});
g    = (дата, описание, индекс, ..., статус);
```

В памяти WIQA проект ***П*** содержит основную задачу проекта *ZP*. Основная задача проекта, в свою очередь, является задачей, содержащей поток работ проекта.

```
П  = ZP;
ZP = (Z | "↓", {Поток работ});
```

При этом поток работ состоит из иерархически связанного набора задач ***ZW***, каждая задача ***Z*** которого представлена в виде наборов ***QA***-объектов, формулирующих задачу:

Проект WIQA предполагает следующие совокупности задач:

- совокупность задач $Z^S = \{Z_{mi}\}$ предметной области ***AC***, которые в проектируемой ***AC*** будут решать её пользователи;
- совокупность нормативных задач $Z^T = \{Z_{ni}\}$ технологии, в рамках которой коллектив проектировщиков $K(\{D_v\})$ создаёт АС;
- совокупность задач адаптации $Z^A=\{Z^A_q\}$, решаемых проектировщиками для настройки задач типа Z^T, инвариантных к проблемной области ***AC***, в их использовании при решении задач типа Z^S;

- совокупность задач управления $Z^W=\{W^m(\{Z_{mi}\})\}\cup\{W^n(\{Z_{ni}\})\}$ **и** $Z^C=\{Z^W_r\})$, решение которых обслуживает работу с задачами в потоках работ, а также согласованное управление в группах потоков работ;

Таким образом,

$$\begin{aligned}
&\textit{Поток работ} = ZW;\\
&ZW \quad = \{\{Z^S\}, \{Z^T\}, \{Z^A\}, \{Z^W\}\};\\
&Z^S = Z \mid (Z, "\downarrow", \{Z\});\\
&Z^T = Z \mid (Z, "\downarrow", \{Z\});\\
&Z^A = Z \mid (Z, "\downarrow", \{Z\});\\
&Z^W = Z \mid (Z, "\downarrow", \{Z\});\\
&Z = \{\text{QA-объект}\};
\end{aligned}$$

2.2.3. Отображение организационной структуры на память WIQA

Процесс проектирования сложных AC носит принципиально коллективный характер. Общую и очень сложную работу приходится разбивать на части и осуществлять согласованно в условиях часто изменяющихся требований и ограничений.

Как было уже упомянуто в п. 3.1, структура организации предприятия может быть представлена несколькими различными вариантами. Каждый из этих вариантов организационной структуры представляет организацию в виде двух основных составляющих частей:

- Коллектив сотрудников, постоянный или временный;
- Структура подразделений организации.

В зависимости от модели оргструктуры подразделения организации могут быть как равноправными между собой, так и иметь иерархическую или другую, более сложную структуру, имеющую как вертикальные связи между подразделениями, так и горизонтальные. Структура подразделений при этом также может быть как статичной, не изменяющейся от проекта к проекту, так и динамической, адаптируемой под каждый новый проект. К примеру, в организации НПО "Марс" структура подразделений не изменяется от одного

проекта к другому и имеет иерархическую организацию, содержащую в себе, к примеру, научно-исследовательские отделы, в подчинении каждого из которых находятся по несколько научно-исследовательских лабораторий.

При этом каждый сотрудник организации входит в то или иное подразделение, в котором он может играть роль руководителя или роль члена коллектива подразделения – рабочей группы. Каждый сотрудник может входить более чем в одно подразделение или быть руководителем сразу нескольких из них. Сотрудники в каждом подразделении находятся на определенной должности. Соответственно, в организации у сотрудника может быть сразу несколько должностей.

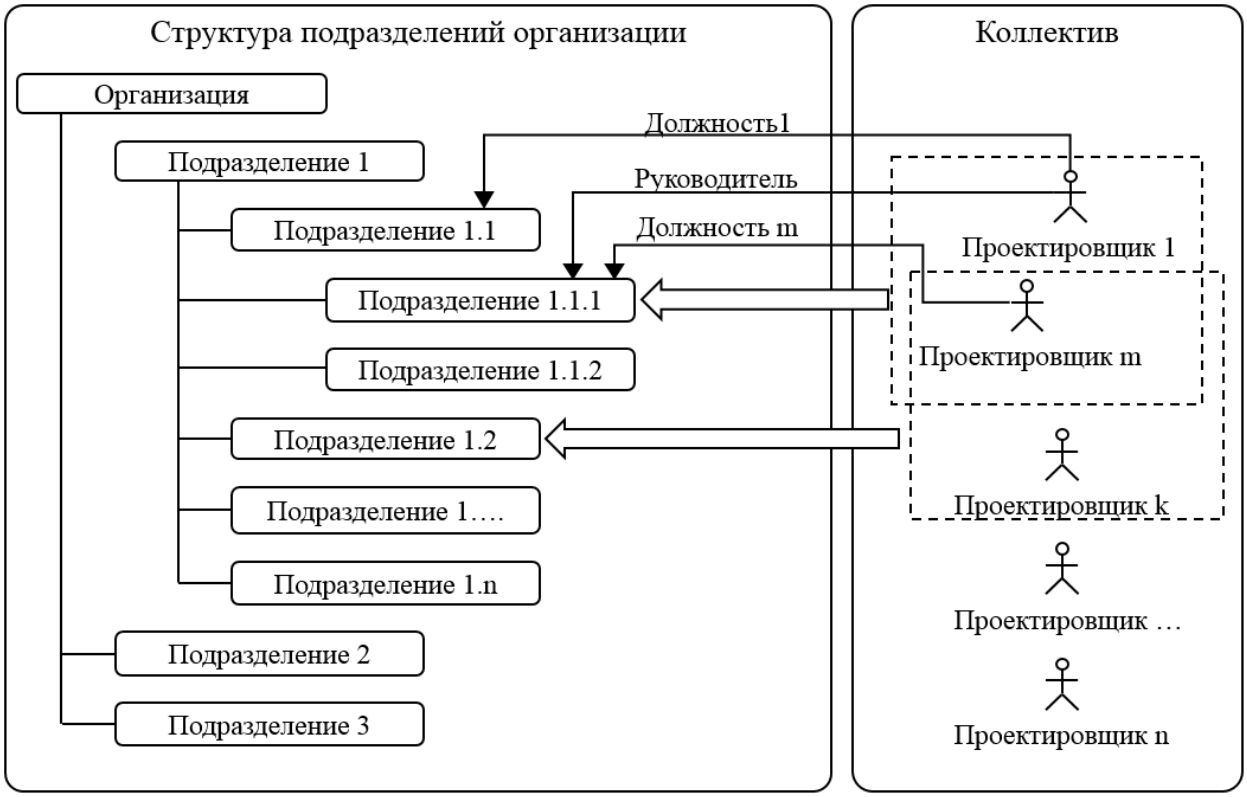


Рис. 2.4. Иерархическая структура подразделений организации

На рисунке 2.4 схематически изображена организационная структура предприятия, в котором подразделения связаны между собой иерархически. Показанный на схеме сотрудник "Проектировщик 1" работает в подразделении 1.1 на должности "Должность 1". Кроме этого, он является руководителем входящего в подразделение 1.1 подразделения 1.1.1. В

коллективе сотрудников выделены рамкой сотрудники, относящиеся к подразделению 1.1.1. Проектировщик m находится в нем на должности m.

Подразделение 1.1.1 организации вместе с относящимися к нему сотрудниками "Проектировщик1 " и "Проектировщик m" образуют рабочую группу 1.1.1, отмеченную на рисунке 2.5. Кроме неё, на рисунке 2.5 отображено, что сотрудник "Проектировщик k" также является участником рабочей группы, образованной подразделением 1.2 и его сотрудниками.

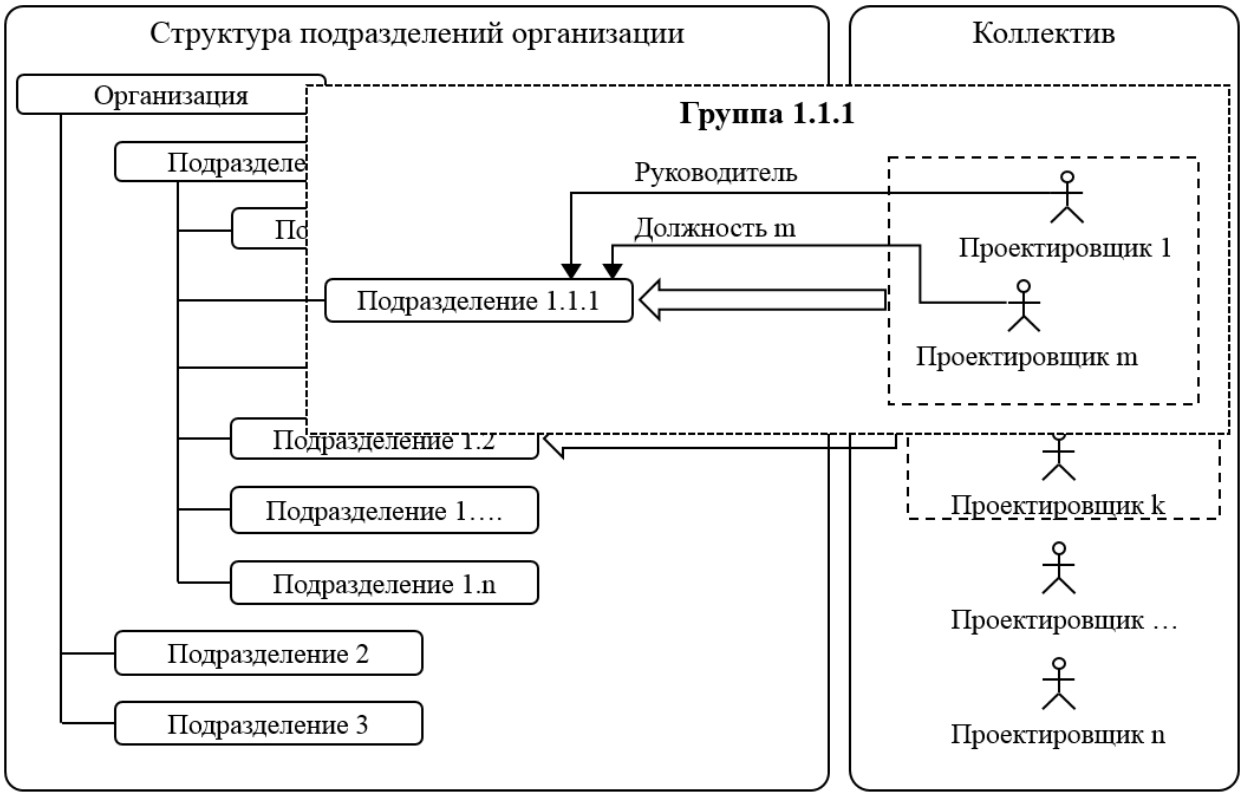


Рис. 2.5. Рабочая группа подразделения организации

Перейдем к отображению коллектива организации ***K**** на память ***WIQA***. Организационная структура ***WIQA*** строится также по иерархическому принципу в виде дерева. Корнем дерева оргструктуры является организация, узлами его – группы сотрудников ***GS***, которые также могут содержать в себе вложенные группы.

К примеру, в организационной структуре НПО "Марс" максимальная глубина дерева оргструктуры без учета отдельных сотрудников составляет 3,

а максимальная степень узла дерева равна 9.

С группами ассоциируются входящие в них сотрудники ***D***, составляющие набор сотрудников ***D**** организации.

```
K* = Организация, D*;
D* = {Сотрудник}
Сотрудник = D
O = {GS | (GS, "↓", GS)};
```

Каждая группа сотрудников состоит из подразделения ***G***, и ассоциированных с ней сотрудников ***D***.

```
GS = G, "←", {D};
```

Каждая группа определена как набор её атрибутов, включающих код группы G_c, состояние её активности G_s, наименование G_n, руководителя G_d.

```
G = Gc, Gs, Gn, Gd;
```

Набор сотрудников ***D**** представляет собой их линейный список, в котором каждому сотруднику ***D*** соответствует набор характеризующих его основных атрибутов D_a:

- Фамилия $\mathbf{D_s}$;
- Имя, $\mathbf{D_n}$;
- Отчество $\mathbf{D_p}$;
- Дата рождения $\mathbf{D_b}$;
- Табельный номер $\mathbf{D_{num}}$;
- Руководитель $\mathbf{D_c}$.

Кроме основных, каждый сотрудник имеет набор атрибутов D_{sg}, обеспечивающих его доступ к системе:

- Логин
- Пароль

Для каждого сотрудника формируется набор его контактных данных *контакты*, содержащих *телефоны*, *факсы*, *e-mail*, *www*, *icq*.

Также для каждого сотрудника формируется перечень его должностей

должности, список групп, в которых он состоит, а также – набор задач D_{nz}, выполняя каждую из которых, он играет определенную роль D_r в проектном процессе.

$$D = D_a,\ D_{sg},\ \text{контакты},\ \text{должности}$$
$$D_a = D_s,\ D_n,\ D_p,\ D_b,\ D_{num},\ D_c;$$
$$D_{sg} = \text{Логин},\ \text{Пароль};$$
$$D_{cd} = \text{телефоны},\ \text{факсы},\ \text{e-mail},\ \text{www},\ \text{icq};$$

Перечень должностей сотрудника выбирается из общего перечня должностей $\mathbf{P_d}$*.

$$\text{Должность} = P_d;$$
$$P_d^* = \{P_d\};$$
$$D_{pd} = \{P_d\};$$

В оргструктуре организации НПО "Марс" представлено более 300 должностей P_d. Каждая должность представлена её кодом и наименованием.

$$P_d = \text{Код},\ \text{Наименование};$$

Кроме должностей, атрибутом каждого проектировщика является набор его компетенций C_D, характеризующих умение проектировщика выполнять тот или иной класс проектных работ. Каждая проектная работа, при этом, характеризуется набором компетенций, которыми должен обладать проектировщик для выполнения этой работы.

$$C_D = D,\ \{\text{Компетенция}\};$$

В процессе выполнения проектной работы, проектировщик или их группа в проектном процессе играет определенную роль. Эта роль образуется из набора компетенций проектной работы. Таким образом, проектировщик или группа проектировщиков в процессе работы играет набор определенных ролей R_d. Роли в ***WIQA*** представлены в виде справочника ролей R^*.

$$R_d = \{R\};$$

Каждой роли соответствует определенный набор компетенций, а также, она представлена её кодом R_c и наименованием R_n.

$$\text{Роль} = R;$$
$$R = \{\text{Компетенция}\},\ R_c,\ R_n;$$

2.2.4. Назначение задач

Перед началом выполнения проектной работы в организации происходит анализ проекта с целью определения набора потоков проектных работ, результатом выполнения которых становится реализованный проект. В процессе этого анализа каждый поток работ представляется в виде набора проектных задач $Z_{p0}...Z_{pn}$, каждая из которых относится к тому или иному проектному процессу.

Процесс проектирования в рамках методологии RUP разбивается на четыре фазы[87]:

- Разработка технического задания;
- Разработка технического проекта;
- Создание системы;
- Внедрение системы.

Эти фазы разнесены по времени, и в каждой из них существуют определенные, присущие только ей задачи в каждом из проектных процессов ***RUP***. Причем наборы этих задач зависят от результатов решения проектных задач предыдущей фазы.

Кроме того, методология ***RUP*** является итеративной. Соответственно, на каждой итерации происходит уточнение требований, предъявляемых к проекту, и, в соответствии с этим уточнением перед коллективом организации неизбежно возникают новые задачи. Эти задачи могут относиться к любому из проектных процессов ***RUP*** и к любой фазе процесса проектирования.

Из этого следует, что в самом начале работы над проектом невозможно полностью сформировать все потоки работ проектного процесса. И каждый из этих потоков работ может пополняться новыми работами, соответствующими решению возникших новых задач. Поэтому можно говорить о том, что набор потоков работ проекта является динамической структурой, которая непрерывно изменяется во время проектного процесса.

Для осуществления поддержки динамического управления набором потоков работ проекта в комплекс средств ПКУ включена подсистема назначения задач, схематически изображенная на рисунке 3.8.

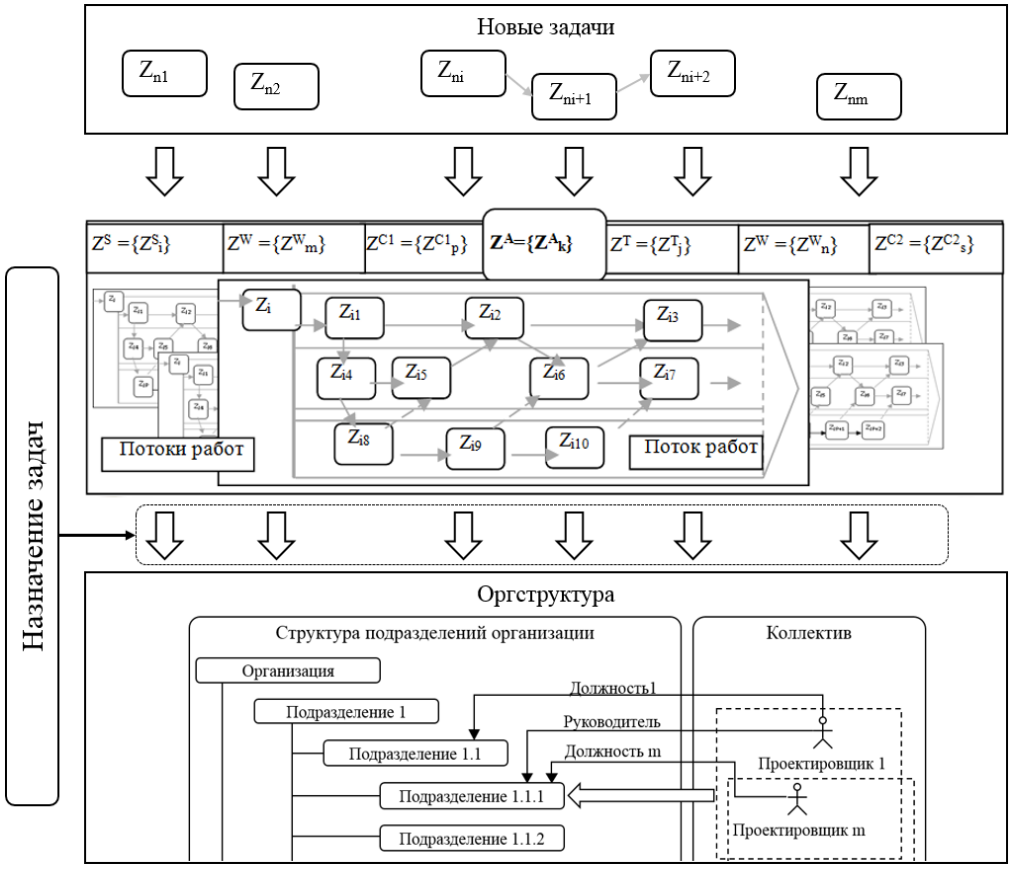


Рис. 2.6. Назначение задач

Из рисунка 3.8 видно, что подсистема назначения является ответственной за формирование связи между потоками работ и оргструктурой.

Проект в среде ***WIQA*** состоит из набора проектных работ, оформленных в виде задач. Набор задач проекта, каждая из которых выполняется определенным участником проектного процесса, обеспечивает формирование ***рабочей силы*** (workforce) организации.

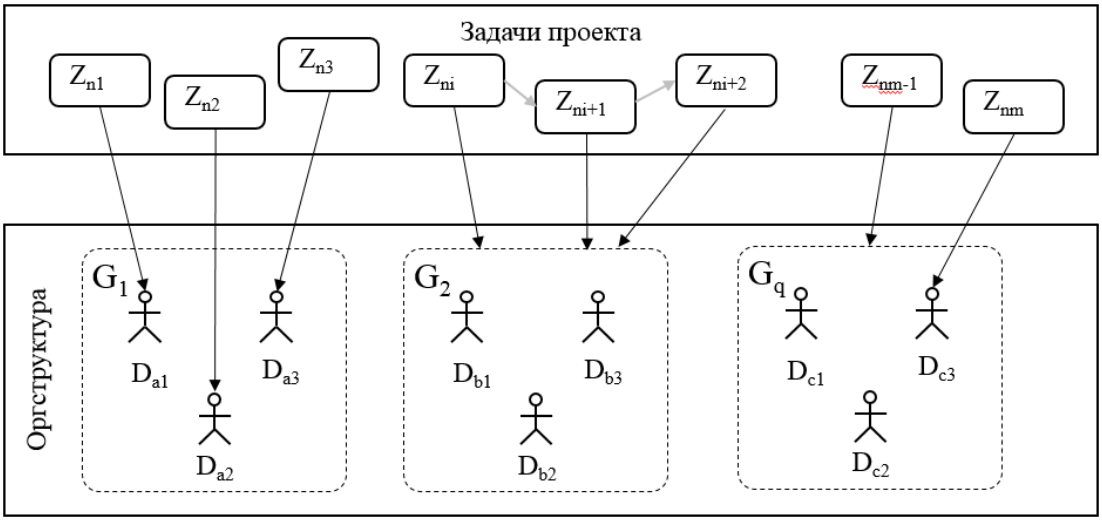


Рис. 2.7. Рабочая сила

Рабочая сила организации схематично представлена на рисунке 2.7. Каждому сотруднику приписывается определенный набор задач, выполняя каждую из которых сотрудник играет ту или иную роль в организации.

```
Рабочая сила = K*, Набор назначений;
Набор назначений = {Назначение};
```

Кроме отдельных сотрудников, задачи, особенно состоящие из множества подзадач, также могут быть также могут быть назначены группе сотрудников ***G***.

Под назначением здесь подразумевается сотрудник или группа и ассоциированный с ним набор задач Z^*_d

$$Назначение = (D \mid G), Z^*_d;$$

В процессе проектной деятельности неизбежно возникают новые проектные задачи, которые необходимо включить в проект и связать с их выполнением определенных сотрудников. Поэтому можно говорить о том, что рабочая сила проекта является динамической структурой, на которой определена операция включения в нее нового назначения.

Для включения новой задачи ***Zn*** в набор назначений:

```
Набор назначений = NP;
NP = NP, "включение", Zn;
```

После выполнения операции "включение":

$$NP = NP, Zn;$$

После того, как новая задача возникла в проекте, появляется необходимость назначить её выполнение участнику или их группе. При таком назначении происходит выбор соответствующей роли проектировщика, поэтому можно говорить об образовании новой пары ***<Задача, Роль>*** и, соответственно, нового назначения

$$Назначение = D \mid G, Zn;$$

2.2.5. Отображение подсистемы контроля поручений на память WIQA

Практически все современные проектные организации не обходятся без исполнения подчиненными поручений руководства. Руководство, формируя план работ над проектом, осуществляет не только назначение сотрудников, которые будут заниматься их исполнением, а также – устанавливает сроки выполнения той или иной проектной работы. Кроме планирования, перед руководством проектной деятельности стоит задача донесения своих планов до исполнителей проектной работы, а также – контроль за её исполнением. Для достижения этих целей в среде ***WIQA*** в комплекс средств ПКУ была включена система контроля исполнения поручений. Она позволяет своевременно и качественно выполнять поручения, содержащиеся в документах, получать аналитическую информацию, необходимую для оценки деятельности подразделений и отдельных исполнителей, помогает оптимизировать работу аппарата управления, дисциплинирует работников, повышает ответственность исполнителей за порученное дело.

Подсистема контроля поручений позволяет представить план проектных работ в виде набора поручений.

Можно сказать, что подсистема контроля поручений дополняет собой процесс назначения задач, при этом характеризуется набором атрибутов,

таких, как причина, содержание, автор, руководитель, тип, приоритет, контролеры, снято (флаг снятия поручения с контроля), дата исполнения, дата переноса, оценка и статус поручения. Таким образом, происходит формирование поручения ***TC***, представляемого следующим образом:

```
Поручение = TC
TC = Назначение, Атрибуты;
Атрибуты = причина, автор, руководитель, тип,
         приоритет, контролеры, снято, дата
         исполнения, дата переноса, лицо резолюции,
         оценка, статус;
статус = На выполнении | выполнено | закрыто;
```

Поручения отображаются на память ***WIQA*** в виде таблицы поручений ***TC****

$$TC^* = \{Поручение\};$$

При этом ***TC**** в QA-память ***WIQA*** отображается как QA-объект TC^{*qa}, состоящий из одного вопроса "Поручения" Q_{TC}, включающего в себя вопросы назначения $Q_{назн}$ – вопрос субъекта назначения Q_{DG} и вопрос объекта назначения Q_Z, и набор вопросов Атрибуты $Q_{Атр}$, каждый из которых соответствует определенному атрибуту поручения:

$$TC^{*qa} = Q_{TC}, "\downarrow", Q_{назн}, Q_{Атр};$$
$$Q_{назн} = Q_{DG}, Q_Z;$$
$$Q_{Атр} = Q_{причина}, Q_{автор}, Q_{руководитель}, Q_{тип}, Q_{приоритет}, Q_{контролеры}, Q_{снято}, Q_{дата\_исполнения}, Q_{дата\_переноса}, Q_{оценка}, Q_{лицо\_резолюции}, Q_{статус}.$$

Сами поручения при этом отображаются в виде ответов на вопросы назначения и атрибутов. Таким образом, ***Поручение*** в QA-памяти WIQA раскрывается как $Поручение^{QA}$, представляющее собой набор ответов на эти вопросы.

$$Поручение^{QA} = (Q_{DG}, "\leftarrow", A_{DG}), (Q_Z, "\leftarrow", A_Z), (\{Q_{Атр}\}, "\leftarrow", \{A_{Атр}\});$$

2.3. Организация процесса программно-картотечного управления

2.3.1. Отбор задач для оперативного выполнения

В первой главе, когда было представлено обобщенное введение в Kanban и Scrum-технологии, было отмечено, что для каждой из них должен быть сформирован пакет задач, который в этих технологиях принято обозначать словом **Бэклог** (Backlog). В этих технологиях в понятии бэклога есть некоторые отличия, в то же время они обладают общими чертами. Количество задач бэклога в обеих технологиях является ограниченным и этот набор задач выбирается в соответствии с расставленными приоритетами набора задач проекта. В технологии Kanban размер каждой очереди ограничен. Технология Scrum не ставит принципиальных ограничений на общий объем бэклога и на объем работ каждого отдельного проектировщика, но в то же время этот объем должен быть таким, какой Scrum-команда в состоянии выполнить за заданный отрезок времени.

Таким образом, основным критерием для отбора задач в бэклог является приоритет. Каждый поток работ содержит определенный набор задач, результаты выполнения которых требуется получить первоочередно, таким образом, можно говорить, о том, что приоритет этих задач является наибольшим. Здесь также необходимо учитывать и то, что для выполнения определенных задач требуется выполнить и другие задачи, результаты выполнения которых необходимы. Таким образом, можно говорить о том, что в бэклог Bl попадают задачи:

$$Bl = (\{P_f,\ Z_{max}\},\ \{BZ_{necc}\},\ \{P_{time},\ Z_{time}\},\ \{BZ_{Tnecc}\});$$
$$BZ_{necc} = \{P_{f+n+i},\ Z_{necc},\ B_{Znecc}\};$$
$$BZ_{Tnecc} = \{P_{time+m+j},\ Z_{time},\ B_{ZTnecc}\};$$

Здесь P_f – приоритет задач Z_{max}, выполнение которых является необходимым в течение этапа разработки, для которого формируется бэклог. В процесс выполнения этих задач вовлекаются результаты других задач, из

которых отбираются блок ещё не выполненных задачи Z_{necc}, каждая из которых тоже может иметь задачи, от выполнения которых она зависит. При этом их приоритет P_{f+n+i} должен быть больше приоритета задач Z_{max}, чтобы эти задачи оказались выполнены ранее.

Кроме этих задач, бэклог проекта наполняется задачами Z_{time}, которые соответствуют определенному критерию, зависимому от особенностей проекта. Эти задачи образуют блок задач BZ_{Tnecc}. Таким критерием может быть, например, назначенный срок выполнения задач t_{max} с помощью подсистемы контроля поручений, который не должен превышать определенного значения. Приоритет этих задач меньше приоритета задач "первой необходимости" Z_{max}. В связи со спецификой потока работ каждая из этих задач может использовать в своем решении результаты выполнения других задач Z_{tnecc}. Приоритеты этих задач $P_{time+m+j}$ также должны быть больше приоритетов задач, являющихся зависимыми с целью их более раннего выполнения. Здесь важным является такой момент, чтобы общий объем работ бэклога не превышал ограничения, устанавливаемого методологией проектирования.

Следовательно, одним из верных вариантов распределения является линейный:

- Наибольший приоритет P_{f+n} имеют задачи Z_{necc}.
- После их выполнения должны выполняться задачи Z_{max}.
- Далее идет выполнение задач Z_{tnecc}.
- И в последнюю очередь выполняются задачи Z_{time}.

Поэтому справедливым является следующее неравенство:

$$P_{f+n} > P_f > P_{time+m} > P_{time};$$

Такой вариант распределения представлен на рисунке 2.8.

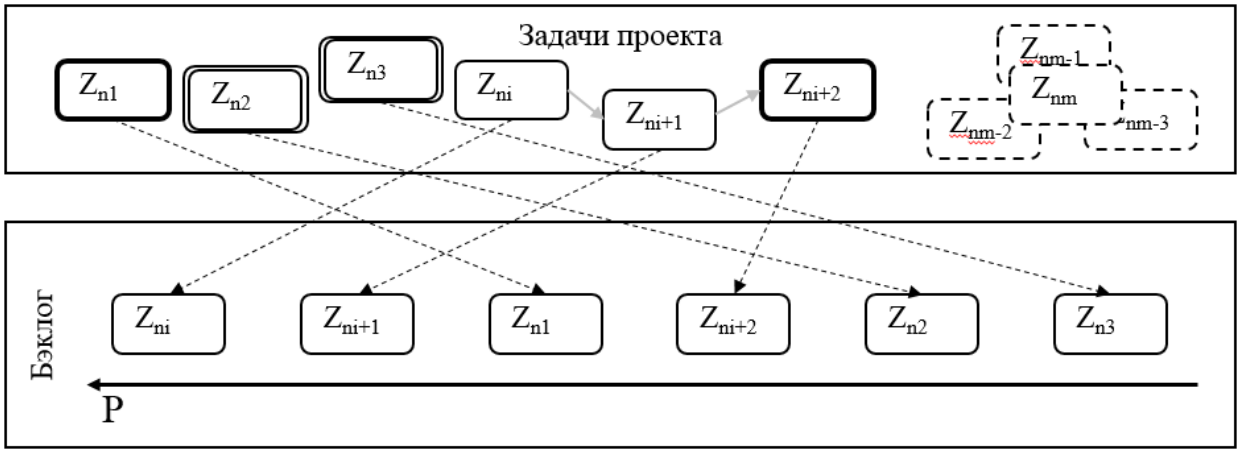


Рис. 2.8. Линейный вариант распределения задач в бэклоге

На рис. 2.8 задачи Z_{max} выделены жирным контуром. Задачи Z_{time} обозначены двойным контуром. Как видно из этого рисунка, сначала должнвы выполняться все задачи Z_{necc}, и только после этого начинается процесс выполнения задач Z_{max}.

Другим вариантом распределения приоритетов является блочный, при котором формируются блоки ***Zb***, состоящие из задач Z_{max} и зависимых от них задач Z_{necc}, а также блоки ***Zbt***, состоящие из задач Z_{time} и зависимых от них задач Z_{necc}:

$$Bl = (\{Zb\}, \{Zbt\});$$
$$Zb = (Pb, (P_f, Z_{max}, BZ_{neccb}));$$
$$Zbt = (Pbt, (P_{time}, Z_{time}, BZ_{Tneccb}));$$

Блоки задач BZ_{neccb}, и, соответственно, BZ_{Tneccb}, содержат задачи, от выполнения которых, соответственно, зависит выполнение задач P_{Zmax} и P_{Ztime}. Эти блоки формируются аналогично блокам задач ***Zb*** и ***Zbt*** и содержат свои подблоки задач, от которых зависит их выполнение.

$$BZ_{neccb} = (P_{neccb}, (P_{f+n+i}, Z_{neccb}, BZ_{neccb}));$$
$$BZ_{Tneccb} = (P_{Tneccb}, (P_{f+n+i}, Z_{neccb}, BZ_{neccb}));$$

Блочный вариант распределения приоритетов задач бэклога показан на рисунке 2.9.

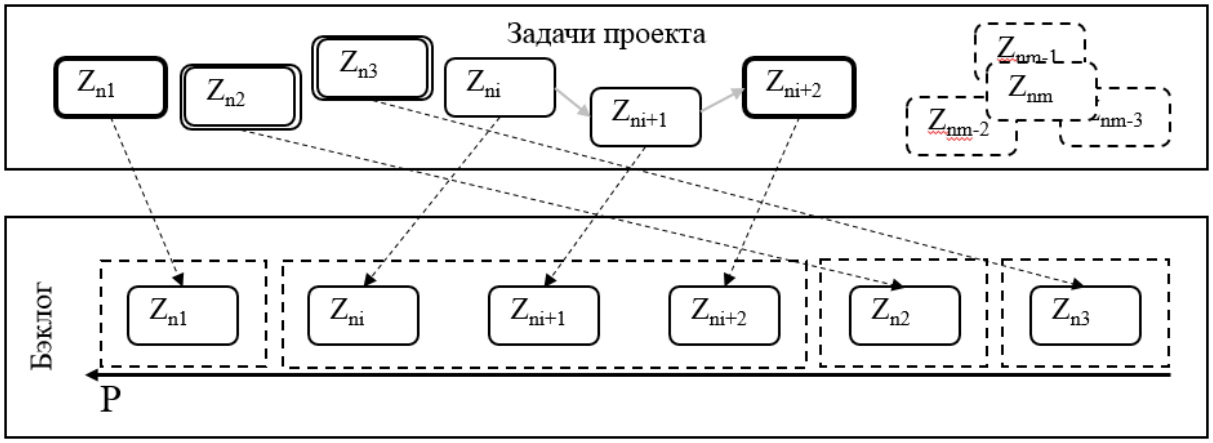


Рис. 2.9. Блочный вариант распределения задач в бэклоге

У каждого блока в этом случае формируется свой приоритет ***Pb/Pbt***, а приоритеты задач для каждого ***k***-го блока соответствуют следующим условиям:

$$Pb_k > P_f > P_{neccb1}..P_{neccbn} > Pb_{k+1};$$
$$Pbt_k > P_{time} > P_{Tneccb1}..P_{Tneccbn} > Pbt_{k+1};$$

Таким образом, отбор этих задач несложно запрограммировать по содержимому таблиц контроля поручений, отобрать перечень задач по таком-то параметру.

В процессе формирования бэклога происходит построение распределенного в соответствии с их приоритетами списка задач, подлежащих выполнению в течение заданного времени. В качестве примера на рисунке 2.13 приведен алгоритм отбора задач с распределением их по приоритетам по линейной схема, для которых условием отбора задач Z_{time} является дата выполнения не позднее одного месяца.

Комплекс средств ПКУ предполагает программирование алгоритма формирования бэклога на языке псевдокодового программирования L^{WIQA}.

2.3.2. Отображение Kanban-процесса в ПКУ

Все участники проектного процесса видят доску Kanban и содержащиеся на ней карточки. При этом каждый участник может найти представляющие его работу карточки на текущем шаге проектного процесса, посмотреть их и таким

образом увидеть набор задач, которые предстоит ему выполнять параллельно на данном этапе проектного процесса.

Система Kanban применяется, в том числе и для разработки программных продуктов. Поскольку в среде ***WIQA*** осуществляется коллективное проектирование автоматизированных систем, в данном процессе он также оказывается применимым. В контексте ПКУ рассматривается реализация средство визуализации доски Kanban в среде ***WIQA***.

Подсистема Kanban в системе ПКУ визуализирует доску Kanban для группы сотрудников ***G*** в виде таблицы ***KT***. Доска делится на набор колонок - шагов, соответствующих этапам выполнения проектного процесса.

$$KT = G, Колонки;$$

Шаги представляют собой этапы проектного процесса, которые доска Kanban отображает в виде столбцов. Каждый шаг содержит в себе карточки, соответствующие задачам члена группы ***GD***, которым может быть как подгруппа, так и сотрудник.

$$Колонки = \{Колонка\};$$
$$Колонка = I, \{GD\};$$

I в данном случае характеризует индекс колонки, соответствующий шагу выполнения проектной работы. Таким образом формируются ячейки таблицы, расположенные на доске Kanban. На рисунке 2.10 представлена схема доски Kanban в версии, реализованной в средствах ПКУ:

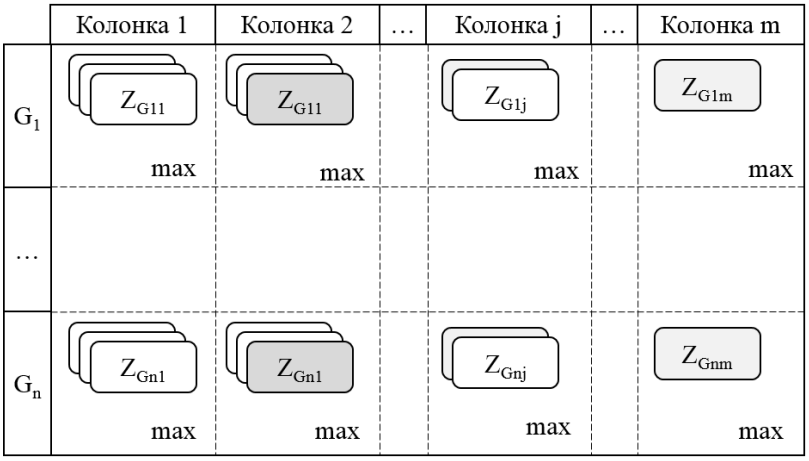


Рис. 2.10. Доска Kanban в ПКУ

Каждая ячейка содержит в себе набор карточек, соответствующих проектным задачам, выполняемым членом группы на определенном этапе работы над проектом. Таким образом, член проектной группы видит набор задач, который ему будет необходимо выполнить параллельно в течение времени шага проектного процесса.

$$GD = \{Карточка\};$$
$$Карточка = Cr;$$

Карточка Kanban характеризуется определенным набором атрибутов. Карточка Cr является визуальным представлением задачи, поэтому одним из важнейших её атрибутов является ссылка Z_{ref} на задачу, которую она визуализирует. Для задачи карточка описывает её имя $Desc$. Также карточка характеризуется цветом C_c. Каждая карточка может быть отмечена флажком Cr.

$$Cr = Z_{ref},\ Z_{desc},\ Z_c,\ C_f;$$

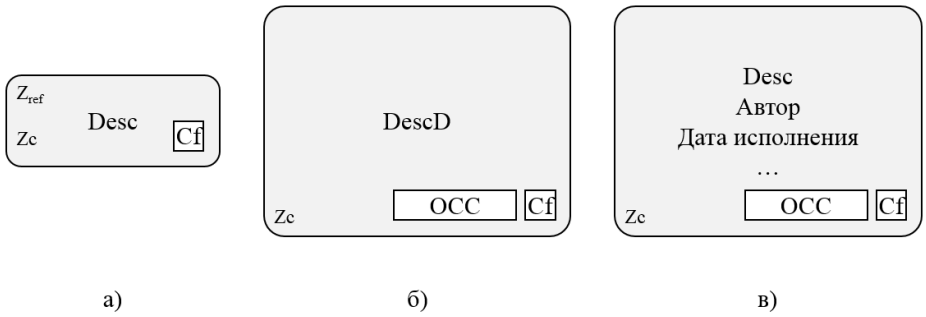
Флажок может быть установлен или снят:

$$Cf = Установлен \mid Снят;$$

Каждая карточка может быть раскрыта, т.е. представлена в увеличенном масштабе, отображая детальную информацию о задаче.

$$Раскрытая\ карточка = Cr^O;$$
$$Cr^O = Cr,\ "раскрытие";$$

Раскрытая карточка увеличивается в размере и отображает ряд дополнительной информации о задаче, кроме сведений, отображаемых карточкой. Карточка имеет две стороны – лицевую Cr^F и обратную Cr^B, а также – ряд элементов управления OCC, позволяющих перевернуть карточку на другую сторону, перейти к выполнению задачи и отметить задачу выполненной. Лицевая сторона карточки отображает её детальное описание $Desc_D$



а) б) в)

Рис. 2.11. Карточки Kanban в ПКУ

На рисунке 2.11 схематично изображены карточки Kanban. 2.11, а) схематично изображает карточку ***Cr***. На 2.11, б) представлена лицевая сторона карточки Cr^F, а 2.15, в) показывает схему обратной стороны карточки карточки Cr^B.

$$Cr^O = (Cr^F \mid Cr^B), Cf, OCC;$$
$$Cr^F = Desc_D;$$
$$Cr^B = Desc, Автор, Дата\ исполнения;$$
$$OCC = Перевернуть, К\ задаче, Выполнена;$$

С помощью этих элементов управления сотрудник может перевернуть карточку, переключив отображение лицевой стороны на обратную или наоборот, перейти к выполнению задаче, а также отметить выполнение задачи.

Доска Kanban отображается на память ***WIQA*** в виде двух таблиц: таблицы задач ***TZ****, содержащей сведения о задаче, отображаемые на карточке, и таблицы ассоциаций ***TA****, устанавливающей соответствие между карточкой и положением её на доске Kanban.

В таблице задач каждая строка представлена её идентификатором Z_{DescId} и набором атрибутов Z_{Desc}.

$$TZ* = \{Z_{DescId}, Z_{desc}\};$$

Идентификатор Z_{DescId} в данном случае представляет числовой код, соответствующий описанию задачи для организации ссылки на неё из таблицы ассоциаций.

$$Z_{desc} = Поручение, Desc, Desc_D, Z_{ref};$$

Поручение представляет собой ссылку на поручение из подсистемы

контроля поручений, также представленную в виде натурального числа.

Desc – краткое текстовое описание задачи, $Desc_D$ – детальное текстовое описание.

$$Desc = Строка;$$
$$Desc_D = Строка;$$

Z_{ref} представляет собой ссылку на задачу ***Z***, которую представляет карточка.

Перейдем к представлению таблицы ассоциаций, отвечающей за размещение карточек на доске. Таблица ассоциаций представлена набором строк ***As***, содержащих ссылку на описание задачи Z_{DescId}, номер колонки таблицы, состояние задачи As_{st}, ссылку на группу ***G***, которой назначена задача и отметка о выполнении As_{mk}.

$$TA* = \{As\};$$
$$As = Z_{DescId}, Колонка, As_{st}, G, As_{mk};$$

Описание задачи Z_{DescId} в данном случае представляет собой ссылку на описание задачи из таблицы ***TZ**** посредством его идентификатора.

Шаг соответствует номеру шага проектного процесса, иными словами – номер колонки в таблице доски Kanban, в котором будет располагаться карточка.

Состояние характеризует выполнение задачи и влияет на цвет карточки в отображении.

$$As_{st} = Выполняется \mid Выполнено;$$

G представляет собой ссылку на группу, на которую назначено выполнение задачи. Эта ссылка представляется в виде числового кода. ***G*** может быть как, собственно, группой проектировщиков, так и одним проектировщиком.

$$G = G_{ref};$$

Отметка позволяет отмечать карточки для выполнения групповых операций над ними. Примером такой операции служит формирование отчета по нескольким задачам.

$$As_{mk} = TRUE \mid FALSE;$$

Теперь рассмотрим отображение этих таблиц на QA-память WIQA. В ней для хранения этих таблиц формируется QA-объект $TKanban^{QA}$, представляющий собой вопрос Q_{Kanban}, "БД Kanban", содержащий в себе подчиненные QA-объекты обеих таблиц:

$$TKanban^{QA} = QKanban, "\downarrow", (TZ*^{qa}, TA*^{qa});$$

При этом **TZ*** в QA-память WIQA отображается как QA-объект $\mathbf{TZ*^{qa}}$, состоящий из одного вопроса "Задачи" $\mathbf{Q_{TZ}}$, включающего в себя ряд вопросов, соответствующих атрибутам описания задачи:

$$TZ*^{qa} = Q_{TZ}, "\downarrow", Q_{AtpZ};$$
$$Q_{AtpZ} = QZ_{DescId}, QПоручение, QDesc, QDesc_D, QZ_{ref};$$

Сами описания задач при этом отображаются в виде ответов на вопросы атрибутов описания задачи. Таким образом описание задачи в QA-памяти WIQA раскрывается как $ОписаниеZ^{QA}$, представляющее собой набор ответов на эти вопросы.

$$ОписаниеZ^{QA} = (\{Q_{AtpZ}\}, "\leftarrow", \{A_{AtpZ}\});$$

Теперь рассмотрим отображение **TA*** на ***QA***-память ***WIQA***.

TA* в QA-память WIQA отображается как ***QA***-объект $TA*^{qa}$, состоящий из одного вопроса "Ассоциации" Q_{TA}, включающего в себя ряд вопросов, соответствующих атрибутам ассоциации:

$$TA*^{qa} = Q_{TA}, "\downarrow", Q_{AtpA};$$
$$Q_{AtpA} = QZ_{DescId}, QКолонка, QAs_{st}, QG, QAs_{mk};$$

Ассоциации при этом отображаются в виде ответов на вопросы атрибутов ассоциации аналогично тому, как это организовано для описаний задач. Таким образом ассоциация в ***QA***-памяти ***WIQA*** раскрывается как $Ассоциация^{QA}$, представляющее собой набор ответов на эти вопросы.

$$Ассоциация^{QA} = (\{Q_{AtpZ}\}, "\leftarrow", \{A_{AtpZ}\});$$

2.3.3. Отображение Scrum-процесса в ПКУ

Scrum представляет собой итеративный, инкрементальный процесс для

разработки любого продукта или управления любой работой. Он производит потенциально готовый продукт после каждой итерации.

Для формирования набора задач итерации и команды, выполняющей работы над этими задачами учитываются особенности выполнения командами предыдущих итераций, что приводит к возможности сформировать новую итерацию более рационально.

Итерация в Scrum именуется **спринтом** (Sprint). Длительность такого спринта фиксирована. Для формирования такого спринта производится выбор задач из общего объема проектных задач, подлежащих выполнению в текущий момент времени для выполнения Scrum-командой. Этот общий объем задач проекта в терминологии Scrum называется **Бэклог проекта** (Project Backlog). Набор задач для выполнения в течение времени спринта называется **бэклогом спринта** (Sprint Backlog).

Итеративный процесс работы S_{proc} по методологии Scrum состоит из следующих основных частей:

- Формирование бэклога спринта (SB_f);
- Выполнение спринта (SB_w);
- Расчет метрик спринта и их анализ (SB_a).

$$S_{proc} = \{SB_f, SB_w, SB_a\};$$

SB_f заключается в формировании спринта **SS**, представляющего собой бэклог спринта SB и команду, выполняющую спринт ST.

Для формирования бэклога спринта используются метрики, полученные в результате анализа выполнения предыдущих спринтов. Таким образом, можно говорить о возникновении обратной связи, соединяющей результаты метрик с новым бэклогом спринта.

$$SS = SB, ST;$$

Бэклог спринта в свою очередь представляет собой набор задач, каждая из которых имеет оценку трудозатратности Zv, распределенный по

приоритетам. Каким именно образом осуществляется формирование бэклога спринта, описано в п.3.3.1.

$$SB = \{SB_a\}, \{Z, Zv\};$$

Кроме бэклога спринта, обратная связь также воздействует на формирование команды. Например, в процессе выполнения спринтов может выясниться, что проектировщики работают недостаточно эффективно из-за выполнения непривычных им ролей. В этом случае для рационализации проектного процесса может потребоваться изменение состава команд, например, осуществить частичный обмен сотрудников в разных командах.

$$ST = \{\{SB_a\}, ST\};$$

Поэтому для формирования команды ***ST*** также должны учитываться метрики, полученные в результате выполнения предыдущих спринтов. На рисунке 2.12 схематически показаны обратные связи в Scrum-процессе.

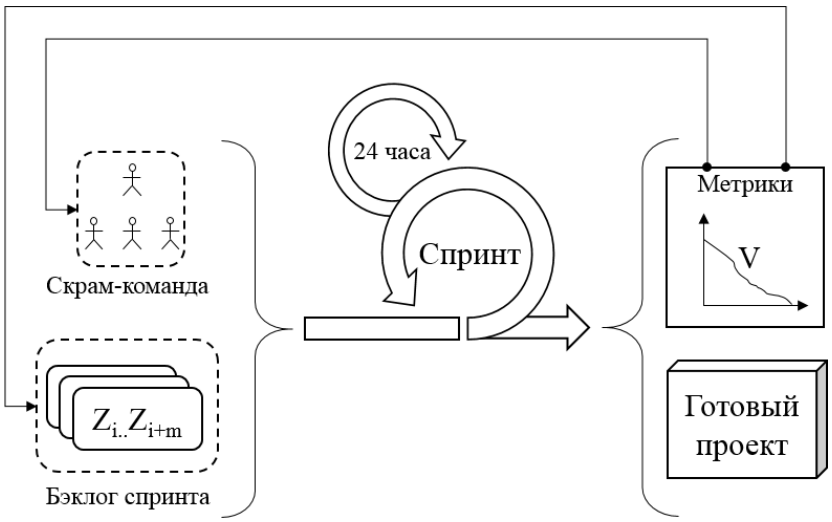


Рис.2.12. Обратная связь в Scrum-процессе

Команда по правилам Scrum не должна содержать более ***9*** членов команды (рекомендуется ***7***, +\- ***2*** участника). Меньшие команды более продуктивны хотя бы по той причине, что между участниками возникает меньше путей коммуникации.

Следующая формула позволяет рассчитать количество коммуникационных путей в команде, в которой все участники равноправны.

$$CP = [N * (N - 1) / 2],$$

где ***CP*** – количество путей коммуникации, ***N*** – количество участников команды.

В процессе планирования спринта происходит оценка трудозатрат на выполнение каждой задачи. Для такой оценки команда может воспользоваться техникой, получившей название **Покер планирования** (Planning Poker). Суть её заключается в том, что каждый член команды имеет набор карточек, пронумерованных числами ряда Фибоначчи (в Scrum используются значения ***1, 2, 3, 5, 8, 13, 21, 36, 57***). Использование этих чисел позволяет легче оценить трудозатратность задачи, так как с каждым следующим числом его значение резко возрастает. Каждый проектировщик при этом оценивает трудозатраты на выполнение задачи одним из этих чисел и показывает соответствующую карточку. Проектировщики, показавшие очень низкую и очень высокую оценку задачи объясняют свое решение.

После оценки пользователем происходит вычисление трудозатратности задачи в **единицах очков** (Story Points) с учетом коэффициентов:

$$Zv = X (1 + (Diff + DF + EF + FMC)),$$

где ***X*** – оценочная трудозатратность задачи, ***Diff*** – сложность требований и используемых технологий. Она определяется в соответствии со следующей таблицей:

Таблица 2.1. Сложность требований и технологий

Сложность	**Коэффициент**
Простая (Simple)	0
Усложненная (Slightly Complicated)	0.1
Сложная (Complicated)	0.2
Очень сложная (Very Complicated)	0.6

DF – **Замедление** (Drag Factor) – определяется в зависимости от навыков группы и опыта её совместной групповой работы. Таким образом, для каждой задачи формируется атрибут её объема работы Z_V.

$$Zs = Z, Zv;$$

Далее представлена таблица для определения значения замедления:

Таблица 2.2. Коэффициент Drag Factor

DF	**Опыт командной работы**	**Технические навыки**	**Бизнес-навыки**
0.8	< 3 месяцев	Низкие	Низкие
0.75	< 3 месяцев	Низкие	Средние
0.7	< 3 месяцев	Низкие	Высокие
0.75	< 3 месяцев	Средние	Низкие
0.5	< 3 месяцев	Средние	Средние
0.5	< 3 месяцев	Средние	Высокие
0.75	< 3 месяцев	Высокие	Низкие
0.5	< 3 месяцев	Высокие	Средние
0.35	< 3 месяцев	Высокие	Высокие
0.6	< 1 года	Низкие	Низкие
0.55	< 1 года	Низкие	Средние
0.5	< 1 года	Низкие	Высокие
0.55	< 1 года	Средние	Низкие
0.3	< 1 года	Средние	Средние
0.25	< 1 года	Средние	Высокие
0.5	< 1 года	Высокие	Низкие
0.25	< 1 года	Высокие	Средние
0.2	< 1 года	Высокие	Высокие
0.5	> 1 года	Низкие	Низкие
0.45	> 1 года	Низкие	Средние
0.4	> 1 года	Низкие	Высокие
0.45	> 1 года	Средние	Низкие
0.35	> 1 года	Средние	Средние
0.20	> 1 года	Средние	Высокие
0.4	> 1 года	Высокие	Низкие
0.2	> 1 года	Высокие	Средние
0	> 1 года	Высокие	Высокие

На основе определенных объемов работ каждой задачи происходит расчет общего объема работ спринта ***Sv***. Этот объем работы учитывает не только сумму временных затрат задач, но также и ряд коэффициентов:

EF представляет собой коэффициент – фактор среды. ***EF*** равен ***0***, если вся группа располагается в одном кабинете или офисе, иначе он равен ***0.1***

FMC – фактор множественности команд. ***FMC*** также добавляет ***0.1***, если

в спринте задействовано несколько устоявшихся отдельных групп, не работавших ранее вместе как одна группа.

Теперь перейдём к описанию процесса выполнения спринта.

SB_w представляет собой итеративный процесс выполнения работы над спринтом, в течение которого выполняются краткосрочные **24-часовые спринты**:

$$SB_w = \{SB_t\};$$

Во время этого краткосрочного 24-часового спринта происходят следующие процессы:

- Ежедневный Scrum (Daily Scrum) D_{Sc};
- Выполнение проектных задач;

Таким образом,

$$SB_t = D_{Sc}, \{(Z_i, \text{"выполнение"})\};$$

В свою очередь, ежедневный Scrum представляет собой **15-минутное обсуждение**, в ходе которого происходит оценка проделанной работы и внесение изменений в диаграмму выгорания $\mathbf{P_{ob}}$. Диаграмма выгорания представляет собой график, на котором по оси абсцисс отмечены дни – каждая отметка соответствует ежедневному Scrum, а по оси ординат отмечается оставшийся объём работы $Sv - Z_{Vi}$ для i-го дня.

$$D_{Sc} = (P_{ob}, \text{"отметка"});$$
$$P_{ob} = Sv, \{Sv - Z_{Vi}\};$$
$$\text{"отметка"} = P_{ob}, Sv - Zv_n$$

Здесь Z_{Vi} представляет собой выполненный на текущий момент объем работ спринта.

После завершения времени выполнения спринта происходит вычисление метрик. К этим метрикам относятся:

- Скорость работы сотрудника (Individual Velocity, ***IV***);
- Скорость работы команды (Team Velocity, ***TV***);
- Оценка стоимости выполненной работы (Value Delivered, ***VD***);

- Оценка планирования (On-time Delivery, ***TD***).

Таким образом,

$$SB_a = (IV, TV, VD, TD), \text{"вычисление"};$$

Метрика скорости работы сотрудника определяется как отношение трудозатратности выполненных им работ к длительности спринта.

$$IV = \frac{\sum_{i=0}^{nd} Zv_i}{t},$$

где ***nd*** – количество выполненных задач проектировщика в спринте,

Zvi – трудозатратность задачи **i**,

t – продолжительность спринта.

Скорость работы сотрудника является индивидуальной характеристикой сотрудника, зависящей от набора задач и особенностей их оценки. Её нельзя применять для сравнения производительности сотрудников.

Скорость работы команды определяется как отношение выполненного объема работ в единицах трудозатратности к продолжительности спринта:

$$TV = \frac{\sum_{i=0}^{ns} Zv_i}{t},$$

где ***ns*** – общее количество выполненных задач в спринте,

Zvi – трудозатратность задачи ***i***,

t – продолжительность спринта.

Скорость работы команды также является индивидуальной характеристикой команды, измеренной в единицах, зависимых от самой команды. Поэтому её нельзя использовать для сравнения производительности команд.

Оценка стоимости выполненной работы осуществляется совместно **стейкхолдерами** (stakeholder) проекта. Ему демонстрируют историю выполнения спринта и он ставит оценку ***M*** эффекта от использования результата выполнения каждой задачи спринта в денежном эквиваленте или в любой другой оценке для каждой задачи.

Таким образом,

$$VD = \{Z, M\};$$

Эту метрику можно использовать для определения приоритетов при формировании бэклога нового спринта.

Оценка планирования определяет то, насколько тщательно были произведены оценки скорости работы команды и объемов работ на спринт.

Она высчитывается следующим образом: Устанавливается максимальное количество баллов TD_{max}, которым команда оценивается в том случае, если весь объем спринта оказался выполненным точно в срок. За каждые **n** дней спринта, на которые весь его объем работ оказался выполнен раньше, из этого количества вычитается x баллов. Если команда не успевает выполнить спринт в срок, из TD_{max} вычитается y баллов за каждые m невыполненных единиц трудозатрат. При этом нужно учитывать, что результат TD не может оказаться меньше нуля. В этом случае он приравнивается нулю, что говорит или о неправильном подборе коэффициентов TD_{max}, x и y, или о крайне неудачном планировании спринта.

$$TD = TDmax - nx - ny;$$

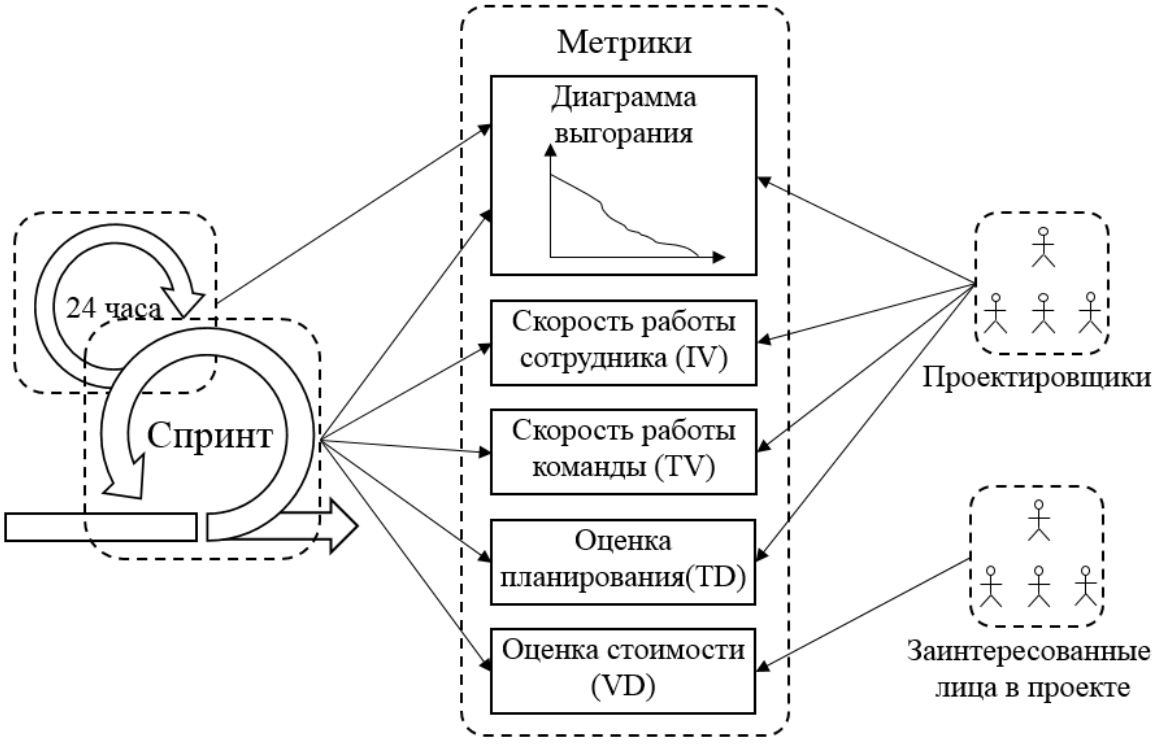


Рис. 2.13. Формирование метрик

Полученные в результате выполнения спринтов метрики влияют как на формирование новой Scrum-команды для работы над следующим спринтом, так и на формирование бэклога следующего спринта:

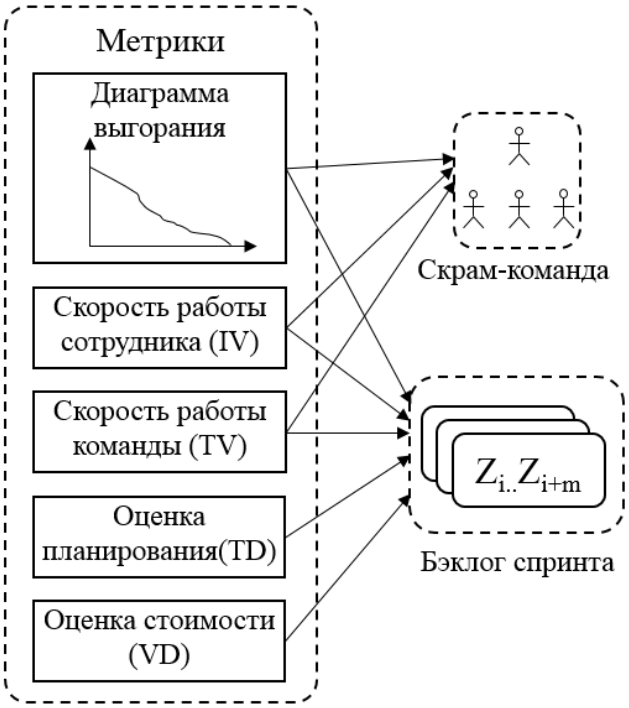


Рис. 2.14. Связь метрик с построением нового спринта

Стоит отметить, что список этих метрик можно расширить путем программирования алгоритмов их вычисления на языке $\boldsymbol{L^{WIQA}}$.

2.4. Распараллеливание проектных процессов в ПКУ

2.4.1. Особенности распараллеливания в ПКУ

Поток проектных работ состоит из множества отдельных, связанных между собой работ, представленных в виде задач. Выполнение каждой из задач назначается определенному участнику проектного процесса, который может быть как группой проектировщиков, так и отдельным проектировщиком. Каждая из этих задач состоит из определенного набора подзадач или вопросов. Решая эти подзадачи или отвечая на поставленные вопросы, участник проектного процесса движется по направлению к решению этой задачи. Очевидно, что непрерывно возникают ситуации, когда в один и

тот же период времени перед участником проектного процесса встает необходимость выполнения нескольких задач.

В том случае, если участником проектного процесса является группа проектировщиков, эта группа представляет собой набор отдельных участников, являющихся независимыми интеллектуальными процессорами, каждый из которых может решать свой набор задач. Таким образом, распараллеливание проектных процессов при решении их группой проектировщиков достигается естественным образом путем разделения потока работ проектного процесса на отдельные потоки работ, каждый из которых поручается отдельному члену группы проектировщиков.

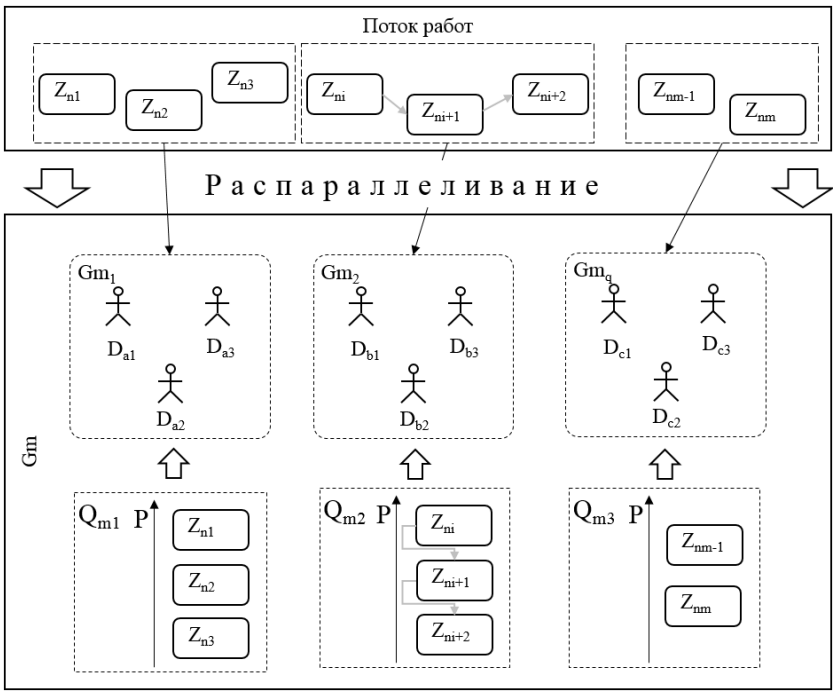


Рис. 2.15. Распараллеливание потока работ по подгруппам

Такой процесс распараллеливания изображен на рисунке 2.15. Здесь показано назначение потока работ группе проектировщиков ***Gm***. При этом происходит разбиение потока работ на отдельные группы, содержащие задачи для назначения их подгруппам. Из каждой такой подгруппы задач происходит формирование набора распределенных в соответствии с их приоритетами задач, подлежащего выполнению определенной подгруппой. При этом

логические связи между результатами выполнения одной задачи и возможностью начать выполнение другую задачу сохраняются. Пример сохранения этих связей представлен в назначении фрагмента потока работ группе $\boldsymbol{Q_{m2}}$. Аналогичным образом распараллеливание происходит и в том случае, если членами группы являются отдельные проектировщики. Такая ситуация представлена на рисунке 2.16.

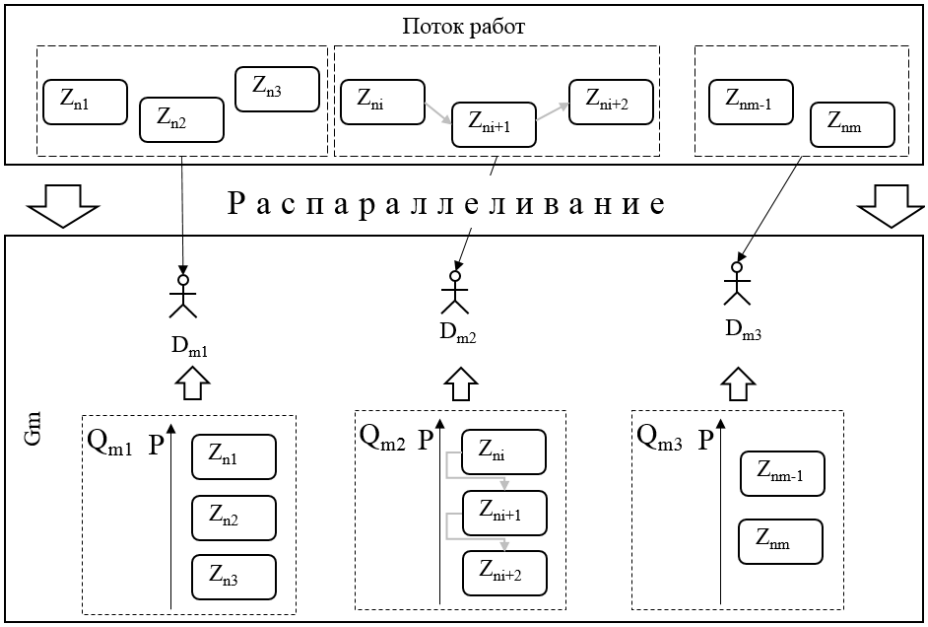


Рис. 2.16. Распараллеливание потока работ по проектировщикам

В случае распараллеливания задач по проектировщикам, каждый проектировщик получает набор проектных задач, выполнение которых поручено именно ему. Этот набор задач формирует очередь. Важно отметить, что задачи могут оказаться зависимыми от выполнения других задач, и в такой ситуации, как она представлена у проектировщика $\boldsymbol{D_{m2}}$ её несложно решить, расставив приоритеты задач для его выполнения таким образом, чтобы он зависимые задачи выполнил позже тех задач, от выполнения которых зависит выполнение других задач. Таким образом, можно говорить о том, что каждая группа проектировщиков или отдельный проектировщик получает свой фрагмент потока проектных работ в виде **очереди задач с приоритетами.**

При распараллеливании может возникнуть такая ситуация, при которой

задачи, связанные между собой зависимостью от результатов оказываются порученными разным исполнителям (рисунок 2.17).

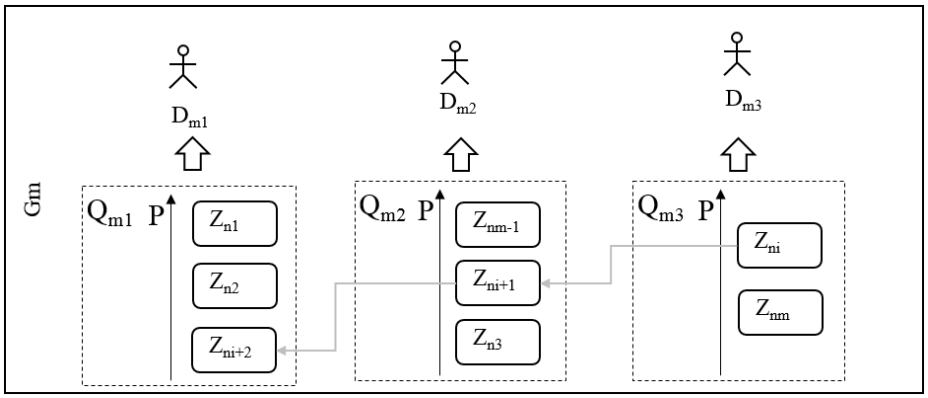


Рис. 2.17. Выполнение связанных задач различными исполнителями

Более того, эти задачи могут оказаться в разных группах ***G*** проектировщиков. Это происходит в том случае, в котором для выполнения этих задач требуются разные компетенции, которыми обладают разные участники проектного процесса, которые могут находиться в разных группах. В такой ситуации необходимо или тщательно планировать проектный процесс таким образом, чтобы у сотрудника не оказалось простоев в связи с невозможностью приступить к выполнению следующей в очереди задачи, или приступить к выполнению следующей задачи после неё, но в этом случае может оказаться, что следующая задача не будет завершена до тех пор, пока отложенная не станет доступной. Здесь также возможны два варианта:

- Изменить приоритет задачи, выполнение которой оказалось невозможным;
- Прервать выполнение следующей задачи и перейти к выполнению ставшей доступной, имеющей более высокий приоритет в очереди задач.

Из первого варианта следует, что очередь задач в процессе проектирования не является статичным объектом, её содержимое динамически изменяется. Такое изменение также может вызвать возникшая в процессе работы над задачами межзадачная связь, которая не была выявлена

во время планирования. Второй вариант требует специфической функциональности средств ПКУ, связанной с обработкой прерываний. Обработка прерываний будет рассмотрена ниже.

Kanban-процесс для каждого шага проектного процесса представляет определенный набор задач для каждого участника. Таким образом, распараллеливание задач между группами в Kanban-процессе отображается в виде соответствующих группам строк Kanban-таблицы. При этом очередь задач оказывается отображенной в виде набора карточек в ячейке таблицы для каждого шага Kanban-процесса. Карточки оказываются отсортированными в порядке их приоритета в очереди, таким образом карточка, лежащая сверху, содержит задачу, выполнение которой имеет наивысший приоритет. Распараллеливание применительно к Kanban-методологии в комплексе средств ПКУ схематично изображено на рисунке 2.18.

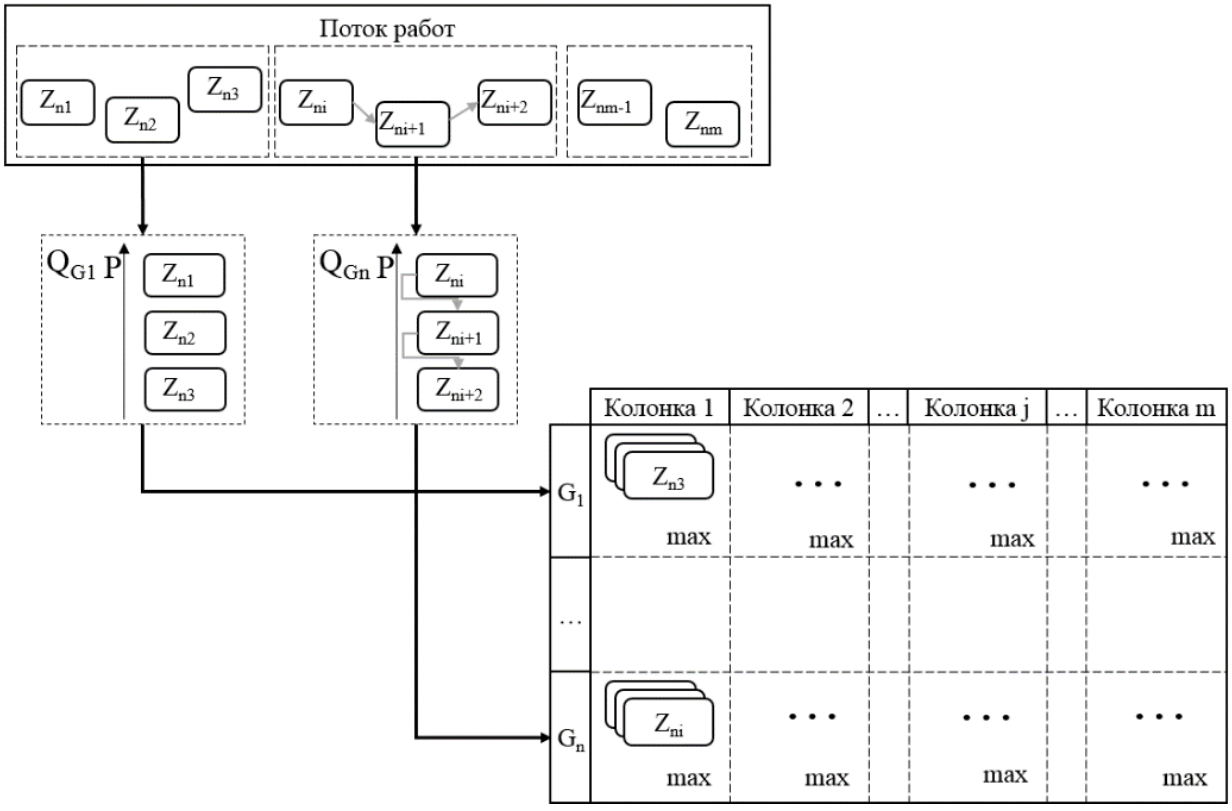


Рис. 2.18. Распараллеливание в Kanban-процессе

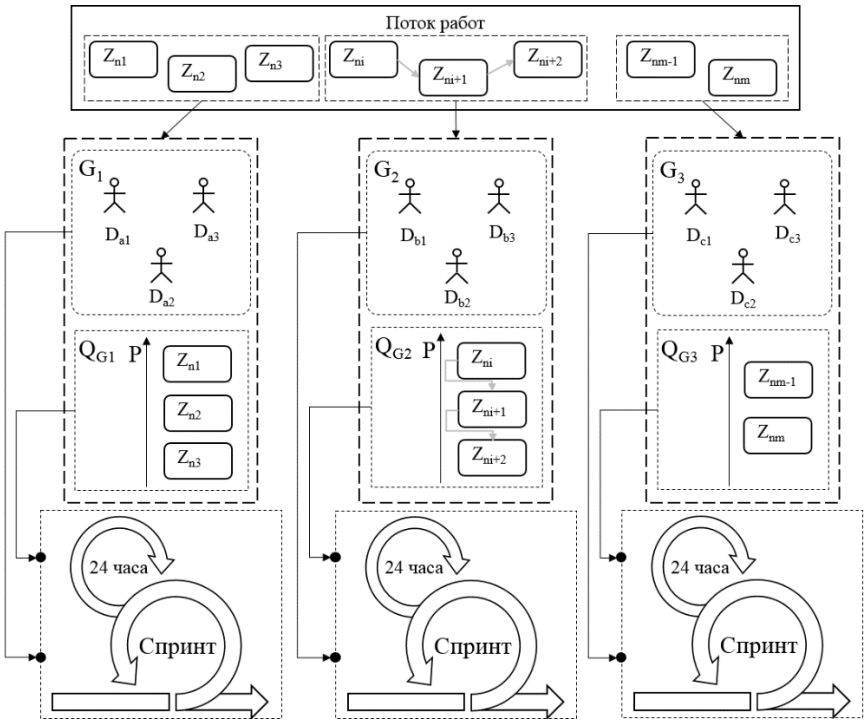


Рис. 2.19. Распараллеливание в Scrum-процессе

Специфика параллельного процесса в Scrum заключается в том, что параллельно могут работать несколько Scrum-команд, каждая из которых выполняет свой спринт. При этом время запуска спринта для каждой команды может быть своим.

Кроме группового параллелизма, который характеризуется наличием нескольких исполнителей, каждый из которых выполняет работу в соответствии со своей очередью задач, также в проектном процессе могут возникать и ситуации индивидуального параллелизма, в которых участнику проектного процесса приходится в одном и том же интервале времени параллельно выполнять несколько проектных работ. Такой вид параллелизма несвойственен интеллектуальному процессору человека и попытки одновременного выполнения нескольких задач могут привести к ошибкам. Поэтому индивидуальный параллелизм осуществляется по аналогии с мультизадачностью, реализованной на ЭВМ. Этот параллелизм осуществляется путем последовательного переключения между задач. Такое переключение сопряжено с прерываниями в человеческой деятельности.

2.4.2. Формализация процессов распараллеливания потоков работ в среде WIQA

Как было отмечено в п. 3.4.1, выполнение потока проектных работ осуществляется несколькими группами проектировщиков ***G*** или отдельными проектировщиками. Для этого поток проектных работ разбивается на отдельные фрагменты WF_F, каждый из которых отдается на выполнение определенной группе или проектировщику. Таким образом, проектный процесс P_p можо представить как набор соответствий фрагмента потока работ его исполнителю.

$$P_p = \{WF_F, (G \mid D)\};$$

Здесь важным является тот момент, что работы ***W***, содержащиеся в ***WFf***, могут быть зависимыми от других работ, или даже от задач, содержащихся в другом ***WFf***. Поэтому ***WFf*** можно представить следующим образом:

$$WFf = \{\{WFf\}, (\{W\}, W)\};$$

В свою очередь, каждая работа характеризуется проектной задачей, описывающей эту работу, и приоритетом этой задачи.

$$W = (P, Z);$$

Таким образом, фрагменты потока работ, назначенные определенному исполнителю представляют собой очереди с приоритетами задач.

$$Q = ((P_1, Z_1), (P_2, Z_2), \ldots, (P_n, Z_n));$$

Особенности формирования приорететов для каждой задачи раскрыты в п. 3.3.1.

В процессе проектной работы вне зависимости от методологии проектирования каждому испонителю приходится последовательно выполнять фрагменты потоков работ, сформированные в виде очередей. Поэтому участие исполнителя в проектном процессе P_{pGD} можно представить как набор его очередей, исполняемых последовательно. Эти очереди также формируют очередь, действующую по принципу **FIFO**.

$$P_{pGD} = ((D \mid G), (Q_1, Q_2, \ldots, Q_n));$$

Параллелизм в таблице Kanban P_{pK} представлен в виде одновременного нахождения на каждом этапе проектного процесса, представленном в виде колонки Kanban-таблицы, нескольких исполнителей, представленных в виде

набора строк Kanban-таблицы. Все карточки, расположенные в столбце Kanban-таблицы характеризуют параллельное исполнение этих задач различными исполнителями в течение шага проектного процесса. Таким образом, набор ячеек столбца таблицы представляет собой паралельное выполнение очередей задач исполнителями.

Таким образом, параллельный процесс Kanbana P_{pK} заключается в последовательном выполнении шагов проектного процесса, на каждом из которых происходит параллельное исполнение задач исполнителями:

$$P_{pK} = (Колонка_1, Колонка_2, \ldots, Колонка_n);$$

Из п. 1.3.2:

$$Колонка = I, \{GD\};$$

В этом случае на ячейку GD отображается очередь Q в виде набора карточек. Таким образом, в контексте управления очередями ячейка GD содержит в себе очередь Q:

$$GD = (Cr_1, "отображение", (P_1, Z_1), Cr_2, "отображение", (P_2, Z_2), \ldots, Cr_n, "отображение", (P_n, Z_n));$$

Параллелизм Scrum-процесса P_{pS} предполагает отображение очереди проектных задач **Q** на бэклог спринта. В этом случае в проектной организации может происходить параллельное выполнение сразу нескольких спринтов одновременно.

Из п.3.3.3 Scrum-процесс представлен следующим образом:

$$S_{proc} = \{SB_f, SB_w, SB_a\};$$

В контексте параллелизма каждая проектная группа осуществляет свой собственный Scrum-процесс

$$P_{pS} = \{G, S_{proc}\};$$

При этом формирование бэклога спринта SB_f осущетсвляется путем отображения на бэклог очереди Q.

$$SB_f = SB, "отображение", Q;$$

Если же говорить об индивидуальном параллелизме P_{pI}, то в этом случае за один период времени t человеку приходится выполнять сразу несколько проектных задач, представленных набором задач ZZ:

$$P_{pI} = D, t, ZZ, \text{"выполнение"};$$

Набор задач содержит в себе задачи, имеющие одинаковый приоритет для выполнения:

$$ZZ = \{Z, Pf\};$$

Каждая из этих задач содержит элементарные неделимые действия **Zd**, последовательно выполняя которые, проектировщик решает задачу. Параллельное выполнение задач осуществлялется путем поочередного выполнения определенных наборов элементарных действий для каждой задачи.

$$Z = \{Zd\};$$

Таким образом, параллельное выполнение набора задач $Z_1, Z_2, \ldots, Z_n$ проектировщиком можно рассматривать как последовательное выполнение элементарных действий ZZd, которые относятся к разным задачам.

$$ZZd = \{(\{Zd_1\}, \{Zd_2\}, \ldots, \{Zd_n\})\};$$

2.4.3. Имитация механизмов массового обслуживания

Поток работ как задание для выполнения его группой проектировщиков можно охарактеризовать как набор **заявок** на выполнение этих работ. Т.к. проектные работы имеют связи между собой, эти заявки поступают в определенной последовательности. Поэтому иможно говорить о потоке работ как о потоке заявок на выполнение этих работ.

Выполняя эти работы, группа тем самым выполняет поток заявок. Таким образом, группа проектировщиков имитирует **систему массового обслуживания** этих заявок.

Существует ряд метрик, вычисляемых в процессе проектной работы по методологии Kanban. Средние значения для этапа проектного процесса этих метрик естественным образом вычисляются как характеристики системы массового обслуживания.

Таблица 2.3. Имитация основных характеристик СМО в ПКУ

Характеристика СМО	**Компонент ПКУ**	**Имитация в ПКУ**	**Комментарий**
Режим обслуживания	Поток работ группы проектировщиков	Параллельно-групповое обслуживание	Одновременно поступает набор работ, что имитирует параллельное

	Поток работ проектировщика		поступление группы заявок в СМО
Время пребывания заявок в очереди	Поток работ группы проектировщиков, поток работ проектировщика	Ограниченное ожидание	Каждая задача должна быть выполнена в течение времени, ограниченного длительностью этапа проектной работы или всего проекта. После превышения этого времени происходит отказ в выполнении задачи
Дисциплина очереди	Поток работ группы проектировщиков, поток работ проектировщика	Очередь с отбором заявок по критерию приоритетности	Каждая проектная задача имеет определенный приоритет относительно других задач. Очередь задач имитирует очередь заявок с приоритетами.
Канальность	Поток работ группы проектировщиков	Многоканальная	Группа проектировщиков, параллельно выполняя набор задач, имитирует работу многоканальной СМО
	Поток работ проектировщика	Одноканальная	Один проектировщик не выполняет в один момент времени более одной задачи
λ (интенсивность поступления заявок)	Поток работ группы проектировщиков	Интенсивность постановки новых проектных задач	Новые задачи ставятся в проекте с определенной интенсивностью, которая имитирует λ
	Поток работ проектировщика	Интенсивность постановки задач	Формирование бэклога определяет интенсивность
μ (интенсивность обслуживания заявки одним каналом)	Поток работ группы проектировщиков	Скорость выполнения задач членом группы	Скорость выполнения задач членом группы имитирует μ в СМО
Время обслуживания заявки $t_{обс}$	Kanban	Метрика Throughput (пропускная способность)	Пропускная способность характеризуется отношением $1/t_{обс}$, где $t_{обс}$ имитирует время выполнения одной задачи
n (число каналов)	Оргструктура	Количество проектировщиков в группе	Каждый проектировщик в группе имитирует канал в СМО
m (длина очереди)	Отбор задач	Длина бэклога	Очередь задач для исполнения имитирует очередь заявок в СМО

Методология Kanban предполагает вычисление ряда метрик, которые

имитируют характеристики СМО. Эти метрики отображены в следующей таблице:

Таблица 2.4. Метрики Kanban как характеристики СМО

Метрика	**Описание**	**Характеристика СМО**
Время цикла (Cycle Time)	Время, которое задача находилась в разработке от момента, когда ей начали заниматься, до момента, когда она прошла фазу конечной поставки	Среднее время обслуживания
Работы в процессе (WIP)	Количество задач, одновременно находящихся в разработке	Среднее число заявок, находящихся в СМО
Время обработки (Lead Time)	Время от появления задачи до ее конечной поставки. Включает Cycle Time и время ожидания в очереди на реализацию.	Среднее время ожидания в очереди на реализацию характеризуется средним временем нахождения заявки в очереди
Пропущенное время (Waste Time)	Включает в себя все время, в течение которого задача ожидает своего выполнения	Среднее время нахождения заявки в очереди каждой СМО
Эффективность (Effectiveness)	Процент времени, которое тратится непосредственно на работу с задачей, а не на ожидания в различных очередях	Отношение среднего время нахождения заявки в очереди к среднему времени пребывания заявки в СМО
Пропускная способность (Throughput)	Количество задач, которое может выполнять команда в единицу времени	Пропускная способность СМО

Группа проектировщиков работает со связанными между собой наборами задач, следовательно, заявки поступают в неё также группами. Поэтому речь идёт о группе проектировщиков как о **СМО с параллельно-групповым обслуживанием.**

Для успешного выполнения всей проектной работы каждая из этих заявок должна быть обработана, поэтому заявки формируют очередь и дожидаются их обработки. Таким образом, группа проектировщиков как СМО является **СМО с ожиданием**.

Причем длительность этого ожидания зависит как от пропускной способности группы, так и от общего объема проектной работы, т.е. от длины

очереди заявок. Проектная организация в общем случае не имеет ограничений на размер и длительность проекта. Однако из самого определения проекта следует, что он является ограниченным по длительности. Таким образом, группа проектировщиков является **СМО с ограниченным ожиданием.**

Расположение заявок в очереди зависит от приоритета проектных работ. Таким образом, дисциплина очереди СМО группы проектировщиков определяется как **очередь с отбором заявок по критерию приоритетности.**

Каждая группа проектировщиков состоит из набора подгрупп или проектировщиков, между которыми распределяются проектные задачи. Таким образом, заявки внутри СМО распределяются на несколько потоков заявок, что позволяет говорить о СМО группы проектировщиков как о многоканальной СМО.

Объединяя сказанное выше, группа проектировщиков имитирует многоканальную СМО с параллельно-групповым обслуживанием, с неограниченным ожиданием.

Схематично группа проектировщиков как СМО представлена на рисунке 2.20.

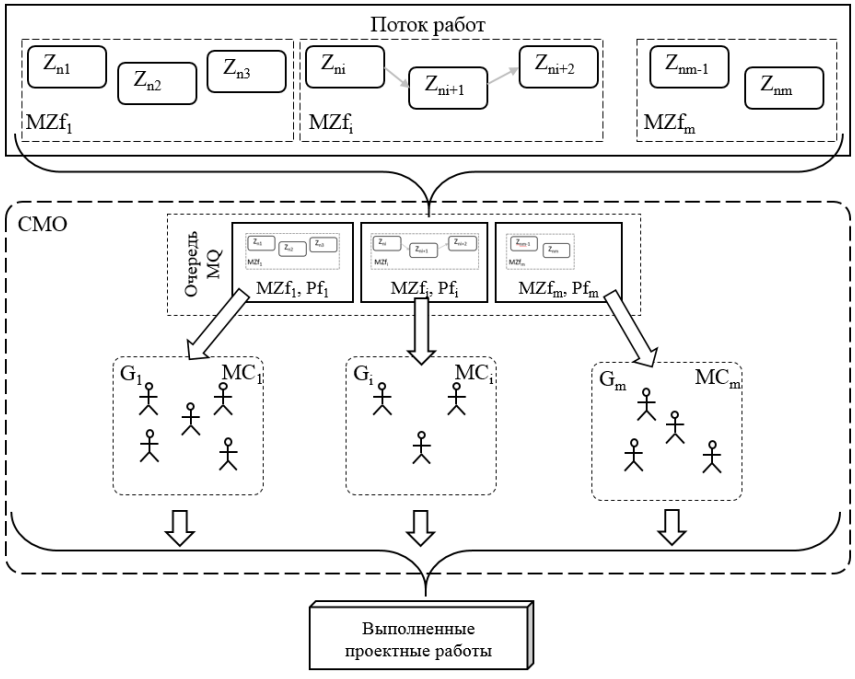


Рис. 2.20. Группа проектировщиков как СМО

СМО группы проектировщиков ***MG*** характеризуется набором заявок ***MZ****, очередью **MQ** и внутренней структурой **MS**:

$$MG = (MZ^*, MQ, MS);$$

MZ* представляет собой фрагменты, на которые разбивается поток работ для выполнения группой в процессе отбора задач для оперативного выполнения (подробнее механизм отбора задач рассмотрен в п. 3.3.1). Каждый из этих фрагментов характеризуется набором компетенций ***C***, которые требуются ждя выполнения этих работ.

$$MZ^* = \{MZf\};$$
$$MZf = \{C\}, \{Z\};$$

MQ представляет собой упорядоченную по приоритетам ***Pf*** очередь заявок, поступающих на вход СМО:

$$MQ = ((MZf_1, Pf_1), (MZf_2, Pf_2), \ldots, (MZf_n, Pf_n));$$

Структура ***MS*** СМО группы проектировщиков представляет собой набор каналов ***MC***, в котором каждый канал является или подгруппой проектировщиков, или проектировщиком – членом проектной группы. Важной особенностью каждого канала является то, что он характеризуется определенным набором компетенций, позволяющих ему решать определенный набор задач. Этот набор компетенций ограничивает возможности канала СМО принимать на обслуживание ту или иную заявку, поэтому заявки распределяются между каналами с учетом возможности их выполнения.

$$MS = \{MC\};$$
$$MC = \{C\}, (G \mid D);$$

Рассматривая отдельного проектировщика, можно отметить, что он, также, как и группа, выполняет набор проектных работ – заявок. Таким образом, отдельного проектировщика также можно рассматривать в качестве имитации системы массового обслуживания. Но, в отличие от группы, эта система будет иметь некоторые особенности.

Во-первых, в очереди работ проектировщика задачи расположены по одной и поступают туда они по отдельности. Поэтому данная СМО работает с единичными заявками.

Во-вторых, проектировщик представляет собой один канал обслуживания. Поэтому СМО проектировщика является одноканальной.

Наконец, в-третьих, в зависимости от используемой методологии проектирования, время на выполнение одного этапа проектного процесса, которым может быть, например, спринт в Scrum, является ограниченным, поэтому при нахождении заявки в очереди на исполнение в течение длительного времени, превышающего время этапа проектного процесса, происходит отказ в обслуживании этой заявки. Таким образом, эта СМО является системой смешанного типа.

СМО проектировщика MD характеризуется, также, как и для СМО группы, набором заявок MZ_D*, очередью MQ_D и внутренней структурой MS_D:

$$MG = (MZ_D*, MQ_D, MS_D);$$

MZ_D* представляет собой набор заявок, характеризующих задачи, требуемыми для выполнения которых компетенциями обладает проектировщик.

$$MZ* = \{(\{C\}, Z)\};$$

MQ_D представляет собой очередь с приоритетами задач сотрудника. Такая очередь была рассмотрена в п. 3.4.2.

$$MQ_D = Q;$$

Структура СМО проектировщика в данном случае представляет собой всего один канал обработки заявок – собственно проектировщика.

$$MS_D = MC;$$
$$MC = D;$$

Имитация механизмов массового обслуживания позволяет внедрить в процесс проектирования наборы метрик, характерные для СМО, такие, как среднее время обслуживания, среднее время в очереди, интенсивности, коэффициенты нагрузки, среднее число свободных и занятых каналов, и

многие другие.

2.4.4. Управление прерываниями

В процессе проектирования неизбежно возникновение ситуаций, в которых человеку приходится прерывать выполнение текущей задачи и переключаться на выполнение другой, после чего возвращаться к прерванной задаче. Такие ситуации могут возникать в том случае, если перед человеком поставлена задача, имеющая более высокий приоритет, чем задача, выполняемая в текущий момент времени, или в случае псевдопараллельного выполнения нескольких задач. Кроме того, возникают прерывания, не связанные с проектным процессом.

Обработка прерываний проектировщика как интеллектуального процессора человека осуществляется во многом аналогично обработке прерываний центральным процессором (ЦП) ЭВМ.

В однозадачном режиме обработка прерывания ЦП осуществляется в три этапа:

1. Прекращение выполнения программы. Этот эта прекращает работу текущей программы так, чтобы потом можно было вернуться к ней и продолжить. При этом происходит сохранение блока регистров, содержащего значения регистров процессора, в стек, работающий в режиме ***LIFO***.
2. Переход к выполнению обработчика прерывания
3. Возврат к выполнению прерванной программы. Для этого из стека извлекается блок данных программы, восстанавливаются значения регистров, в том числе и регистра указателя команд, что приводит к продолжению выполнения прерванной программы.

На рисунке 2.21 изображена общая схема прерывания человеческой деятельности.

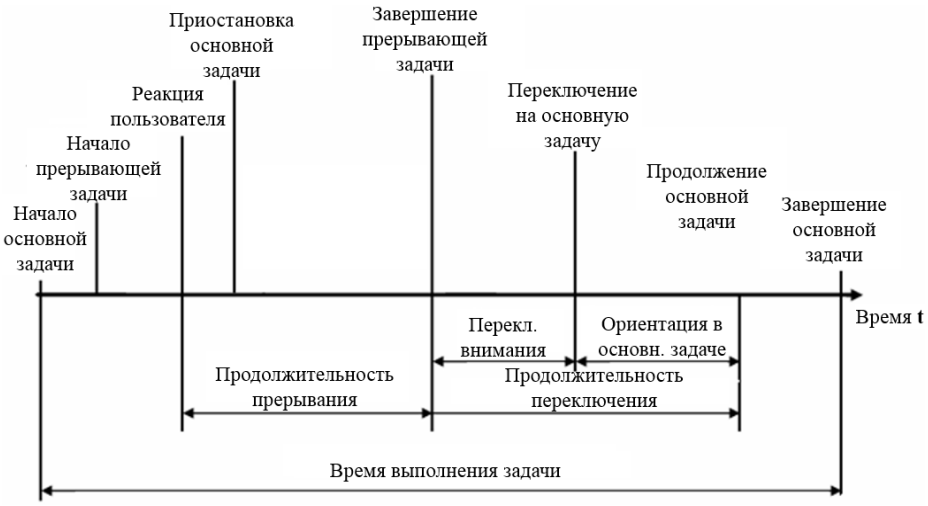


Рис. 2.21 Общая схема прерывания человеческой деятельности

Из схемы прерывания 2.21 видно, что с точки зрения человека процесс прерывания **Int** делится на две основные части:

Выполнение прерывания I_{ex}, которое заключается в реакции на возникшее прерывающее событие, в приостановке выполняемой основной задачи и в выполнении прерывающей задачи.

Переключение к основной задаче SW, состоящее из переключения фокуса внимания SW_f, на прерванную задачу и из ориентации в прерванной задачи Z_o.

$$Int = Iex,\ SW;$$

$$SW = SW_f,\ Z_o$$

Каждое прерывание сопровождается связанными с ним негативными эффектами[52][53], такими, как:

- Увеличение общего времени выполнения задачи за счет времени переключения внимания к прерванной задаче[70];
- Увеличение общего времени выполнения задачи и вероятности возникновения ошибок[70][62] за счет необходимости в ориентации в основной задаче.

Для минимизации этих негативных эффектов в состав средств ПКУ был включен инструментарий управления прерываниями. Данные средства позволяют минимизировать влияние обоих негативных эффектов.

Минимизация первого из них достигается за счет включения прерванной задачи в очередь задач проектировщика ***Q***. Причем, поскольку приоритет выполняющейся задачи заведомо выше, чем приоритеты остальных задач, находящихся в этой очереди, приоритет её устанавливается большим, чем у задачи, имеющей в ***Q*** наибольший приоритет. Таким образом, после завершения прерывающей задачи (а её выполнение может быть длительным), проектировщику не придётся ориентироваться в проектных задачах и вернуться к прерванной как к первоприоритетной. Поскольку прерывающая задача может также прерываться, возникает каскад прерываний. При этом очередь прерванных задач начинает работать в режиме, близком к **LIFO**, т.е. по аналогии со стеком.

Минимизация второго эффекта достигается путем сохранения текущего состояния прерванной задачи с целью помощи проектировщику в его восстановлении путем автоматизации процесса восстановления.

Обработка прерывания человеческой деятельности осуществляется по аналогии с прерыванием ЦП, при этом она содержит те же самые этапы обработки. К особенности первого этапа можно отнести то, что вместо блока данных программы необходимо сохранять текущее состояние прерываемой задачи.

Текущее состояние Z_{oc} состоит из следующих компонентов:

- Текущий решаемый вопрос задачи Qc;
- Состояние переменных задачи Z_v;
- Протокола работы над задачей Z_k;
- Представление решения задачи проектировщиком D_c.

Таким образом,

$$Z_{oc} = Q_c,\ Z_v,\ Z_k,\ D_c;$$

Состояние переменных для каждой задачи постоянно сохраняется в QA-памяти WIQA, следовательно, при прерывании выполнения задачи их значения не теряются.

Для обеспечения автоматизации процесса восстановления состояния задачи, поскольку в интеллектуальном процессоре человека отсутствует аналог регистра указателя команд, необходимо сохранять набор компонентов прерывания Z_{sc} для каждой прерванной задачи в виде списка прерываний I_L.

$$I_L = \{Z_{ref},\ Zsc\};$$

$$Z_{sc} = Q_c,\ Z_k,\ D_c;$$

Здесь Z_{ref} представляет собой ссылку на прерванную задачу в очереди. Q_c – ссылку на решаемый в ней вопрос. Z_k хранит в себе ссылку на протокол работы над задачей. Поскольку представление решения задачи проектировщиком невозможно сохранить напрямую, при возникновении прерывания проектировщику дается возможность дать его текстовое описание, которое и сохраняется как D_c.

Список прерываний отображается на вопросно-ответную память в виде таблицы T_{IL}, содержащей столбцы ссылки на задачу TZ_{ref}, ссылки на вопрос TQ_c, ссылки на протокол решения TZ_k и текстового описания собственного представления решения задачи TD_c.

$$T_{IL} = \{TZ_{ref},\ TQ_c,\ TZ_k,\ TD_c\};$$

В QA-памяти WIQA для хранения этой таблицы формируется QA-объект $TIntData^{QA}$, представляющий собой вопрос $Q_{IntData}$, "Данные прерываний", содержащий в себе подчиненный QA-объект таблицы списка прерываний:

$$TIntData^{QA} = QIntData,\ "\downarrow",\ (T_{IL}^{qa});$$

При этом T_{IL} в QA-память WIQA отображается как QA-объект T_{IL}^{qa}, состоящий из одного вопроса "Список прерываний" Q_{IL}, включающего в себя ряд вопросов, соответствующих атрибутам описания прерывания:

$$T_{IL}^{qa} = Q_{TZ},\ "\downarrow",\ Q_{AtpIL};$$

$$Q_{AtpIL} = QZ_{ref},\ QQ_c,\ QZ_k,\ QD_c;$$

Сами элементы списка прерываний при этом отображаются в виде ответов на вопросы атрибутов описания задачи. Таким образом описание задачи в QA-памяти WIQA раскрывается как IL_{Elem}^{QA}, представляющее собой набор ответов на эти вопросы.

$$IL_{Elem}^{QA} = (\{Q_{AtpIL}\},\ "\leftarrow",\ \{A_{AtpIL}\});$$

Этап восстановления прерванной задачи осуществляется следующим образом:

При получении из очереди следующей по приоритету задачи происходит проверка наличия её в списке прерванных задач. Если она там отсутствует, то решение задачи начинается сначала. В противном случае осуществляется переход к вопросу задачи, на котором возникло прерывание, а также отображение пользователю протокола его работы по ссылке Z_k и данное им описание D_c, что дает ему возможность быстрее сориентироваться в задаче. После этого пара $<Z_{ref}, Zsc>$ удаляется из списка прерываний I_L.

2.5. Программирование потоков работ

2.5.1. Расширение языка L^{WIQA} для программирования потоков работ

Язык L^{WIQA} в составе комплекса средств ***WIQA*** изначально определялся как открытый, к базовому ядру которого, специфицирующему традиционные данные и операторы, добавлялись и добавляются расширения. С целью программирования потоков работ язык L^{WIQA} расширен набором операторов, применяемых для программирования потоков работ.

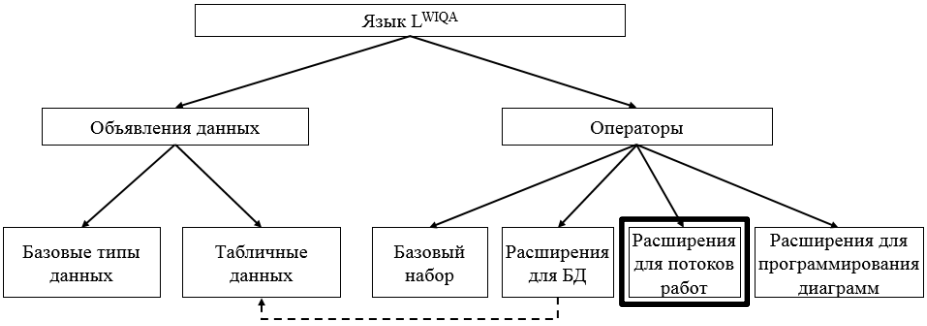


Рис. 2.22. Язык L^{WIQA} с расширениями и расширением для программирования потоков работ

Для моделирования потоков работ применяются приемы программирования на языках имитационного моделирования. В отличие от программирования на языках имитационного моделирования, в языке L^{WIQA} происходит создание статичных моделей потоков работ, которые впоследствии компилируются в соответствующие структуры данных. Но для описания их также можно применять операторы имитационного моделирования. В качестве образца был использован язык GPSS.

Таким образом, для осуществления псевдо-кодового программирования моделей потоков работ язык был дополнен следующими операторами:

- SEIZE – назначение задачи сотруднику
- QUEUE – добавление задачи в очередь на выполнение
- CANCEL – отмена выполнения задачи
- INTERRUPT – прерывание выполнения задачи.

Оператор **SEIZE** осуществляет ассоциирование между группой исполнителей и задачей в оргструктурах.

В расширенной БНФ-нотации данный оператор представляется следующим образом:

```
SEIZE = "SEIZE", "(", Z, D, ")";
```

Оператор **QUEUE** предназначен для помещения задачи в очередь на выполнение. Задачи в очереди выполняются при наступлении условия ***E***, указанного в этом операторе.

Синтаксис в расширенной БНФ-нотации:

```
QUEUE = "QUEUE", "(", Z, E, ")";
E = "\"", текст_условия, "\"";
```

Оператор **CANCEL** отменяет выполнение задачи, удаляя ее из очереди задач. Пользователь имеет возможность выполнить этот оператор вручную, а также – расширенный компилятор для обработки потоков работ выполняет эту функцию.

Синтаксис в расширенной БНФ-нотации:

```
CANCEL = "CANCEL", "(", Z, ")";
```

Оператор **INTERRUPT** прерывает выполнение текущей задачи, после чего она оказывается в очереди задач данной группы исполнителей с меньшим приоритетом, вычисляемым автоматически посредством системы управления прерываниями.

Синтаксис в расширенной БНФ-нотации:

```
INTERRUPT = "INTERRUPT", "(", Z, ")";
```

2.5.2. Особенности программирования потоков работ

В управлении бизнес-процессами накоплен богатейший опыт, очень важная часть которого аккумулирована в паттернах потоков работ[19]. Роль паттернов настолько значительна, что существует международная ассоциация учёных и практиков. активность которых регистрируется на специальном Интернет-сайте (http://www.workflowpatterns.com). На текущий момент на этом сайте зарегистрировано 43 паттерна, содержание которых было использовано авторами для выявления совокупности псевдо-кодовых операторов, включение которых в язык L^{WIQA}, обслуживающий ***QA***-программирование задач типов Z^{D} и Z^{N}, позволит использовать этот язык и в ***QA***-программировании задач типов Z^{W} и Z^{G}.

При поиске и спецификациях операторов, расширяющих язык L^{WIQA} до версии, обеспечивающей программное управление потоками работ, проводились параллели между расширением и языками имитационного

моделирования. Более того, для родственных операторов были сохранены их имена, устойчиво используемые в имитационном моделировании.

Рассмотрим паттерн «parallel split», одна из структур которого приведена на риунке 2.23.

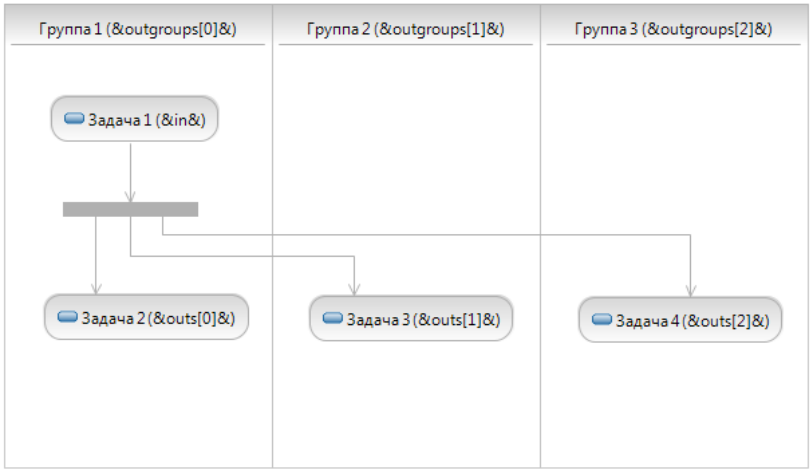


Рис. 2.23. Паттерн «Параллельное расщепление»

В этой версии паттерна происходит разделение потока на три ветки. Участвуют три группы разработчиков.

Для унификации используется оператор **SET**.

Рассматриваемый паттерн программируется на языке LWIQA с расширенным для программирования потоков работ набором операторов следующим образом:

```
    //Инициализация
    SET &in&, 1; &outs[0]&, 2; &outs[1]&, 3; &outs[2]&, 4;
&outgroup[0]&, 1; &outgroup[1]&, 2; &outgroup[2]&, 3; &cnt&, 0
    LABEL &L1&
    SEIZE &outs[&cnt&]&, &outgroup[&cnt&]&
    INC &cnt&
    TEST L, CNT, outs.length &L1&
    SET &cnt&, 0;
    //Выполнение задач
    LABEL &L2&
    QUEUE &outs[&cnt&]&, &in&.state == done
    INC &cnt&
    TEST L, &cnt&, outs.length &L2&
    FINISH
```

Библиотека паттернов разработана для облегчения оперативного программирования потоков работ, необходимость в которых выявлена в

процессе проектирования. Одним из важнейших источников таких конструктов является пошаговая детализация[111], по ходу которой, например, задача Z_J определённого уровня (пусть уровня J), разбивается на связную совокупность задач $C(\{Z_{J+1}\})$ уровня J+1. Разумеется, до QA-программирования такого потока динамические отношения между подчинёнными задачами должны быть представлены схемой потока W^k, например в следующем[111] виде:

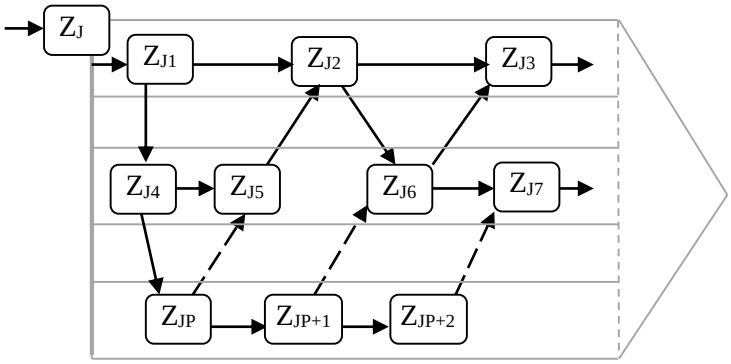


Рис. 2.24. Эскиз потока работ

Как уже отмечалось выше, ***QA***-программирование управляющей программы начинается с подготовки потока $\boldsymbol{W^k}$ к исполнению, включающей назначения задачам исполнителей и установлению плановых характеристик для их решения.

Предположим, что эта работа и все необходимые остальные работы учтены, и ***QA***-программа для потока $\boldsymbol{W^k}$ построена. Разумеется, построенную программу необходимо проверить. Одной из версий проверки является представление ***QA***-программы в базисе паттернов проектирования, псевдокоды которых заимствуются из библиотеки паттернов и встраиваются в проверяемую программу. Именно такая нагрузка и возлагается в совокупности средств программного управления на библиотеку паттернов потоков работ. Сам факт разложения ***QA***-программы потока в базисе паттернов является важным аргументом в пользу корректности построенной QA-программы.

Предположим, что ***QA***-программа для потока (пусть W^k) построена и проверена. причём не только через её разложение в базисе паттернов. После этого её следует исполнить. В авторской версии программного управления потоками исполнение каждой ***QA***-программы потока осуществляется компилятором, который по информации о задачах, вложенной в псевдокод потока, дополняет соответствующие очереди задач новыми составляющими. Другими словами, программное управление потоками работ нацелено на формирование очередей задач для каждого члена коллектива разработчиков.

Более того, очереди задач различаются и строятся в виде специализированных программ двух типов (M^1-программ и M^2-программ), состоящих из операторов, каждый из которых может исполняться только в случае истинности зафиксированного в операторе условия. То есть каждый элемент в любой очереди независимо от её типа оформляется и интерпретируется как условный оператор

If <условие доступа к элемену очереди> Then <Доступ к задаче по её идентификатору>,

исполняемый проектировщиком в роли I-процессора (рисунок 2.25).

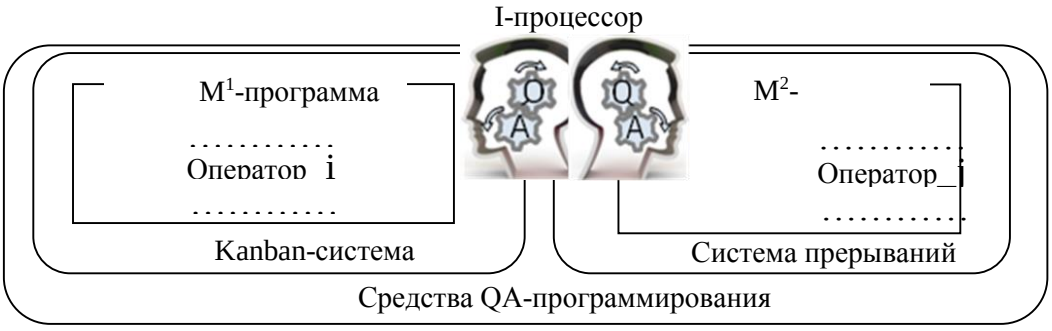


Рис.2.25. Взаимодействие I-процессора с очередями задач

Различия между очередями M^1- и M^2-типов состоит в том, что:

- в M^1-программах условия доступа регистрируют логику отношений между задачами потока во времени, управляя тем самым параллельно-согласованным решением задач группой проектировщиков;

- в то время как условия операторов M^2-программ управляют псевдопараллельным решением задач конкретным проектировщиком, каждая из которых может быть прервана по ходу решения, если в этом возникла необходимость.

Использование очередей двух типов позволяет отделить визуализацию общей картины исполнения задач группой проектировщиков от интерактивных взаимодействий проектировщика с задачами, которые он может выполнять псевдопараллельно.

Псевдопараллелное решение задач обслуживается средствами системы прерываний I-процессора с учетом информации по контролю за поручениями, которая используется для оперативных вычислений динамических приоритетов. Коррекция приоритетов осуществляется автоматически при очередном обращении проектировщика к очереди. Приоритеты помогают в выборе, но не диктуют его проектировщику, который свободен в действиях такого типа.

Выводы по второй главе

1. В качестве хранилища данных для инструментария программно-картотечного управления потоками работ выбрана ***QA***-память ***WIQA***
2. Табличное представление данных средств ПКУ можно отобразить на ***QA***-память ***WIQA*** посредством использования механизмов псевдокодовых баз данных;
3. Инструментарий ***WIQA*** содержит компоненты оргструктуры, интегрируемые со средствами ПКУ посредством отображения данных этих компонентов на ***QA***-память ***WIQA***;
4. В состав ***WIQA*** включен язык псевдокодового программирования L^{WIQA}, являющийся расширяемым. С целью программирования потоков работ, язык L^{WIQA} расширен набором операторов для программирования потоков работ;

5. Очереди задач как фрагменты потока работ являются программируемыми с использованием языка $\boldsymbol{L^{WIQA}}$;
6. Существует два типа программ, управляющих очередью задач. Первый отвечает за расположение задач относительно друг друга и параллелизм в их решении внутри потока работ проекта или плана этапа проектной работы, а второй отвечает за управление псевдопараллельным выполнением задач определенным проектировщиком;
7. Количество задач, отбираемых в процессе оперативного планирования, является ограниченным, и, в зависимости от применяемой технологии проектирования, это ограничение может представлять собой как численный лимит количества задач, так и максимальный объем работ, который команда способна выполнить за единицу времени;
8. Существуют технологии, позволяющие проектировщикам в процессе оперативного планирования произвести оценку трудоемкости той или иной задачи. Одной из таких технологий, которая применяется в Scrum-процессе, является покер планирования;
9. Параллельное и псевдопараллельное выполнение задач проектировщиками имитирует систему массового обслуживания, ряд характеристик которой можно применить в качестве метрических характеристик потока работ;
10. Данные метрические характеристики потока работ выступают в качестве обратной связи в процессе управления проектированием, позволяющей рационализировать оперативное планирование этапа проектной работы.

Глава третья. Методологическое обеспечение управления потоками работ.

3.1. Сценарная структуризация ПКУ

Программно-картотечное управление осуществляется путем решения набора задач управления, осуществляемого на протяжении всего проектного процесса. Проведем систематизацию этих задач, рассматривая их с детализацией, достижимой путем построения общей диаграмм вариантов использования. Для этого сначала произведем построение общей диаграммы вариантов использования всей системы ПКУ, а затем более детально рассмотрим варианты использования её составных частей.

Процесс управления потоком проектных работ можно представить как задачу управления $\boldsymbol{Zc}$. Решение этой задачи начинается с **формирования потока проектных работ**. Обозначим эту задачу как $\boldsymbol{Z_F}$. Поток проектных работ формируется на основе проектных задач, выполнение которых ведет к выполнению проекта. Введем обозначение $\boldsymbol{Z_{FZ}}$ для задачи определения набора проектных задач. В этом процессе принимает участие как непосредственный **руководитель проекта** $\boldsymbol{D_B}$, так и **другие стейкхолдеры** проекта $\boldsymbol{D_S}$, в частности, заказчики. Далее в процессе работы над потоком проектных работ перед руководителем проекта встает задача **формирования групп проектировщиков** $\boldsymbol{Z_{FG}}$ из коллектива проектной организации и **назначение проектных задач** $\boldsymbol{Z_{FZa}}$ группам с учетом их взаимосвязанности путем назначения **поручений** $\boldsymbol{Z_{FBa}}$. Этот процесс не является единовременным для проекта, так как в процессе проектной работы могут возникать новые проектные задачи.

После формирования потока проектных работ организация переходит к его выполнению. При этом в соответствии с используемой методологией проектирования (такой, как **Kanban** или **Scrum**), происходит формирование **бэклога этапа проектной работы** $\boldsymbol{Z_{FB}}$. В его формировании принимает

участие как руководитель проекта, ответственный за выбор задач для этапа проектной работы, так и **проектировщики**. Это формирование выполняется с учетом **метрик**, рассчитанных при выполнении предыдущих этапов проектной работы. В вычислении метрик могут участвовать другие стейкхолдеры проекта, к примеру, заказчик может оценить финансовый эффект от уже выполненных проектных задач. Далее в рамках используемой методологии проектирования участники выполняют проектную работу. После завершения этапа проектной работы и в его процессе происходит вычисление метрик, характеризующих особенности выполнения данного этапа. В процессе выполнения проектных задач проектировщики постоянно отчитываются о проделанной ими работе перед руководителем проекта.

Таким образом, можно выделить трех основных акторов, участвующих в проектном процессе:

- Руководитель проекта $\boldsymbol{D_B}$;
- Проектировщик $\boldsymbol{D_D}$;
- Другие стейкхолдеры проекта $\boldsymbol{D_S}$.

Эти акторы решают следующие основные задачи ПКУ:

- Формирование потока работ проекта $\boldsymbol{Z_F}$;
- Выполнение проектной работы $\boldsymbol{Z_W}$.

Задача формирования потока работ проекта, в свою очередь, включает в себя следующие подзадачи:

- Формирование набора задач проекта $\boldsymbol{Z_{FZ}}$;
- Формирование групп проектировщиков, ответственных за решение этих задач $\boldsymbol{Z_{FG}}$;
- Назначение задач проектировщикам и их группам $\boldsymbol{Z_{FZa}}$;

Задача выполнения проектной работы $\boldsymbol{Z_W}$ представляет собой итеративное выполнение следующих подзадач:

- Формирование бэклога этапа проектной работы $\boldsymbol{Z_{WB}}$;

- Выполнение этапа проектной работы $\boldsymbol{Z_{WE}}$;
- Формирование отчетов о выполненной работе $\boldsymbol{Z_{WR}}$.

Задача формирования бэклога этапа проектной работы включает в себя задачу назначения поручений.

В свою очередь выполнение этапа проектной работы включает в себя:

- Выполнение проектных задач $\boldsymbol{Z_{WEt}}$;
- Вычисление метрик $\boldsymbol{Z_{WEm}}$;
- Формирование отчетов о проделанной работе $\boldsymbol{Z_{WRp}}$.

Удобнее всего представить все эти задачи в виде ***UML***-диаграммы вариантов использования, представленной на рисунке 3.1.

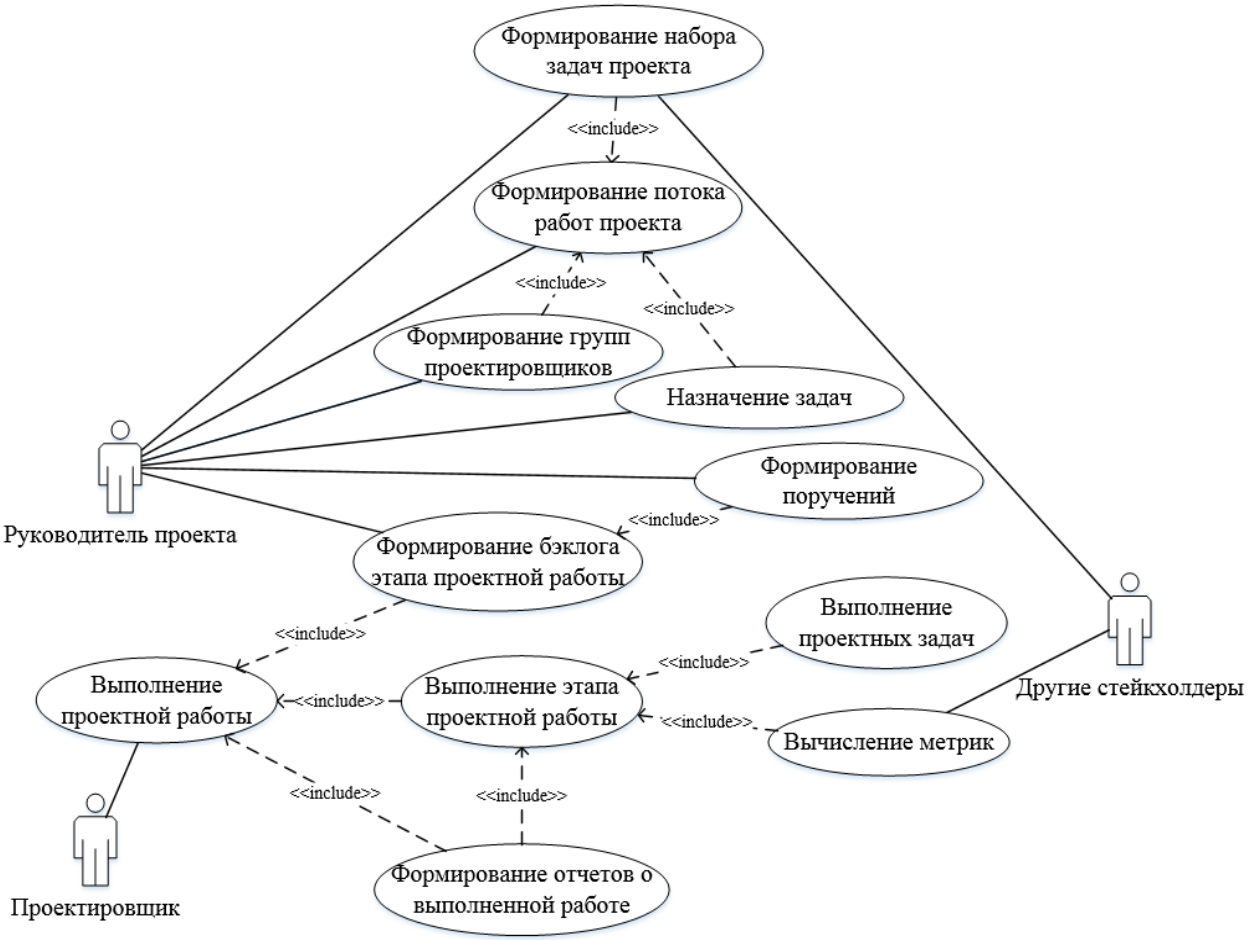


Рис. 3.1. Диаграмма вариантов использования системы ПКУ

Каждая из этих задач выполняется с помощью определенного средства, описание которых представлено в гл. 2. Перейдем к более детальному

рассмотрению задач, выполняемых с помощью данных средств.

Задача формирования набора задач проекта является полностью проектно-зависимой, её автоматизация сопряжена со значительными трудностями и её решение находится за рамками данного диссертационного исследования. Поэтому в комплексе средств ПКУ инструмент формирования набора проектных задач по техническому заданию отсутствует.

Теперь перейдем к задачам управления организационной структурой и назначения задач проектировщикам. Управление группами проектировщиков $\boldsymbol{Z_{FG}}$ включает в себя следующие задачи, выполняемые руководителем проекта:

- Создание группы проектировщиков $\boldsymbol{Z_{FGC}}$;
- Удаление группы проектировщиков $\boldsymbol{Z_{FGD}}$;
- Добавление проектировщика в группу $\boldsymbol{Z_{FGA}}$;
- Перемещение проектировщиков между группами $\boldsymbol{Z_{FGM}}$;
- Удаление проектировщика из группы $\boldsymbol{Z_{FGDp}}$;
- Назначение задач $\boldsymbol{Z_{FGS}}$.

При добавлении проектировщика в группу руководителю проекта необходимо определить для него роли. При этом он решает задачу определения ролей проектировщика. Диаграмма вариантов использования организационной структуры и подсистемы назначения задач в ПКУ представлены на рисунке 3.2.

Рис. 3.2. Диаграмма вариантов использования для формирования групп и назначения задач

Рассмотрим задачу формирования бэклога для выполнения этапа проектной работы. Руководителю проектного процесса здесь необходимо определить сначала набор задач для выполнения их в рамках данного этапа. Этот этап также в рамках данного диссертационного исследования не является программируемым. При этом группа проектировщиков выполняет оценку объема затрат труда на решение проектных задач. Одним из вариантов такой оценки является покер планирования. Использование покера планирования сводится к задачам оценки трудоемкости задачи и обсуждения этой оценки. Таким образом, набор задач формирования бэклога содержит основную задачу:

- Определение набора задач для выполнения на данном этапе проектирования и их приоритетов $\boldsymbol{Z_{WB}}$;

Определение набора задач для выполнения на данном этапе проектирования содержит в себе следующие подзадачи:

- Определение задач $\boldsymbol{Z_{max}}$, которые необходимо выполнить на данном

этапе ZZ_{max};

- Определение задач Z_{necc}, от выполнения которых зависит выполнение задач Z_{max}, но которые ещё не выполнены ZZ_{necc},;
- Определение задач Z_{time}, подлежащих выполнению, соответствующих определенному критерию ZZ_{time}.
- Определение задач Z_{Tnecc}, от выполнения которых зависит выполнение задач Z_{time}, но которые ещё не выполнены ZZ_{Tnecc};
- Определение приоритетов задач Z_{WBP};
- Определение трудоемкости каждой задачи и сопоставление общего объема работы с возможностями проектной группы Z_{WBE};
- Назначение поручений для выполнения задач проектировщиками Z_{WBP}.

При использовании покера планирования определение трудоемкости каждой задачи включает в себя следующие задачи:

- Определение каждым сотрудником трудоемкости задачи Z_{WBEZ};
- Обсуждение выставленных оценок трудоемкости Z_{WBED};
- Вычисление общей трудоемкости набора задач и отбор задач для бэклога с учетом возможностей группы проектировщиков Z_{WBEW}.

Представим набор задач формирования бэклога этапа проектной работы в виде диаграммы вариантов использования:

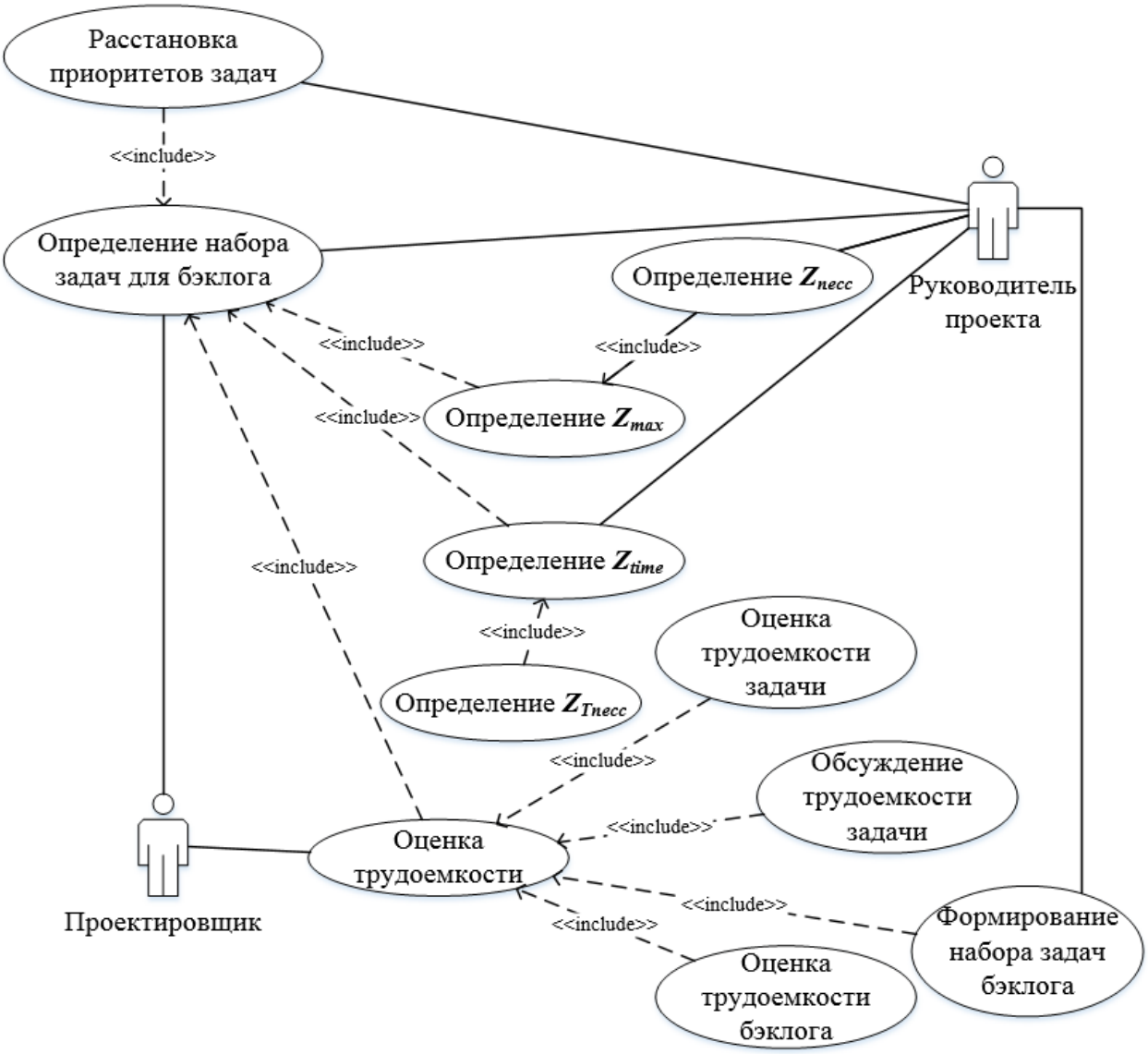


Рис. 3.3. Диаграмма вариантов использования для формирования задач бэклога проектного этапа

Далее рассмотрим задачи работы с поручениями. Руководитель проекта в данном случае является инициатором поручений, который имеет возможность создавать, редактировать, удалять, просматривать поручения, проверять выполнение, а также – перемещать выполненные поручения в архив. Проектировщик может использовать поручения для самоконтроля, при этом он должен иметь возможность просмотра поручений.

Таким образом, работа с поручениями включает в себя следующие задачи:

- Создание поручения Z_{WBPC};

- Редактирование поручения Z_{WBPE};
- Удаление поручения Z_{WBPD};
- Просмотр поручения Z_{WBPV};
- Выполнение поручения Z_{WBPX};
- Проверка выполненного поручения Z_{WBPK}.

По результатам проверки поручения руководитель может выполнить с ним следующие действия:

- Отправить поручение на доработку Z_{WBPR};
- Отправить поручение в архив Z_{WBPA}.

В виде диаграммы вариантов использования работа с поручениями представлена на рисунке 3.4.

Рис. 3.4. Диаграмма вариантов использования работы с поручениями

Выполнение проектных задач Z_{WEZ} зависит от выбранной методологии проектирования и в рамках средств ПКУ может происходить, используя средства Scrum или Kanban, а также оба этих инструмента одновременно. Также в процессе параллельного выполнения задач проектировщиком неизбежны прерывания, и проектировщику необходимо решать связанные с прерываниями задачи.

Таким образом, выполнение проектных задач включает в себя задачи Kanban Z_{WEK} и задачи Scrum Z_{WES}, а также задачи прерываний Z_{WEI}.

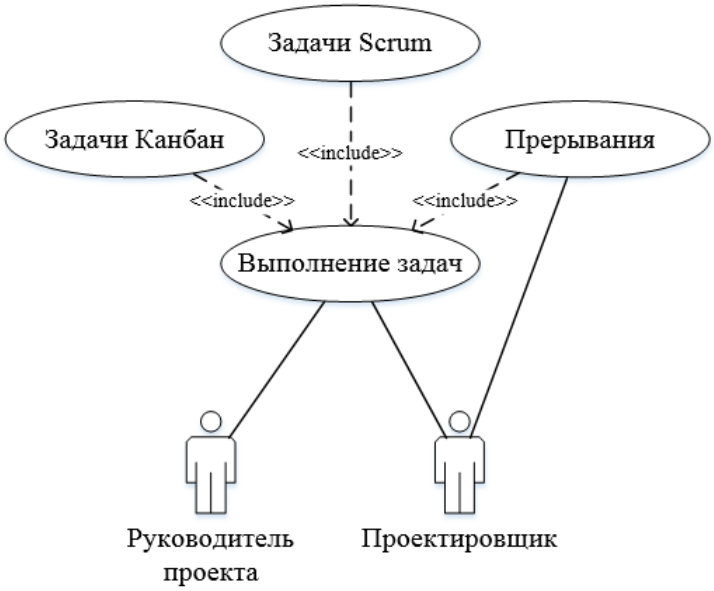


Рис. 3.5. Диаграмма вариантов использования выполнения задач

В процессе работы с использованием методологии Kanban проектировщик контролирует набор задач, которые назначены ему, просматривая карточки. С карточки он может перейти непосредственно к выполнению задачи, может отметить нужные ему карточки, перевернуть карточку, просмотрев дополнительную информацию о задаче, а также – сообщить о выполнении задачи для отметки информации об этом на доске изменением цвета карточки. И руководитель, и проектировщик имеют возможность просматривать доску Kanban.

Таким образом, основной задачей Kanban является просмотр доски и находящихся на ней карточек.

Эта задача включает в себя ряд подзадач:

- Отметка карточек Z_{WEKM};
- Отметка выполнения задачи Z_{WEKX};
- Просмотр детальной информации о задаче Z_{WEKD};
- Переход к выполнению задачи Z_{WEKG}.

Просмотр детальной информации о задаче, в свою очередь, включает в

себя задачи:

- Просмотр детального описания задачи Z_{WEKV};
- Просмотр дополнительной информации о задаче Z_{WEKE}.

На рисунке 3.6 представлена диаграмма вариантов использования Kanban в процессе выполнения проектных задач.

Рис. 3.6. Диаграмма вариантов использования Kanban в процессе выполнения проектных задач

Другой методологией проектирования является методология Scrum. В процессе выполнения работы в её рамках участники проектного процесса проводят ежедневные 15-минутные встречи Z_{WEM}, на которых обсуждается результат работы 24-часового суточного спринта, проектировщики отчитываются о проделанной работе и на диаграмме выгорания Scrum появляется новая точка, характеризующая произошедшее изменение трудоемкости остатка задач спринта. Таким образом, к задачам Scrum выполнения проектных задач относится следующая основная задача:

15-минутная встреча Z_{WEM}.

Эта задача включает в себя такие подзадачи, как:

- Обсуждение результатов работы за прошедший 24-часовой спринт

Z_{WEMR};

- Добавление точки на диаграмму выгорания Z_{WEMP};
- Обсуждение бэклога следующего 24-часового спринта Z_{WEMB};
- Расчет метрик, изменяющихся в течение выполнения спринта Z_{WEMM}.

Диаграмма вариантов использования задач Scrum в процессе выполнения задач бэклога показана на рисунке 3.7.

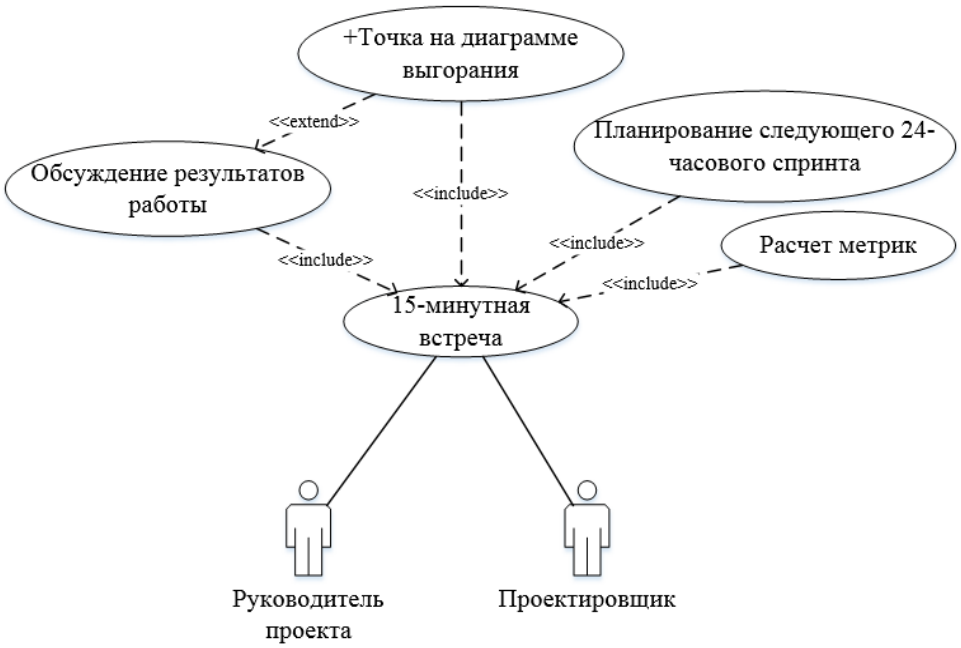


Рис. 3.7. Диаграмма вариантов использования Scrum в процессе выполнения проектных задач

Теперь рассмотрим задачи, возникающие в процессе обработки прерываний, возникающих в работе проектировщика. В случае возникновения прерывания проектировщику приходится **прерывать задачу**. Прерывание выполняемой задачи с использованием средств управления прерываниями ПКУ позволяет **сохранить ряд значений**, позволяющих упростить возврат к прерванной задаче. Кроме этого прерванная задача должна оказаться в очереди прерванных задач с **определенным приоритетом**. **Помещение её в очередь прерванных задач** является другой подзадачей прерывания. Ещё одной задачей обработки прерывания является **задача восстановления прерванной задачи**, включающая в себя **выбор задачи в очереди,**

извлечение прерванной задачи из очереди, переход к методике выполнения прерванной задачи и **отображение проектировщику последних его шагов** по выполнению прерванной задачи. Ещё одной задачей, связанной с прерываниями, является задача протоколирования действий проектировщика по решению им проектных задач.

Таким образом, задачи обработки прерываний следующие:

- Протоколирование Z_{WEIP};
- Прерывание задачи Z_{WEII};
- Восстановление прерванной задачи Z_{WEIR}.

Задача «Прерывание задачи» Z_{WEII} включает в себя следующие подзадачи:

- Сохранение сведений о прерванной задаче Z_{WEIS};
- Расчет приоритета прерванной задачи Z_{WEIE};
- Помещение прерванной задачи в очередь Z_{WEIQ}.

Задача «Восстановление прерванной задачи» состоит из следующих подзадач:

- Выбор задачи в очереди Z_{WEIC};
- Переход к методике выполнения прерванной задачи Z_{WEIG};
- Отображение проектировщику его шагов по выполне
- нию прерванной задачи Z_{WEIL}.

Задачи, связанные с обработкой прерываний являются индивидуальными задачами для участника проектного процесса, следовательно, их выполняет только один актор – сам проектировщик, в работе которого возникло прерывание. На рисунке 3.78 представлены задачи проектировщика, связанные с обработкой прерываний в виде диаграммы вариантов использования.

Рис. 3.8. Диаграмма вариантов использования Scrum в процессе выполнения проектных задач

Рассмотрим более детально задачу вычисления метрик. В процессе работы над проектом могут быть вычислены как специфические метрики Kanban и Scrum, так и любые другие метрики, вычисление которых запрограммировано с использованием псевдокода. К базовым метрикам Scrum, описанным в главе 2 относятся скорость работы сотрудника (IV), скорость работы команды (TV), оценка планирования (TD) и оценка стоимости (VD). К базовым метрикам Kanban относятся время цикла (Cycle Time), работы в процессе (WIP), время обработки (Lead Time), пропущенное время (Waste Time), эффективность (Effectiveness) и пропускная способность (Throughput). Все эти метрики не являются обязательными к вычислению, а также – они могут быть дополнены другими метриками.

Таким образом, к задаче вычисления метрик относятся задачи:

- Вычисление метрик Kanban $\boldsymbol{Z_{WRK}}$;
- Вычисление метрик Scrum $\boldsymbol{Z_{WRS}}$;
- Вычисление пользовательских метрик $\boldsymbol{Z_{WRU}}$.

Подзадачами вычисления метрик Kanban являются следующие задачи:

- Вычисление ***Cycle Time***;

- Вычисление ***WIP***;
- Вычисление ***Lead Time***;
- Вычисление ***Waste Time***;
- Вычисление ***Effectiveness***;
- Вычисление ***Throughput***;
- Вычисление других Kanban-метрик.

К подзадачам вычисления метрик Scrum относятся:

- Вычисление ***IV***;
- Вычисление ***TV***;
- Вычисление ***TD***;
- Вычисление ***VD***;
- Вычисление других Scrum-метрик.

На рисунке 3.9 представлена диаграмма вариантов использования для задачи вычисления метрик.

Рис. 3.9. Диаграмма вариантов использования для задач вычисления метрик

3.2. Систематизация задач программно-картотечного управления

Рассмотренные выше задачи можно свести в таблицу, характеризующую

их с позиции иерархии. Эта характеристика позволяет выделить группы задач, каждая из которых отвечает за выполнение определенного вида работы в рамках ПКУ. Таким образом, можно говорить о том, что процесс ПКУ состоит из набора работ, которые сводятся к решению задач управления. Эти работы в соответствии с группами задач объединяются в потоки работ. Далее представим схематично каждый из этих потоков работ и построим общую схему ПКУ как поток работ, включающий в себя потоки работ подзадач управления.

В процессе выполнения работ ПКУ происходит формирование ряда артефактов, которые требуются для выполнения других работ в рамках ПКУ. Далее приведем схематичное изображение взаимосвязи этих артефактов.

Таблица 3.1. Задачи ПКУ

Обозначение группы задач	**Группа задач**	**Обозначение задачи**	**Задача**	**Комментарий**
		$\boldsymbol{Zc}$	Задача управления потоком работ	Наиболее общая задача ПКУ
$\boldsymbol{Zc}$	Управление потоком работ	$\boldsymbol{Z_F}$	Формирование потока работ	
		$\boldsymbol{Z_W}$	Исполнение потока работ	
$\boldsymbol{Z_F}$	Формирование потока работ	$\boldsymbol{Z_{FZ}}$	Формирование набора задач проекта	
		$\boldsymbol{Z_{FG}}$	Управление группами проектировщиков	
		$\boldsymbol{Z_{FZa}}$	Формирование ассоциаций между задачами и исполнителями	
$\boldsymbol{Z_{FG}}$	Управление группами проектировщиков	$\boldsymbol{Z_{FGC}}$	Создание группы проектировщиков	
		$\boldsymbol{Z_{FGD}}$	Удаление группы проектировщиков	
		$\boldsymbol{Z_{FGA}}$	Добавление проектировщика в группу	
		$\boldsymbol{Z_{FGM}}$	Перемещение	

			проектировщиков между группами	
		Z_{FGDp}	Удаление проектировщика	
Z_{FG}, Z_{FZa}	Управление группами проектировщиков, формирование ассоциаций	Z_{FGS}	Назначение задач	
Z_W	Исполнение потока работ	Z_{WB}	Формирование бэклога этапа проектной работы	Данные задачи выполняются итеративно в соответствии с выполнением этапов проектной работы
		Z_{WE}	Выполнение этапа проектной работы	
		Z_{WR}	Формирование отчетов об этапе проектной работы	
Z_{WB}	Формирование бэклога этапа проектной работы	ZZ_{max}	Определение задач Z_{max}	
		ZZ_{necc}	Определение задач Z_{necc}	
		ZZ_{time}	Определение задач Z_{time}	
		ZZ_{Tnecc}	Определение задач Z_{Tnecc}	
		Z_{WBP}	Определение приоритетов задач	
		Z_{WBE}	Определение трудоемкости набора задач и их вместимости в этап проектной работы	
		Z_{WBAs}	Назначение поручений	
Z_{WBE}	Трудоемкость	Z_{WBEZ}	Определение трудоемкости задачи проектировщиком	
		Z_{WBED}	Обсуждение выставленных оценок трудоемкости	
		Z_{WBEW}	Вычисление общей трудоемкости набора задач, отбор задач для	

			этапа выполнения	
Z_{WBAs}	Управление поручениями	Z_{WBPC}	Создание поручения	
		Z_{WBPE}	Редактирование поручения	
		Z_{WBPD}	Удаление поручения	
		Z_{WBPV}	Просмотр поручения	
		Z_{WBPX}	Выполнение поручения	
		Z_{WBPK}	Проверка поручения	
		Z_{WBPR}	Отправить поручение на доработку	
		Z_{WBPA}	Отправить поручение в архив	
Z_{WE}	Выполнение этапа проектной работы	Z_{WEK}	Задачи Kanban	
		Z_{WES}	Задачи Scrum	
		Z_{WEI}	Задачи управления прерываниями	
Z_{WEK}	Задачи Kanban	Z_{WEKM}	Отметка карточек	
		Z_{WEKX}	Отметка выполнения задачи	
		Z_{WEKD}	Просмотр детальной информации о задаче	
		Z_{WEKG}	Переход к выполнению задачи	
Z_{WEKD}	Детальная информация о задаче	Z_{WEKV}	Просмотр детального описания задачи	
		Z_{WEKE}	Просмотр дополнительный информации о задаче	
Z_{WES}	Задачи Scrum	Z_{WEM}	15-минутная встреча, посвященная 24-часовому спринту	
Z_{WEM}	15-минутная встреча	Z_{WEMR}	Обсуждение прошедшего 24-часового спринта	
		Z_{WEMP}	Постановка точки на диаграмму	

			выгорания	
		Z_{WEMB}	Обсуждение бэклога следующего 24-часового спринта	
		Z_{WEMM}	Расчет метрик 24-часового спринта	
Z_{WEI}	Задачи управления прерываниями	Z_{WEIP}	Протоколирование	Персональные задачи
		Z_{WEII}	Прерывание	
		Z_{WEIR}	Восстановление после прерывания	
Z_{WEII}	Прерывание	Z_{WEIS}	Сохранение сведений о прерванной задаче	
		Z_{WEIE}	Вычисление приоритета прерванной задачи	
		Z_{WEIQ}	Помещение прерванной задачи в очередь задач	
Z_{WEIR}	Восстановление после прерывания	Z_{WEIC}	Выбор задачи в очереди	
		Z_{WEIG}	Переход к методике выполнения прерванной задачи	
		Z_{WEIL}	Просмотр последних шагов по решению задачи	
Z_{WR}, Z_{EMM}	Расчет метрик итоговых за этап и промежуточных	Z_{WRK}	Расчет метрик Kanban	
		Z_{WRS}	Расчет метрик Scrum	
		Z_{WRU}	Расчет пользовательских метрик	
Z_{WRK}	Расчет метрик Kanban	Z_{Cycle_Time}		
		Z_{WIP}		
		Z_{Lead_Time}		
		Z_{Waste_Time}		
		$Z_{Effectiveness}$		
		$Z_{Throughput}$		
Z_{WRS}	Расчет метрик Scrum	Z_{IV}		
		Z_{TV}		
		Z_{TD}		

Весь поток работ управления включает в себя задачу Zc, включающую в себя две подзадачи - Z_F и Z_W. Причем задачи эти являются связанными, т.к.

для выполнения задачи требуются результаты выполнения Z_F задачи Z_W. Схематично этот поток работ можно изобразить следующим образом:

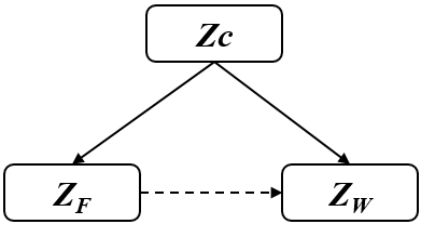


Рис. 3.10. Схема общего потока работ ПКУ

Изображенная на рисунке 3.10 пунктирная линия, связывающая между собой задачи Z_F и Z_W показывает зависимость выполнения задачи Z_W от результата выполнения задач Z_W – очевидно, что невозможно начать выполнение потока работ, пока он ещё не сформирован. Будем использовать такие обозначения и далее.

Каждую из этих задач можно рассматривать как отдельный поток работ – поток работ формирования потока проектных работ и поток работ выполнения проектных работ. Рассмотрим поток работ, соответствующий группе задач Z_F. Выполняя данную работу, невозможно осуществить назначение задач отдельным проектировщикам, не сформировав набора проектных задач. Поэтому задача Z_{FZ} должна выполняться до начала выполнения задачи Z_{FZa}. Также перед назначением задач проектировщикам должно быть произведено распределение проектировщиков по группам. Поэтому задача Z_{FG} должна быть выполнена до начала выполнения задачи Z_{FZa}. Представим схематично этот поток работ:

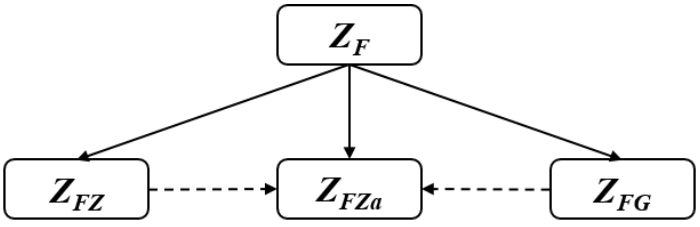


Рис. 3.11. Поток работ формирования потока проектных работ

При решении задачи формирования группы проектировщиков происходит выполнения потока работ управления проектными группами,

соответствующими группе задач $\boldsymbol{Z_{FG}}$. Эти задачи выполняются многократно и независимо друг от друга. На рисунке 3.12 представлен поток работ группы задач управления группами проектировщиков.

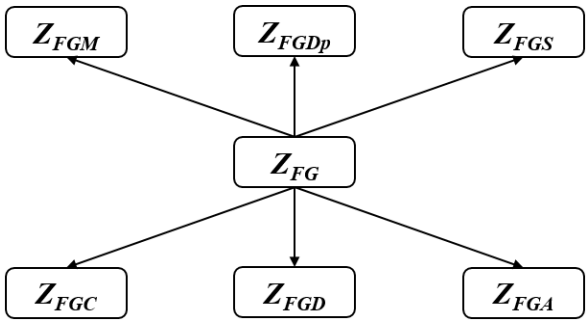


Рис. 3.12. Поток работ управления группами проектировщиков

Отображенная на рисунке 3.12 задача $\boldsymbol{Z_{FGS}}$ на самом деле принадлежит также и группе задач формирования ассоциаций $\boldsymbol{Z_{FZa}}$. Отнесем её к потоку работ управления группами.

Все эти потоки работ можно объединить в общий поток работ. Далее представлена схема потоков работ иерархии этого общего потока работ ПКУ:

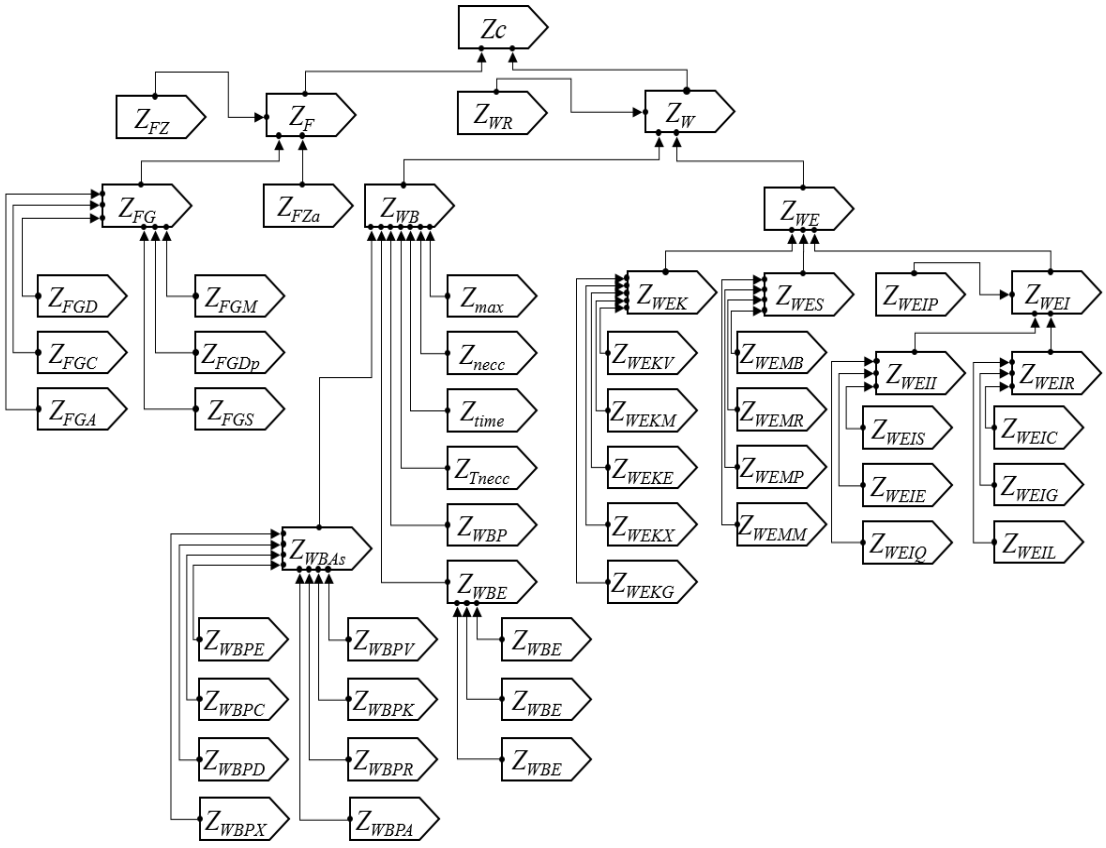


Рис. 3.13. Поток работ ПКУ

При выполнении потоков работ ПКУ происходит формирование ряда артефактов, схема которых представлена на рисунке 3.14.

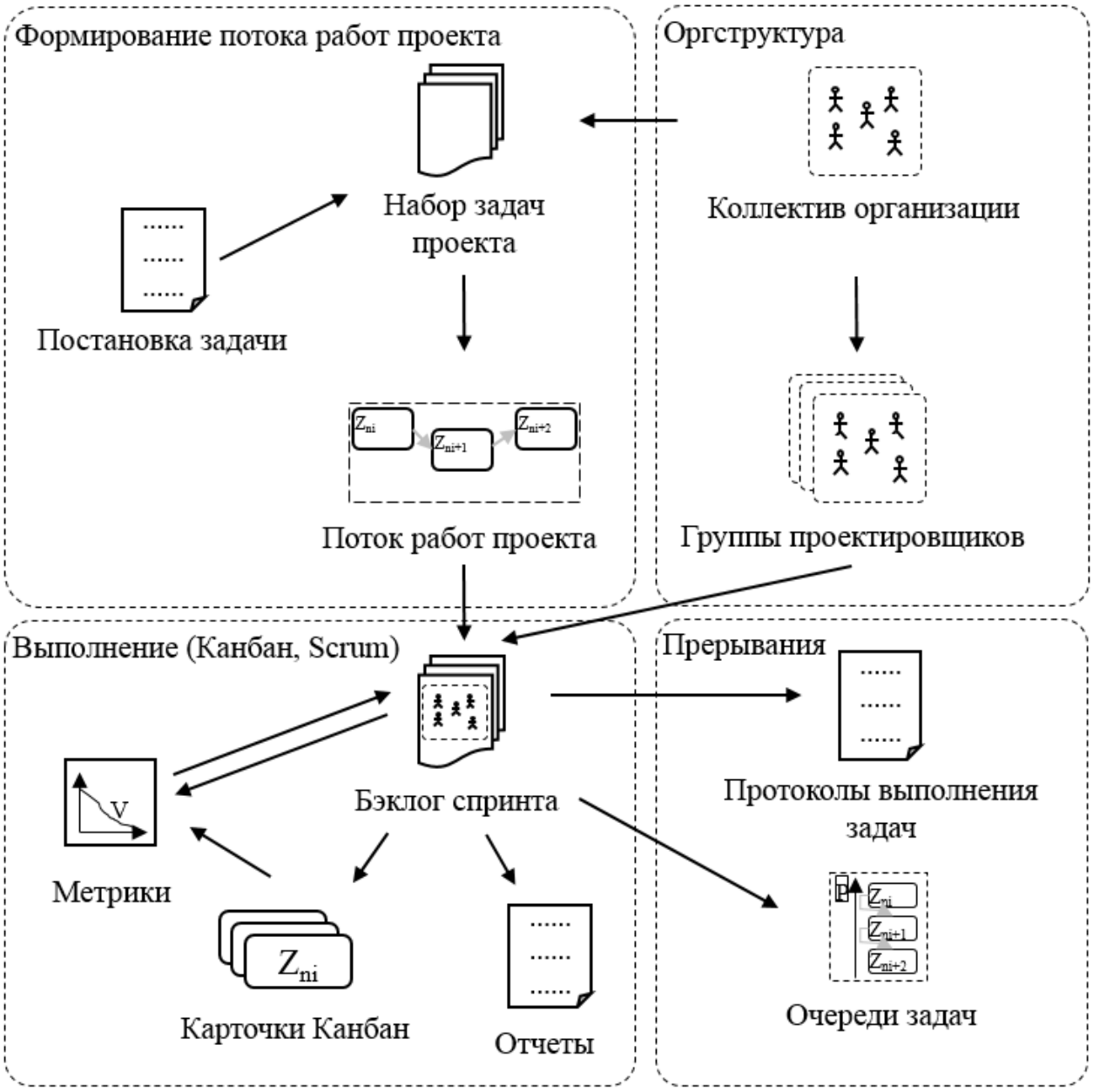


Рис. 3.14. Взаимосвязи артефактов в ПКУ

3.3. Методики программно-картотечного управления

3.3.1. Примеры методик ПКУ

Рассмотрим примеры методик выполнения работ ПКУ. В качестве оного из примеров рассмотрим методику прерывания выполнения задачи,

выполняемой в соответствии с методикой, содержащей элементы псевдокода. Данная методика представляется в виде диаграммы активности следующим образом:

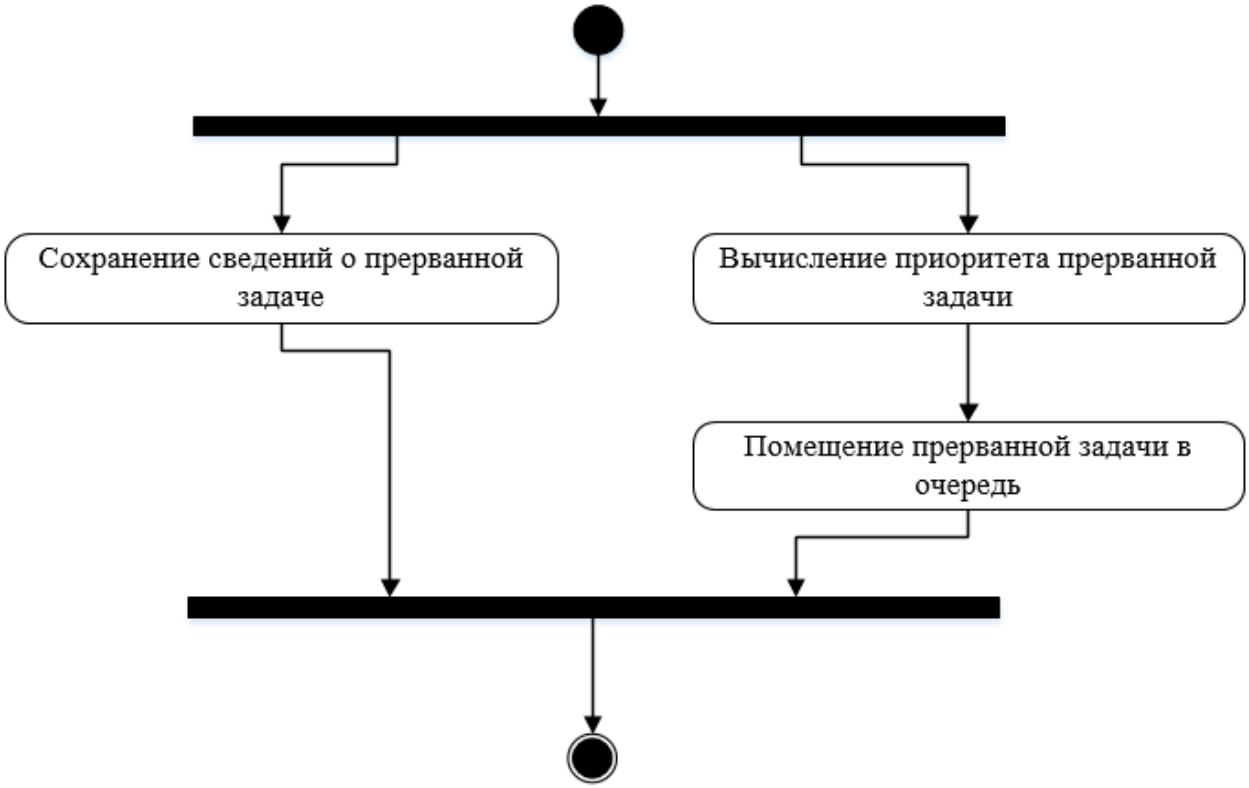


Рис. 3.15. Прерывание выполнения задачи

Из рисунка 3.15 видно, что выполнение данной методики предполагает параллелизм выполнения. Однако выполнение всех этих задач является программируемым, что не приводит к дополнительным временным затратам проектировщика. Это позволяет выполнить действия методики последовательно, без реализации параллелизма. Эту методику можно представить в виде последовательности шагов, каждый шаг которой может содержать вложенные шаги.

1. Сохранение сведений о прерванной задаче;
2. Вычисление приоритета прерванной задачи;

2.1. Поиск в очереди задач для выполнения той задачи, которая имеет наибольший приоритет Pmax;

2.1.1. ***&dbqa& := QA_GetQAId(&current_project&, "БД очереди");***

2.1.2. ***&dbid& := OpenDB(&current_project&, &dbqa&);***

2.1.3. ***&userstr& := INTTOSTR(&current_user&);***

2.1.4. ***&sql& := "SELECT Приоритет FROM Очередь WHERE Пользователь = ";***

2.1.5. ***&sql& := STRCAT (&sql&, &userstr&);***

2.1.6. ***&res& := ExecuteSQL (&sql&);***

2.1.7. ***&nP& := DB_GETROWCOUNT (&res&);***

2.1.8. ***&pcnt& = 0;***

2.1.9. ***&pmax& = DB_GETCELLINT(&res&, 0, 0);***

2.1.10. ***LABEL &pCycle&;***

2.1.11. ***IF &pcnt& >= &nP& THEN GOTO &pFinal&;***

2.1.12. ***IF DB_GETCELLINT(&res&, &pcnt&, 0) > &pmax& THEN &pmax& := DB_GETCELLINT(&res&, &pcnt&, 0);***

2.1.13. ***&pcnt& := &pcnt& + 1;***

2.1.14. ***GOTO &pCycle&;***

2.1.15. ***LABEL &pFinal&;***

2.2. Вычисление приоритета прерванной задачи ***Pnew*** как ***Pmax + Pn***, где Pn – произвольный коэффициент, зависящий от необходимости впоследствии добавлять задачи

2.2.1. ***&pNew& := &pMax& + Pn;***

3. Помещение прерванной задачи в очередь с приоритетом ***Pnew***;

4. Конец методики.

Шаг 2 данной методики выполняется как последовательность псевдокодовых операторов языка $\boldsymbol{L^{WIQA}}$. В приведенном выше примере эти операторы представлены как вложенные в шаги 2.1 и 2.2 действия.

В качестве другого примера методики можно взять методику создания поручения. Представим эту методику в виде последовательности шагов для их выполнения.

1. Просканировать отображения назначений задач на QA-дерево и обнаружить новые задачи;
2. Для обнаруженной задачи отобразить окно создания нового поручения;
3. Установить даты поручения;
4. Проверить по таблице Kanban, не возникает ли превышения числа выполняемых задач на данном этапе проектной работы;
5. Указать зависимости исполнения этого поручения от результатов выполнения других задач;
6. Установить исполнителя поручения;
7. Указать лиц, ответственных за контроль поручений;
8. Зарегистрировать поручения в структурах ПКУ;

8.1. Зарегистрировать поручение в QA-представлении таблицы поручений;

8.2. Сформировать карточку на доске Kanban;

8.2.1. Зарегистрировать информацию о карточке Kanban в QA-представлении доски Kanban;

9. Конец методики.

Данная методика задействует сразу несколько компонентов системы ПКУ и использует результаты работы этих компонентов. Представим набор этих компонентов в виде UML-диаграммы компонентов, представленной на рисунке 3.26.

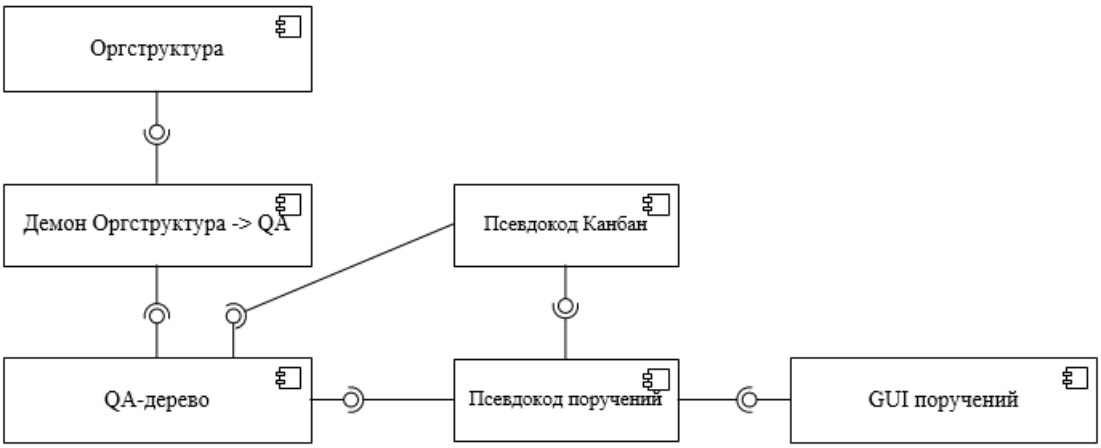


Рис. 3.16. Компоненты ПКУ, задействованные в создании нового поручения

В некоторых случаях выполнение потоков работ системы ПКУ можно представить в виде последовательности переходов из одного состояния в другое. Например, процесс выполнения потока работ проекта происходит поэтапно:

1. Сформировать бэклог этапа проектной работы;
2. Выполнить этап проектной работы;
3. Сформировать отчетов о выполненной работе.

Каждое из этих действий можно представить как состояние, в котором находится группа проектировщиков. На рисунке 3.17 представлена диаграмма состояний для процесса выполнения этапа проектной работы.

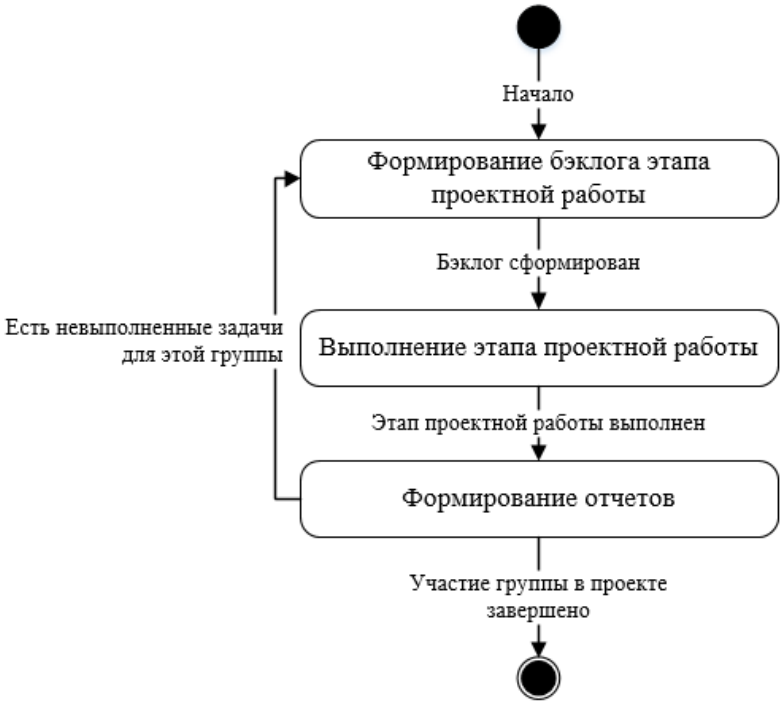


Рис. 3.17. Компоненты ПКУ, задействованные в создании нового поручения

3.3.2. Экспериментирование в ПКУ

Ранее неоднократно отмечалось, что предлагаемая версия гибкого управления в виде ***ПКУ*** относится к классу эмпирических. Её эмпиричность проявляется в следующих возможностях, предоставляемых проектировщикам:

Решая назначенную ему задачу в её псевдокодовом представлении, проектировщик может прервать исполняемую им работу до или после любого оператора и провести концептуальный эксперимент или концептуальный анализ и оценивание, если он видит в этом необходимость.

По ходу исполнения параллельных или псевдопараллельных работ имеется возможность для вычисления совокупности метрик, значения которых можно и следует использовать для воздействий на формирование групп проектировщиков, отбор задач в бэклоги и на выбор задач из их очередей.

Первый класс возможностей связан с ситуациями процесса проектирования, в которых проектировщик, ответственный за прерываемую

задачу (пусть Z_i), столкнулся с необходимостью или целесообразностью (прежде чем продолжить работу) что-то уточнить.

В уточнении, а значит и соответствующем ему снижении неопределённости, различаются два случая, в одном из которых проектировщик, активизировав прерывание, проводит подходящий концептуальный эксперимент, связывая с ним задачу Z_{ij}, подчинённую задаче Z_i. В этом случае задача Z_{ij} включается в общее дерево задач с автоматическим назначением для неё как ответственного за решение, так и плановых характеристик времени их решения (значения наследуются от родительской задачи).

Во втором случае проектировщик реагирует, то есть проводит потребовавшиеся ему концептуальные действия без оформления и регистрации их как задачи, но в условиях явного использования системы прерываний.

В любом из выделенных случаев либо по результату эксперимента, либо по результатам анализа и оценивания проектировщик **принимает определённое решение** о продолжении работы с Z_{ij}, А значит, первый класс возможностей связан с ситуациями, реагирование на которые помогает принять полезное управленческое решение, потребность в котором появились оперативно. Возможность управленческого реагирования на подобные ситуации следует связать с одним из специфических проявлений **гибкости *ПКУ***.

Отметим, что предоставляемый проектировщикам инструментарий реагирования на рассмотренный класс управленческих решений допускает либо возврат к работе с прерванной задачей Z_{ij}, либо переход к задаче по результатам взаимодействия с доступными очередями задач.

Второй класс возможностей связан с определением значений метрик параллельной и псевдопараллельной работы проектировщиков,

представленных выше, после чего по полученным значениям метрик также ***принимаются управленческие решения***. Эмпирика в этом классе возможностей проявляет себя в ***измерениях значений метрик***, причём, ничто не мешает считать данные, по которым вычислены значения метрик, данными, полученными в результате «***экспериментов***», за которыми стоят определённые объемы работ, выполненных в процессе проектирования. Так, например, осуществлённый ***Scrum***-спринт можно интерпретировать как «***эксперимент***».

В качестве примера эмпирической методики, принадлежащей ко второму классу возможностей далее приведена методика выполнения задачи Z_{WBE} - задачи определения трудоемкости набора задач бэклога этапа выполнения проектной работы и определения из них тех задач, которые проектировщики будут способны выполнить за время этапа проектной работы. Рассматривая все прошедшие Scrum-спринты как «эксперименты», можно вычислить ряд метрик, используемых при составлении новых спринтов. К таким метрикам относится средняя скорость работы команды. На начало выполнения этой задачи имеются следующие входные данные: средняя скорость работы команды и продолжительность в днях выполнения спринта. Также имеются множества задач Z_{max}, Z_{necc}, Z_{time}, Z_{Tnecc}. Также имеется. Каждой из этих задач назначен соответствующий ей приоритет. Этот приоритет характеризует два свойства задачи:

- Важность выполнения этой задачи на данном этапе проектной работы;
- Порядок выполнения задач.

В результате выполнения этой задачи в системе ПКУ появляется набор задач, подлежащих выполнению в течение заданного времени – времени этапа проектной работы. Если для решения задач используется методология Scrum, то этим этапом проектной работы является спринт.

На рисунке 3.18 представлена диаграмма последовательности для этой методики:

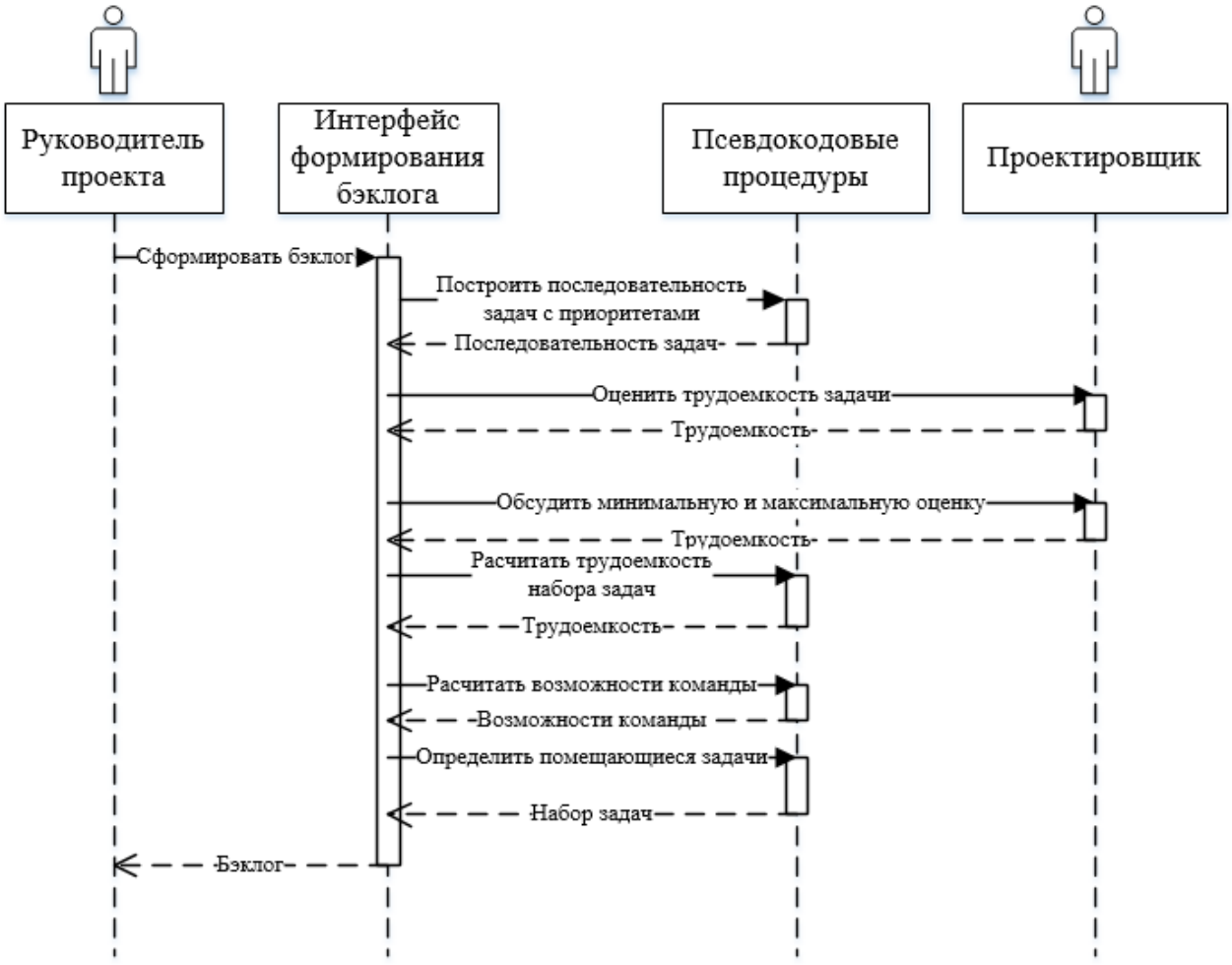


Рис. 3.18. Формирование набора задач бэклога

Представим методику в виде последовательности шагов. Каждый шаг может состоять из нескольких вложенных в него шагов.

1. Сформировать из всех пар ***<задача, приоритет>*** упорядоченную в соответствии с приоритетом последовательность задач Z_p;
2. Для каждой задачи Z_i последовательности, сформированной в п.1 данной методики. рассчитать её трудоемкость;

2.1. Каждому участнику выбрать сложность задачи из последовательности чисел Фибоначчи (1, 2, 3, 5, 8, 13, 21, 34, 65) трудоемкость задачи Z_i;

2.2. Участнику, выбравшему минимальную трудоемкость задачи,

обосновать свою позицию;

2.3. Участнику, выбравшему максимальную трудоемкость задачи, обосновать свою позицию;

2.4. Совместно выбрать определенную трудоемкость задачи;

3. Рассчитать вместимость бэклога спринта как $U = Vcp*N$, где Vcp – средняя скорость работы команды, а N – длина спринта в днях;

4. Определить набор задач, помещающихся в спринт;

4.1. Установить начальное значение общей трудоемкости набора задач X;

4.2. Выбрать задачу Z_i как самую первую задачу в последовательности, отсортированной в приоритетном порядке.

4.3. X увеличить на трудоемкость задачи Z_i

4.4. Вычислить трудоемкость набора задач $Z_1..Z_i$ с учетом факторов по формуле $Zv = X\ (1 + (Diff + DF + EF + FMC))$, описанной в гл. 2.

4.5. Если $Zv < U$, то добавить задачу Z_i в бэклог этапа проектной работы, иначе перейти к п. 5 данной методики;

4.6. Если в последовательности задач Z_p ещё имеются нерассмотренные задачи, то выбрать в качестве Z_i задачу, следующую после текущей задачи Z_i по приоритету;

4.7. Перейти к п. 4.3 данной методики;

5. Методика завершена.

3.3.3. Методики оценки эффективности проектной работы

Методики оценки эффективности проектной работы заключаются в вычислении её метрик. Вычисление метрик является программируемым с использованием языка L^{WIQA}. Большинство их вычисляются в полностью автоматическом режиме.

В качестве примера методики оценки эффективности проектной работы

рассмотрим методику вычисления Kanban-метрики ***CYCLE_TIME*** - среднего значения времени, которое задача проходит от момента её постановки до завершения работы над задачей.

Для вычисления значения метрики ***CYCLE_TIME*** – среднего значения времени, которое задача проходит от момента её постановки до завершения работы над задачей – необходимо выполнить следующие действия:

- отобрать из таблицы контроля поручений выполненные задачи;
- для каждой задачи вычислить в днях разницу между началом её постановкой и завершением работы над задачей и подсчитать среднее арифметическое этих разниц.

Выпонение такой работы осуществляется автоматически путем выполнения следующих шагов:

1. **&dbqa& := QA_GetQAId(&current_project&, "БД поручений");**
2. **&dbid& := OpenDB (&current_project&, &dbqa&);**
3. **&tasks& := ExecuteSQL(&dbid&, "SELECT Дата_исполнения, Дата_поручения FROM Задачи WHERE Статус = 1");**
4. **&ntasks& := DB_GETROWCOUNT(&tasks&);**
5. **&NCNT& := 0;**
6. **LABEL &CCT_L1&;**

6.1. **IF &NCNT& >= &ntasks& THEN GOTO &CCT_L2&;**

6.2. **&DSTART& := DB_GETCELLDATETIME(&tasks&, &NCNT&, 1);**

6.3. **&DEND& := DB_GETCELLDATETIME(&tasks&, &NCNT&, 0);**

6.5. **&NDAYS& := &NDAYS& + DT_GETDATEDIFFERENCE(&DEND&, &DSTART&);**

6.6. **&NCNT& := &NCNT& + 1;**

6.7. **GOTO &CCT_L1&;**

7. **&NDAYS& := 0;**

8. **LABEL &CCT_L2&;**

9. **&CYCLE_TIME& := &NDAYS& / &NCNT&.**

На первом шаге данная процедура с помощью встроенной функции ***QA_GetQAId*** определяет индекс единицы, содержащей базу данных поручений по её имени "***БД поручений***". Следующее действие с помощью функции работы с базами данных ***OpenDB*** открывает базу данных для работы с ней и в переменную ***&dbid&*** сохраняется идентификатор открытой базы данных. Затем выполняется SQL-запрос "***SELECT Дата_исполнения, Дата_поручения FROM Задачи WHERE Статус = 1***", осуществляющий выбор даты исполнения и даты назначения поручения для всех выполненных задач, находящихся в базе данных поручений. Результат запроса сохраняется в виде строки. Затем с помощью функции работы с базами данных ***DB_GETROWCOUNT*** для результата запроса ***SELECT*** происходит определение количества возвращенных строк. Это количество соответствует числу обнаруженных выполненных задач. Далее, с помощью меток, условий и операторов перехода реализован цикл, осуществляющий обход возвращеннвых в результате SQL-запроса значений дат. Для каждой задачи с помощью дополнительной функции работы с датами ***DT_GETDATEDIFFERENCE*** вычисляется число дней от момента постановки задачи до момента её выполнения. Сами эти даты извлекаются из результата запроса с помощью функции работы с базами данных ***DB_GETCELLDATETIME***. Эти значения в цикле суммируются в переменной ***&NDAYS&.*** После завершения цикла переменной ***&CYCLE_TIME&*** присваивается вычисленное значение метрики.

Выводы по третьей главе

1. В процессе программно-картотечного управления существуют как минимум три роли участников: руководитель проектной группы,

проектировщики, а также – другие участники проектного процесса;

2. Процесс программно-картотечного управления является представимым в виде набора сценариев, включающих в себя последовательности действий по планированию, исполнению и анализу выполненных проектных задач;
3. Выполнение сценариев программно-картотечного управления является решением управленческих задач, необходимым для контролируемого выполнения потока проектных работ.
4. Существуют две основные задачи ПКУ – задача формирования потока проектных работ, и задача его выполнения. Эти задачи включают в себя иерархическую структуру подзадач;
5. Данные управленческие задачи с подзадачами формируют поток работ управления, который интегрируется с потоком проектных задач и выполняется проектировщиками и их руководителями;
6. В процессе выполнения управленческих задач происходит формирование ряда артефактов, таких, как, например, очереди задач, рассчитанные метрические характеристики. Эти артефакты требуются другими задачами для начала их выполнения. Соответственно, задачи ПКУ также, как и задачи проекта, являются взаимосвязанными и требующими определенного порядка их выполнения;
7. Часть задач ПКУ являются полностью автоматизируемыми и решаются посредством выполнения псевдокодовых программ, в то время как другие задачи для их решения требуют непосредственного участия проектировщиков;
8. Предлагаемая версия ПКУ допускает возможность экспериментирования в проектном управлении. Она предоставляет возможность использования механизма прерываний для проведения концептуального эксперимента, в результате которого могут

возникать новые проектные задачи;

9. Выполненный Scrum-спринт также можно рассматривать в качестве эксперимента, в результате выполнения которого были получены метрические характеристики. Эти характеристики влияют на принятие управленческих решений в процессе формирования новых спринтов;
10. Вычисление всех рассмотренных в данной диссертационной работе как Kanban-, так и Scrum-метрик выполняется автоматически путем выполнения псевдокодовых программ их вычисления за исключением Scrum-метрики VD, в вычислении которой участвуют различные стейкхолдеры проекта, дающие оценку стоимости выполненной работы.

Глава четвертая. Особенности реализации средств программно-картотечного управления потоками проектных работ

4.1. Организация комплекса средств ПКУ в среде WIQA

Комплекс средств ПКУ реализован в двух версиях среды ***WIQA*** – многопользовательской сетевой ***WIQA.Net*** и однопользовательской ***WIQA***. В большинстве своем компоненты ПКУ являются универсальными и используются в обеих средах за небольшими отличиями.

Рассмотрим реализацию ПКУ в среде ***WIQA.Net***. В этом случае среда ***WI***QA размещается на трех типах устройств:

- Сервер баз данных;
- Сервер ***WIQA.Net***;
- Клиент ***WIQA.Net***.

Сервер баз данных содержит ***СУБД MS SQL***, в которую отображается память WIQA для её хранения. ***Сервер WIQA.Net*** взаимодействует как с базой данных, так и с клиентом, таким образом, он является интерфейсом между клиентом WIQA и сервером баз данных.

Устройство ***Клиент WIQA.Net*** отвечает непосредственно за выполнение алгоритмов компонентов системы ПКУ. Клиент включает в себя приложение ***WIQA клиент***, а также ряд дополнительных подключаемых библиотек, отвечающих за графический пользовательский интерфейс псевдокодовых компонентов ПКУ:

- Дополнительная подключаемая библиотека GUI поручений;
- Дополнительная подключаемая библиотека GUI очередей;
- Дополнительная подключаемая библиотека GUI Kanban;
- Дополнительная подключаемая библиотека GUI Scrum.

Приложение ***WIQA***-клиент содержит следующие компоненты, связанные

с функционированием ПКУ:

- Вопросно-ответный протокол;
- Демон отображения оргструктуры;
- Интерпретатор псевдокода.

Вопросно-ответный протокол применяется для создания, просмотра и редактирования данных, хранящихся в ***QA***-памяти ***WIQA***, в том числе – и задач проекта. Демон отображения оргструктуры осуществляет преобразование оргструктуры в ***QA***-формат и отображение её на ***QA***-память ***WIQA***. Интерпретатор псевдокода позволяет выполнить псевдокодовые алгоритмы ПКУ.

Теперь рассмотрим компоненты ПКУ, относящиеся к серверу ***WIQA***. Эти компоненты представлены тремя типами: ***QA***-данные, ***QA***-программы (компоненты ПКУ, запрограммированные на языке псевдокода), а также данные, не имеющие вопросно-ответной структуры. Из используемых в подсистеме ПКУ к таким данным относятся данные оргструктуры.

К серверным компонентам ПКУ относятся:

- ***QA-БД проекта*** – вопросно-ответное представление проекта, с которым работают средства ПКУ;
- ***Оргструктура*** – представление данных оргструктуры в памяти ***WIQA***;
- ***QA-отображение оргструктуры*** – отображение, полученное с помощью демона отображения оргструктуры;
- ***QA-БД поручений*** – QA-представление таблиц подсистемы управления поручениями;
- ***Псевдокоды поручений*** – программные коды на языке $\boldsymbol{L^{WIQA}}$, отвечающие за работу подсистемы поручений;
- ***QA-БД очередей*** – QA-представление таблиц подсистемы управления очередями и прерываниями;

- ***Псевдокоды очередей и прерываний*** – псевдокодовая программная составляющая подсистемы управления очередями и прерываниями в ПКУ;
- ***QA-БД Kanban*** – ***QA***-представление доски Kanban;
- ***Псевдокоды Kanban*** – алгоритмическое обеспечение Kanban;
- ***QA-БД Scrum*** – ***QA***-представление данных Scrum;
- ***Псевдокоды Kanban*** – алгоритмическое обеспечение Scrum;
- ***Псевдокоды метрик*** – алгоритмы вычисления метрик Kanban и Scrum.

Перечисленные выше компоненты ПКУ в среде ***WIQA.Net*** можно обобщить в виде диаграммы размещения, представленной на рисунке 4.1:

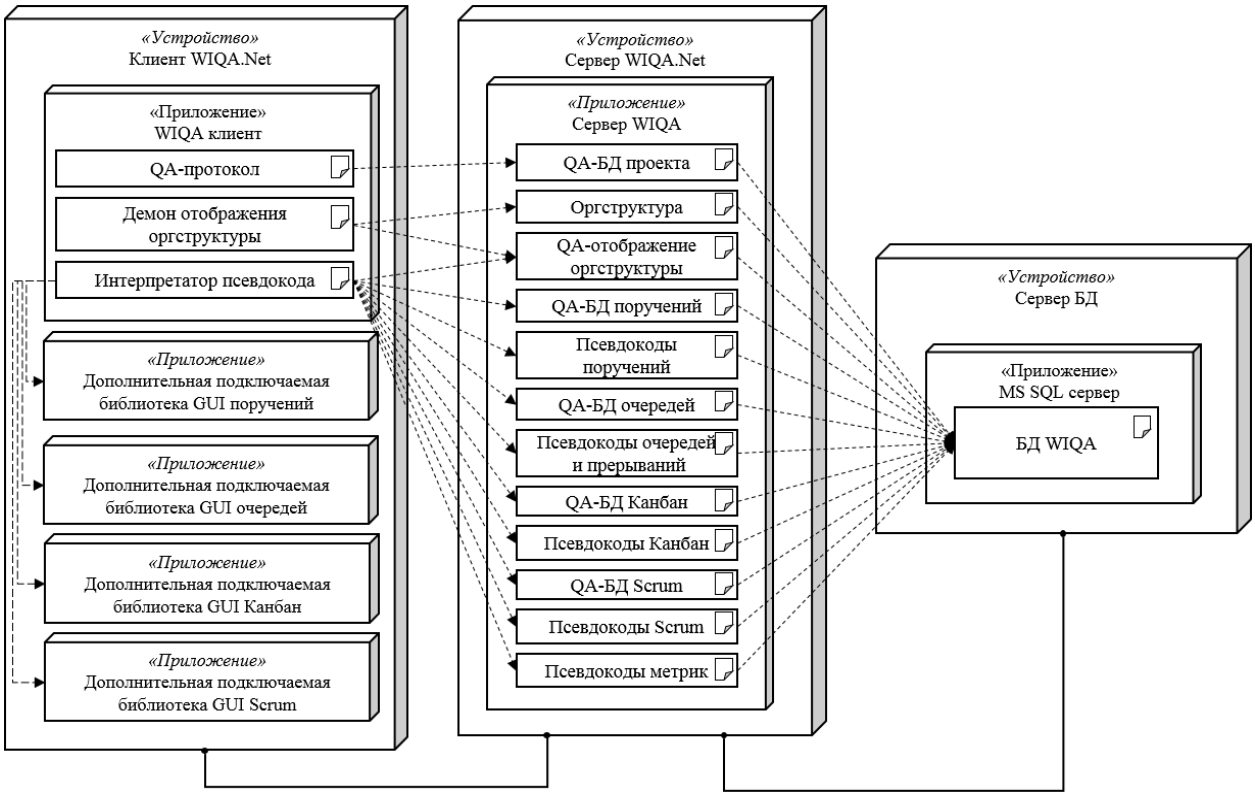


Рис. 4.1. Диаграмма размещения компонентов ПКУ в среде WIQA.Net

Все эти устройства WIQA.Net могут находиться как на различных физических машинах, объединенных с помощью сети, работающей на основе протоколов TCP/IP, так и могут быть расположены на одной машине. В системе ***WIQA.Net*** сервер баз данных и сервер ***WIQA.Net*** находятся в

единственном экземпляре, а клиентов может быть больше одного, при этом для рационального использования средств ПКУ количество клиентов ***WIQA.Net*** не должно быть меньше количества участников проектного процесса – каждый участник взаимодействует со всей системой ПКУ через клиента ***WIQA.Net***.

На рисунке 4.2 схематично представлено взаимодействие пользователей с системой ПКУ в многопользовательском режиме:

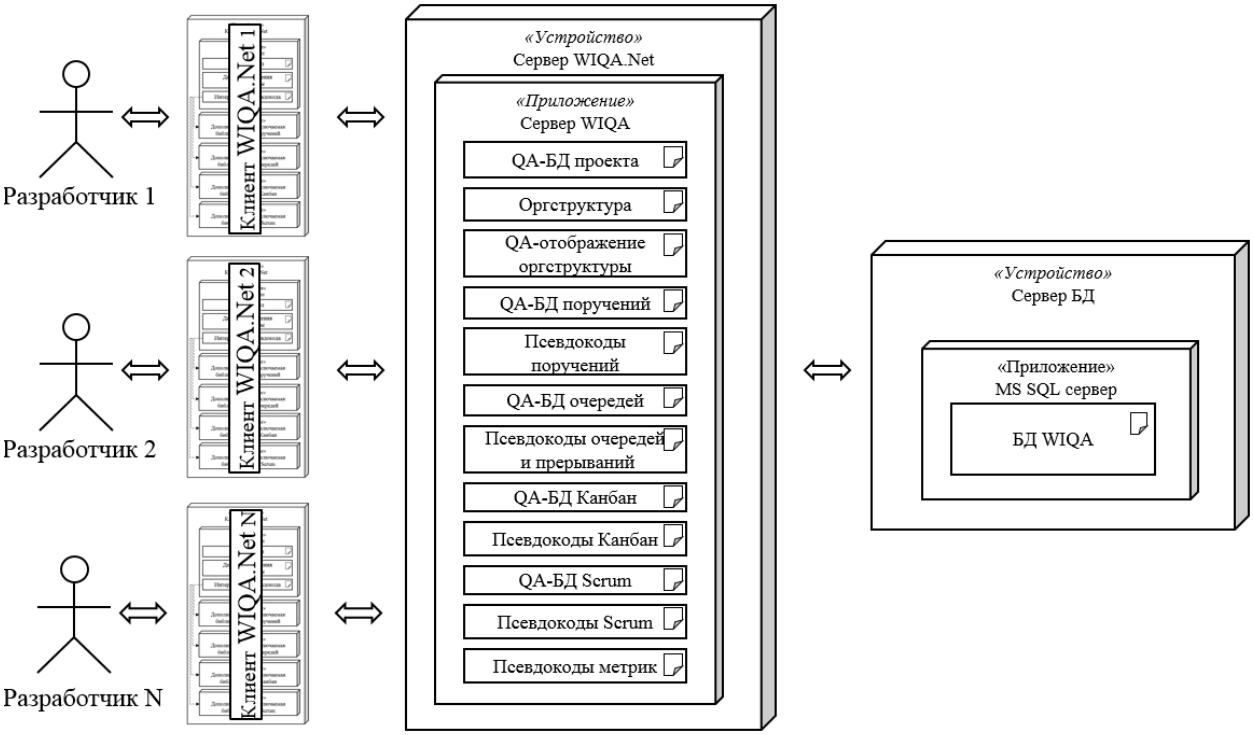


Рис. 4.2. Взаимодействие разработчиков и системы WIQA.Net

Перейдем к особенностям размещения компонентов ПКУ в среде ***OwnWIQA***. Эта среда предполагает работу в ней одного проектировщика, и, соответственно, все её компоненты размещены на одном устройстве. При этом теряется необходимость размещения ***клиента WIQA*** и ***сервера WIQA*** в различных устройствах, что позволяет объединить их функциональность в одном приложении. Среда ***Own*n*WIQA*** не содержит в себе инструментария для работы с оргструктурой, соответственно, в ней не требуется использования каких-либо демонов для работы с ней. В то же время, для выполнения программных средств ПКУ требуется наличие имитации оргструктуры,

реализованной в виде ***QA***-модели оргструктуры, а также – набора псевдокодовых инструментов для работы с ней. Для обеспечения взаимодействия этих инструментов с пользователем была также реализована дополнительная подключаемая библиотека GUI имитации оргструктуры.

Кроме этого, для обеспечения взаимодействия между сотрудниками был введен инструментарий синхронизации, обеспечивающий синхронизацию данных ПКУ посредством репликации базы данных ***WIQA***.

Итак, компоненты клиента и сервера ***WIQA***, которые были исключены из набора компонентов ПКУ для реализации в среде ***OwnWIQA***:

- Оргструктура;
- Демон отображения оргструктуры;

При этом в объединенное приложение ***WIQA*** были добавлены следующие компоненты:

- ***QA-БД имитации оргструктуры*** – вопросно-ответное представление оргструктуры для обеспечения функционирования средств ПКУ;
- ***Псевдокоды имитации оргструктуры*** – программные компоненты, обеспечивающие работу с данными, имитирующими оргструктуру;
- ***Синхронизатор*** – инструментарий, обеспечивающий репликацию БД ***WIQA*** для обеспечения взаимодействия средств ПКУ со средствами ПКУ других участников проекта.

Кроме этого, была реализована ещё одна библиотека:

- **Дополнительная подключаемая библиотека GUI имитации оргструктуры**.

Представим диаграмму размещения компонентов ПКУ в среде ***OwnWIQA***:

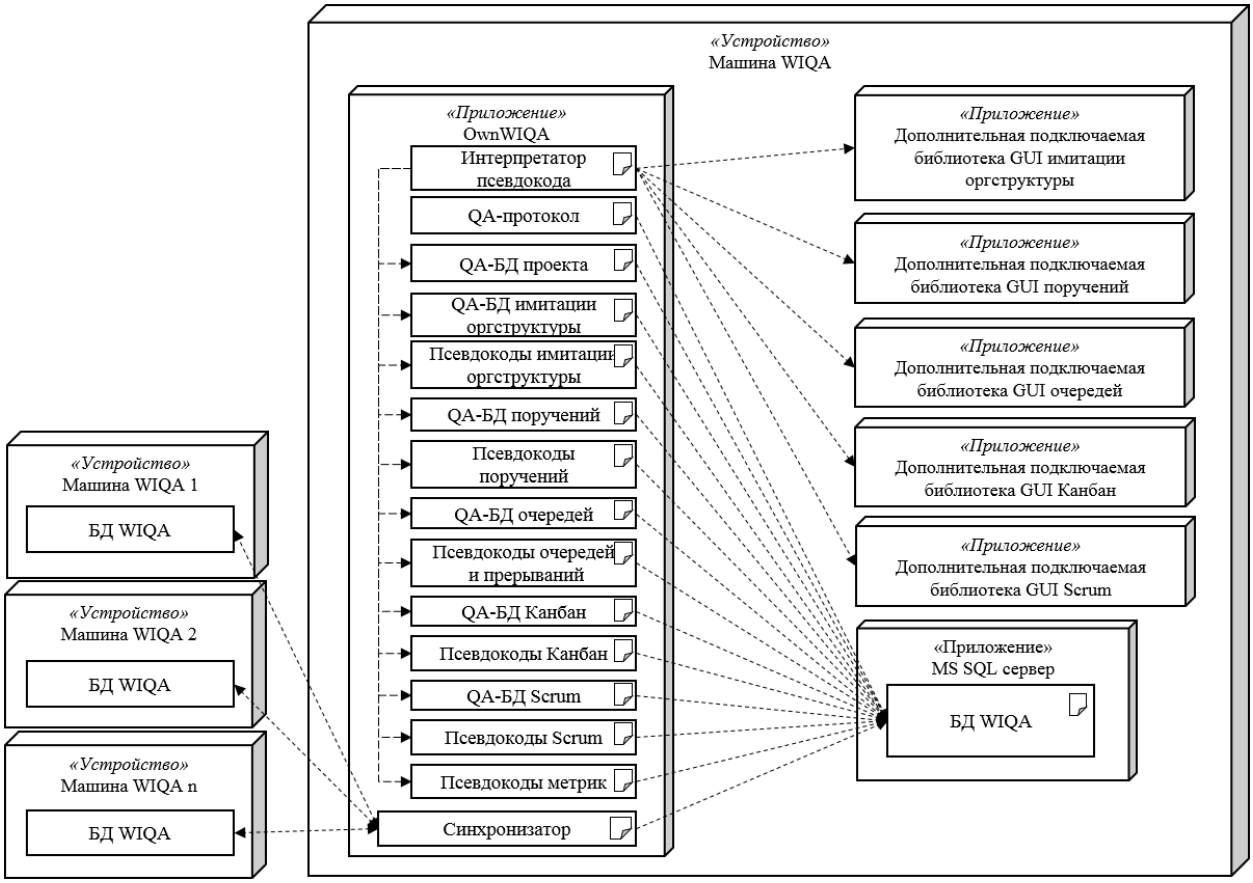


Рис. 4.3. Диаграмма размещения компонентов ПКУ в среде OwnWIQA

Каждый участник проектного процесса в среде ***OwnWIQA*** работает с собственным экземпляром машины ***WIQA***, которые синхронизируются между собой путем репликации их баз данных. Этот процесс показан на рисунке 4.4.

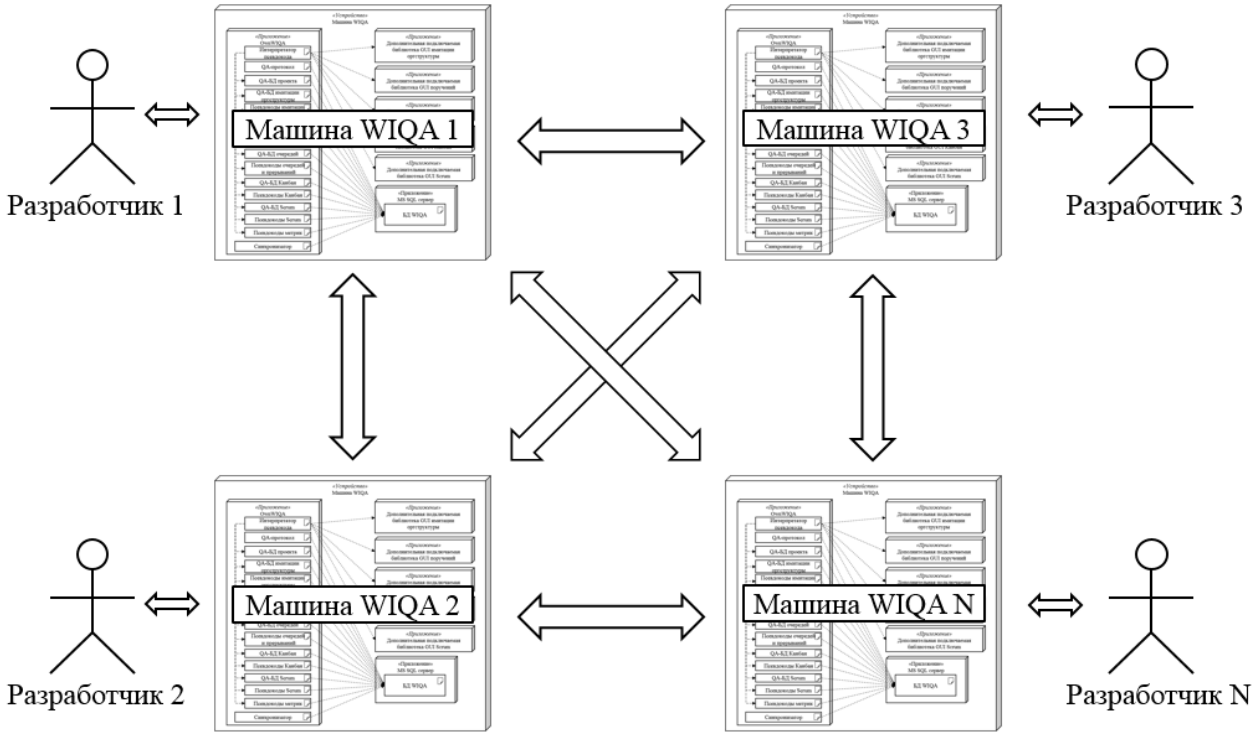


Рис. 4.4. Взаимодействие разработчиков и системы OwnWIQA

Далее в следующих параграфах представим описание реализации ПКУ с позиции отдельных её компонентов.

4.2. Особенности реализации первой версии комплекса средств ПКУ

4.2.1. Общие особенности реализации средств ПКУ

Комплекс средств ПКУ реализован в двух версиях среды ***WIQA***:

- Многопользовательская сетевая ***WIQA.Net***;
- ***OwnWIQA***, рассчитанная на одного пользователя.

Эти две системы объединяет то, что они содержат универсальные для них инструменты выполнения псевдокодовых программ на языке L^{WIQA}. На этом языке реализованы все структуры данных средств ПКУ за исключением структур данных, хранящие в себе проект и организационную структуру.

Подсистемы ПКУ объединяет то, что их реализация опирается на следующие элементы:

- Описанная на языке L^{WIQA} или отображаемая в ***QA***-память ***WIQA***

структура данных подсистемы;

- Реализованный на $\boldsymbol{L^{WIQA}}$ алгоритм работы подсистемы;
- Набор диалоговых окон, реализованный в виде дополнительной подключаемой библиотеки для языка $\boldsymbol{L^{WIQA}}$.

Такие подсистемы ПКУ, как Kanban, Scrum опираются на модель организации, реализованной в виде оргструктуры ***WIQA.Net***. Однако эти инструменты могут быть применены не только для представления работы групп проектировщиков, но также и для представления индивидуальной работы проектировщика в том случае, если он участвует в нескольких проектах, параллельно в нескольких проектных группах или играет несколько ролей, выполняя проектные задачи. Чтобы иметь возможность воспользоваться этими инструментами в среде ***OwnWIQA***, возникла необходимость в имитации оргструктуры в этой среде. В этом случае корнем дерева оргструктуры становится сам проектировщик, узлами дерева – группы, соответствующие его проектам, группам и ролям.

Для реализации древовидной имитации оргструктуры рациональным является использование ***QA***-памяти, хранящей данные в древовидной форме. Поскольку эта имитация является общей для всех проектов, в которых задействован разработчик, было принято решение создать для неё отдельный проект, содержащий как древовидную структуру организации, так и псевдокодовые реализации механизмов её управления. Реализация имитации оргструктуры в ***QA***-памяти обеспечивает возможность доступа к ней из других псевдокодовых программ, погруженных в ***OwnWIQA***, и, соответственно, она обеспечивает доступ для псевдокодовой реализации компонентов ПКУ.

Итак, имитация оргструктуры состоит из двух основных блоков:

- База данных имитации оргструктуры;
- Реализация механизмов управления.

4.2.2. Особенности реализации подсистемы контроля поручений

Основная особенность подсистемы контроля поручений в ***WIQA.Net*** является следствием наличия в ней организационной структуры, которая содержит акторов контроля поручений. ***WIQA.Net*** хранит дерево организационной структуры в ***SQL***-базе данных отдельно от деревьев проектов и задач. В то же время псевдокодовые программы оперируют с данными, представленными в ***QA***-памяти, хранящимися в виде задач, вопросов и ответов. Таким образом, для обеспечения доступа к данным оргструктуры из псевдокодовых программ необходимым является отображение оргструктуры на ***QA***-память. Для этого средствами ПКУ используется демон, который также востребован другими инструментальными средствами ***WIQA.Net***

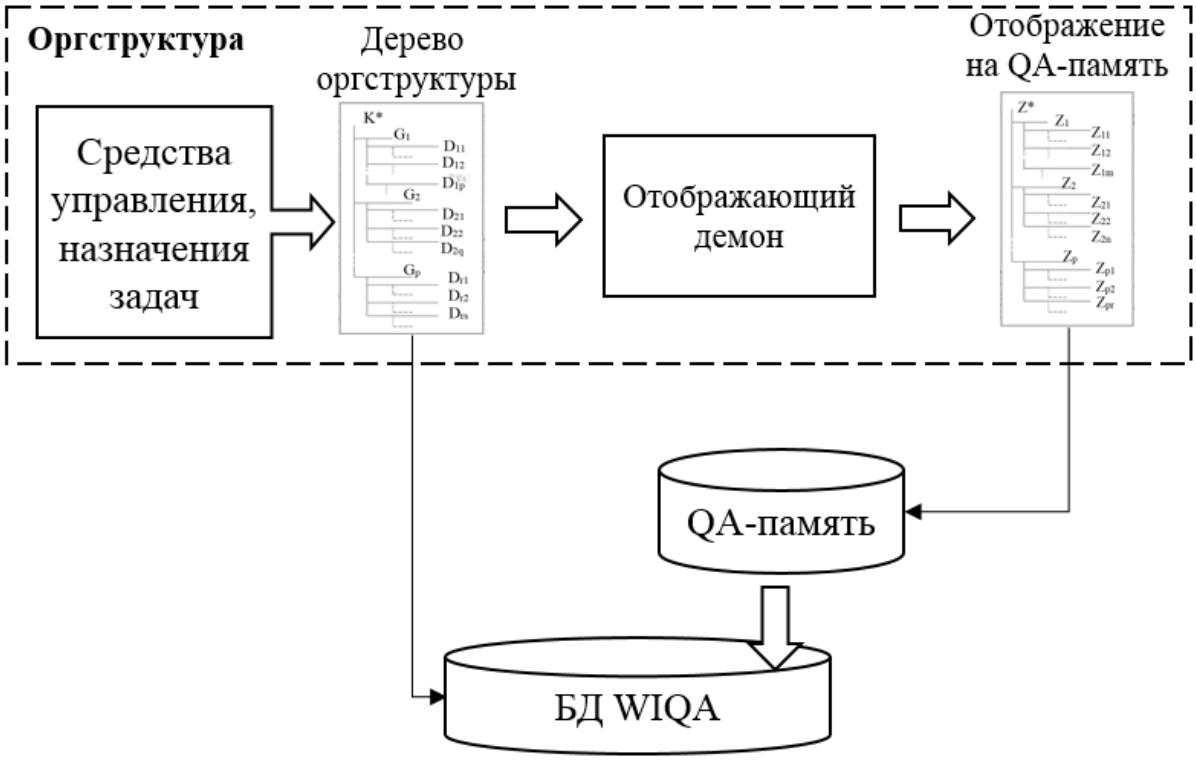


Рис. 4.5 Отображение оргструктуры на QA-память WIQA

В результате работы данного демона происходит формирование проекта "Организационная структура", содержащего две задачи:

- Сотрудники;

- Группы.

А также, для каждого проекта, в котором было произведено назначение, создается вопрос, подчиненный непосредственно проекту. В этот вопрос в виде подчиненных элементов происходит копирование назначенной задачи и всех её предков с сохранением иерархической структуры. В назначенной задаче происходит формирование вопроса "Дополнительно", который характеризует назначенного на выполнение задачи проектировщика и некоторые дополнительные свойства назначения.

Для хранения данных в подсистеме контроля поручений был разработан ряд сущностей.

Основной сущностью, хранящей в себе описания поручений, является сущность «***Поручения***». Она включает в себя следующие поля:

Таблица 4.1. Сущность «***Поручения***»

Поле	**Комментарий**
ID	Уникальный идентификатор поручения
Причина	Причина назначения поручения
Содержание	Текстовое описание поручения
Тип поручения	Тип
Назначен	Группа субъектов, которой назначается поручение
Руководитель	Субъект
Автор	Субъект
Приоритет	Числовое значение, показывающее важность скорейшего выполнения данного поручения
Санкционирующий контролер	Лица, занимающиеся контролем над исполнением поручения
Переписывающий контролер	
Утверждающий контролер	
Снят с контроля	Флаг, указывающий, что поручение снято с контроля

Дата переноса	При откладывании исполнения поручения указывается дата переноса
Дата исполнения	Дата исполнения поручения
Оценка	Оценка, выставленная за исполнение поручения
Лицо резолюции	Указывается лицо резолюции, если оно назначено
Статус поручения	Состояние поручения

Как видно из таблицы 4.1, сущность «***Поручения***» содержит поля, которые могут повторяться для разных поручений. Для минимизации ошибок вводятся дополнительные, связанные с сущностью «Поручения» сущности.

Сущность «***Рабочая группа***», описывающие группы лиц, занятые определенными видами деятельности, представлена в таблице 4.2.

Таблица 4.2. Сущность «***Рабочая группа***»

Поле	**Комментарий**
ID	Уникальный идентификатор группы
Номер группы	Идентификатор группы, используемый в других корпоративных приложениях
Название группы	Текстовое название группы

Каждый проектировщик принадлежит к той или иной группе, и система управления поручениями должна хранить данные каждого сотрудника. Для этого вводится сущность «***Субъект***».

Таблица 4.3. Сущность «Субъект»

Поле	**Комментарий**
ID	Уникальный идентификатор субъекта
ID группы	Идентификатор группы, к которой принадлежит сотрудник
ФИО	Фамилия, имя, отчество сотрудника
Контакты	Контактные данные сотрудника

В процессе выполнения поручений или по результатам их выполнения сотрудники или их группы отчитываются по их выполнению. Для хранения данных по отчетам о выполнении поручений вводится сущность «***Отчеты***».

Таблица 4.4. Сущность «Отчеты»

Поле	**Комментарий**
ID	Уникальный идентификатор отчета
ID поручения	Идентификатор поручения
ID автора	Автор отчета
Дата	Дата сдачи отчета

Для функционирования и поддержания целостности данной системы между сущностями устанавливаются следующие связи:

Таблица 4.5. Связи сущностей

Исходная сущность	**Связанная сущность**	**Тип связи**
Рабочие группы	Поручения	Один ко многим
Субъекты	Поручения	Один ко многим
Рабочие группы	Субъекты	Один ко многим
Поручения	Отчеты	Один ко многим
Субъекты	Отчеты	Один ко многим

Для реализации базы данных было использовано расширение языка $\boldsymbol{L^{WIQA}}$ по работе с базами данных. Это расширение предполагает хранение базы данных в виде вопросно-ответного дерева, в котором вопросами является табличная структура, а данные хранятся в виде ответов на вопросы, соответствующие заголовкам столбцов таблиц.

Типы данных в столбцах таблиц, ключевые поля, а также отношения между таблицами представлены в виде дополнительных атрибутов единиц вопросно-ответного дерева.

Общий вид дерева баз данных в вопросно-ответном протоколе представлен на следующем рисунке:

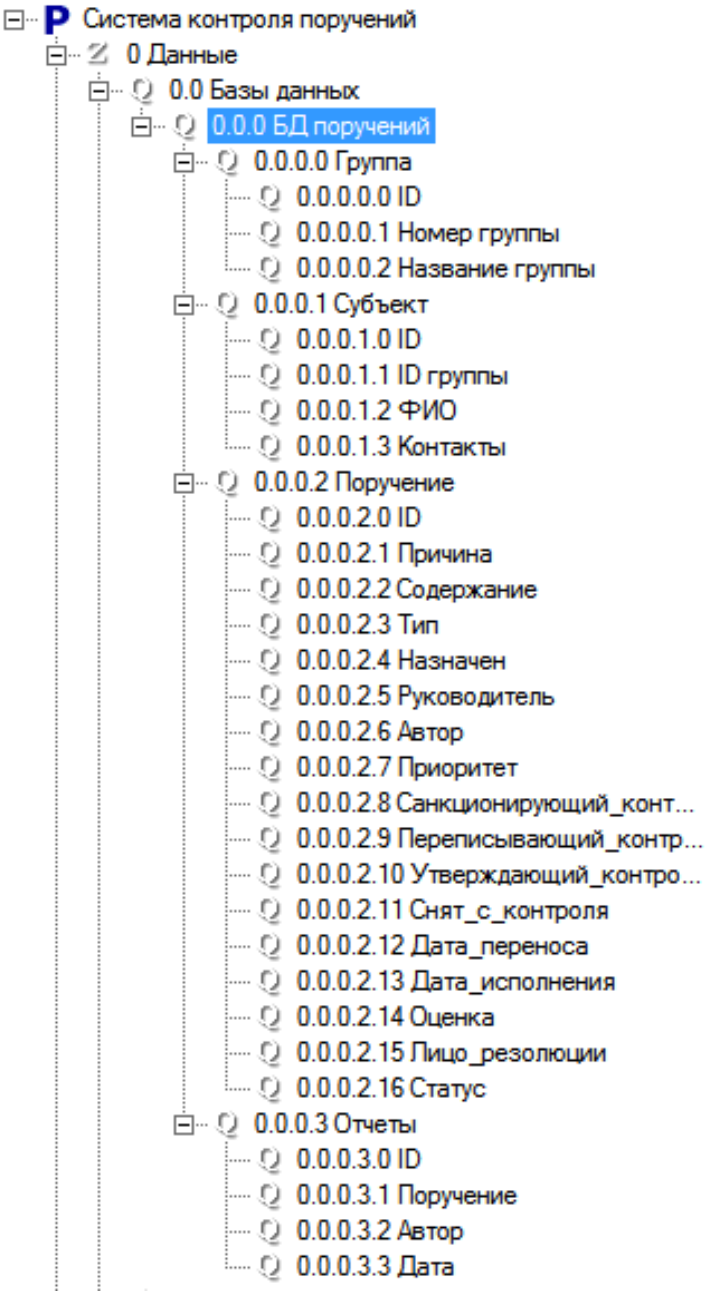


Рис. 4.6. БД поручений в QA-памяти WIQA

Теперь перейдем к реализации процедурной части подсистемы контроля поручений. В соответствии с использованием системы управления поручениями, реализуется ряд псевдокодовых функций, каждая из которых ответственна за выполнение того или иного действия. Пользователь в процессе работы с системой вызывает ту или иную, необходимую ему в данный момент процедуру.

Псевдокодовые процедуры, используемые в реализации:

Таблица 4.6. Псевдо-кодовые процедуры системы контроля исполнения поручений

Название процедуры	**Описание**
&Получить_Индексы&	Процедура отыскивает QA-индексы таблиц и заголовков столбцов в псевдокоде и помещает их в переменные для быстрого доступа к ним. Эта процедура используется в других процедурах
&Добавить_группу&	Добавление группы в таблицу групп
&Добавить_субъект&	Добавление сотрудника в группу
&Создать_поручение&	Создание поручения
&Просмотр_поручений&	Просмотр поручений, их состояний, отчетов по поручениям
&Редактировать_поручение&	Редактирование поручений
&Поиск_назначений&	Поиск назначений в оргструктуре и формирование на их основе новых поручений

Рассмотрим псевдокодовую реализацию этих псевдокодовых процедур на примере процедуры ***&Просмотр_поручений&***:

Процедура просмотра поручений и аналогичная ей процедура редактирования поручений также выполнены как процедуры работы с базой данных на расширенном языке L^{WIQA}. Далее представлен код этих процедур:

```
Q 2.5 PROCEDURE &Просмотр_поручений&
     Q 2.5.1 CALL &Получить_индексы&
     Q 2.5.2 &groups& := ExecuteSQL(&dbid&, "SELECT ID, Название группы
FROM Группа")
     Q 2.5.3 &tasks& := ExecuteSQL(&dbid&, "SELECT ID, Содержание FROM
Поручение")
     Q 2.5.4 &Query& := AC_FINDTASK(&groups&, &tasks&)
     Q 2.5.5 &foundtasks& := ExecuteSQL(&dbid&, &query&)
     Q 2.5.6 &Query& := AC_SelectTask(&foundtasks&)
     Q 2.5.7 &task& := ExecuteSQL(&Query&)
     Q 2.5.8 &persons& := ExecuteSQL(&dbid&, "SELECT ID, ФИО FROM
Участник")
     Q 2.5.9 AC_VewTasks(&persons&, &groups&, &task&)
     Q 2.5.10 ENDPROC &Просмотр_Поручений&.
```

Данная процедура осуществляет вызов ряда функций, таких, как ***AC_FINDTASK***, ***AC_SELECTTASKS***, ***AC_VIEWTASKS***, которые запрограммированы в виде дополнительной подключаемой библиотеки TaskControlInt. Эти функции отвечают за отображение диалоговых окон, причем в качестве входных параметров они используют результаты ***SQL-запросов*** к БД поручений.

Рассмотрим реализацию дополнительной подключаемой библиотеки ***TaskControlInt***. В этой библиот реализованы следующие функции:

Таблица 4.7. Функции библиотеки отображения диалоговых окон системы контроля поручений.

Функция	**Описание**
AC_CREATEPERSON	Окно добавления участника
AC_CREATEGROUP	Окно добавления рабочей группы
AC_ADDTASK	Окно добавления поручения
AC_EDITTASK	Окно редактирования поручения (окно добавления поручения с измененным заголовком окна)
AC_VIEWTASK	Окно просмотра поручения (Окно добавления поручения с измененным заголовком окна и с неизменяемыми полями)
AC_FINDTASK	Окно поиска поручения по заданным параметрам
AC_SELECTTASK	Окно выбора поручения

Для редактирования поручения используется та же самая форма, что и для создания нового с измененным заголовком окна. Перед отображением окна редактирования поручения псевдо-кодовая процедура отображает окно выбора поручения, позволяя пользователю выбрать поручение для редактирования.

4.2.4. Особенности реализации Kanban

База данных KANBAN реализуется как вопросно-ответная псевдо-кодовая база данных с целью облегчения доступа к ней из разработанных на языке $\boldsymbol{L^{WIQA}}$ средств управления потоками работ.

База данных содержит две таблицы: Задачи и Ассоциации. Далее представлены поля этих таблиц:

Таблица 4.8. Таблица «Задачи» БД KANBAN

Поле	**Тип**	**Описание**
Id	[PK] Целое число	Уникальный идентификатор записи
id в поручениях	Целое число	Идентификатор этой задачи в базе данных системы управления поручениями
Описание	Строка	Название задачи

Детальное описание	Строка	Детальное описание задачи
QAId_задачи	Целое число	Идентификатор задачи в вопросно-ответном дереве

Таблица 4.9. Таблица «Ассоциации» БД KANBAN

Поле	**Тип**	**Описание**
Id	[PK] Целое число	Уникальный идентификатор записи
id_задачи	[FK]Целое число	Идентификатор ассоциированной задачи
id_колонки	Целое число	Порядковый номер колонки, к которой относится задача
Состояние	Целое число	Состояние выполнения задачи. 0 – в процессе, 1 – готово.
id_группы	Целое число	Идентификатор группы разработчиков из системы контроля поручений
Отметка	Целое число	0 – задача не отмечена флажком, 1 – отмечена.

В вопросно-ответном протоколе эта база данных выглядит следующим образом:

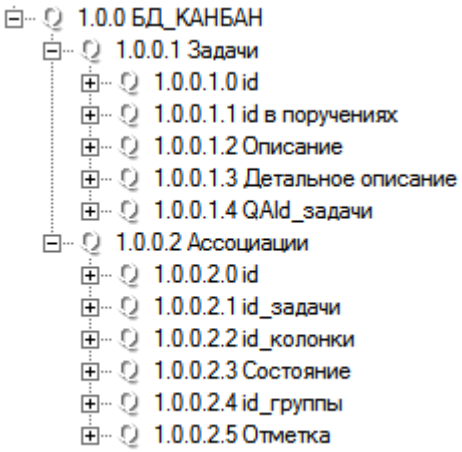


Рис. 4.7. База данных доски KANBAN

Для визуализации доски разработана псевдокодовая процедура ShowKanban, выполняющая запросы к базам данных и вызывающая функцию визуализации доски.

Данный метод возвращает сроку, содержащую запросы к базе данных, разделенные разделителями. Запросы добавляются в строку в результате следующих событий в процессе работы в главной форме:

- Изменение текста детального описания задачи
- Установка или снятие отметки на задаче
- Завершение срока выполнения шага задачи

Графический интерфейс доски KANBAN

При вызове процедуры ShowKanban после осуществления работы с базами данных происходит вызов функции KANBAN_SHOW, отображающей окно, представленное на рисунке 4.8.

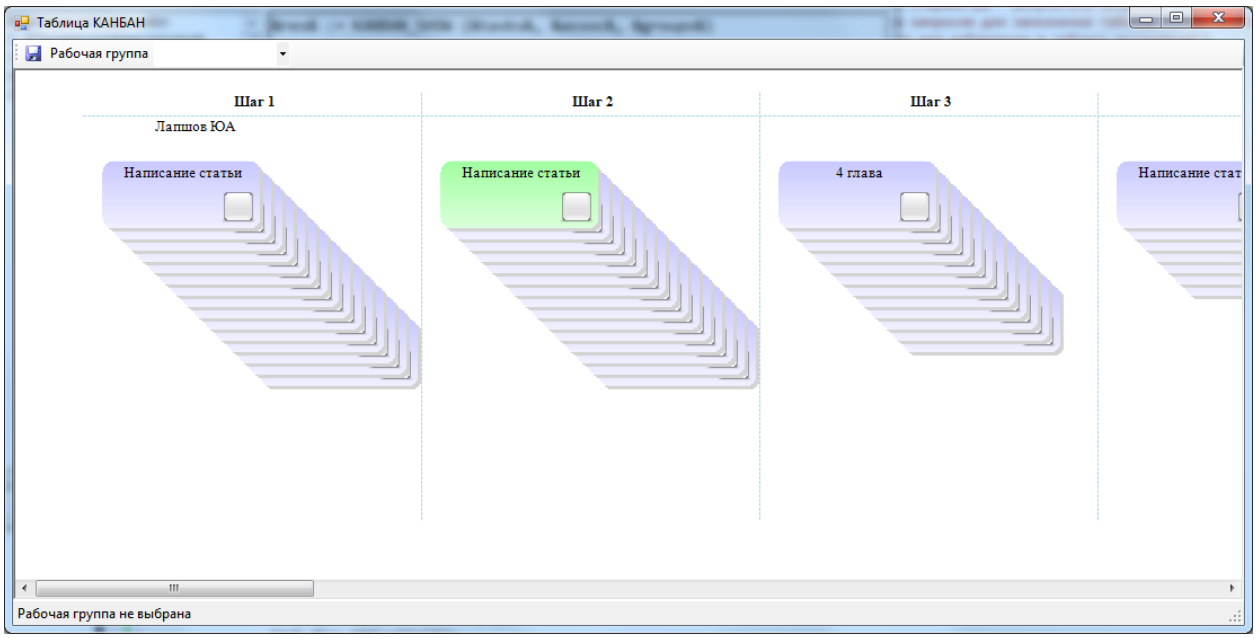


Рис. 4.8. Основное окно доски KANBAN

В данном окне представлена сетка, по вертикали в которой расположены группы разработчиков, а по горизонтали – шаги выполнения задач. В ячейках сетки располагаются карточки. Синий цвет соответствует задаче, находящейся

на данном этапе в состоянии работы, а зеленый – завершенной на данном этапе задаче. Сверху располагается панель инструментов. Окно содержит полосы прокрутки, при их перемещении происходит перемещение по доске:

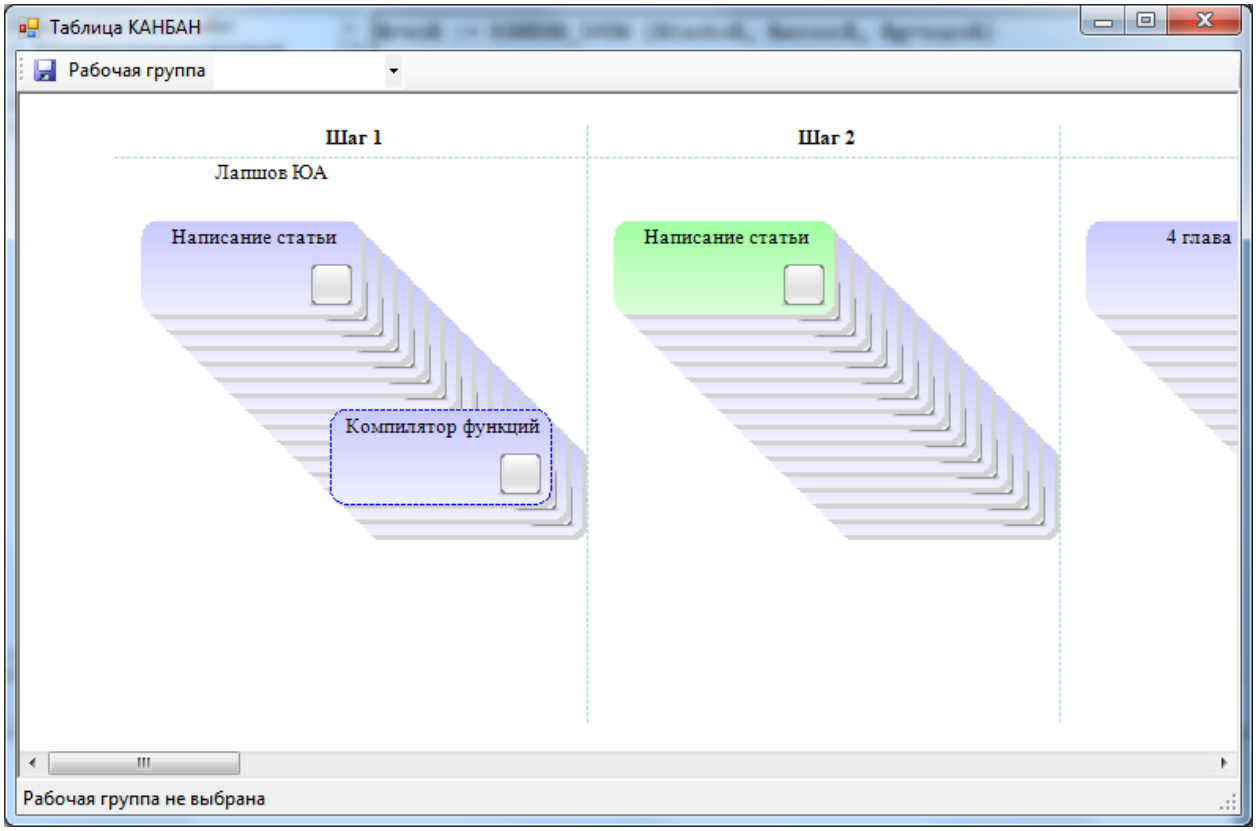


Рис. 4.9. Выделение карточки

При щелчке мышью по иконке-флажку происходит его установка или снятие. Установленные или снятые флажки сохраняются в базе анных и восстанавливаются после перезапуска средства визуализации доски KANBAN.

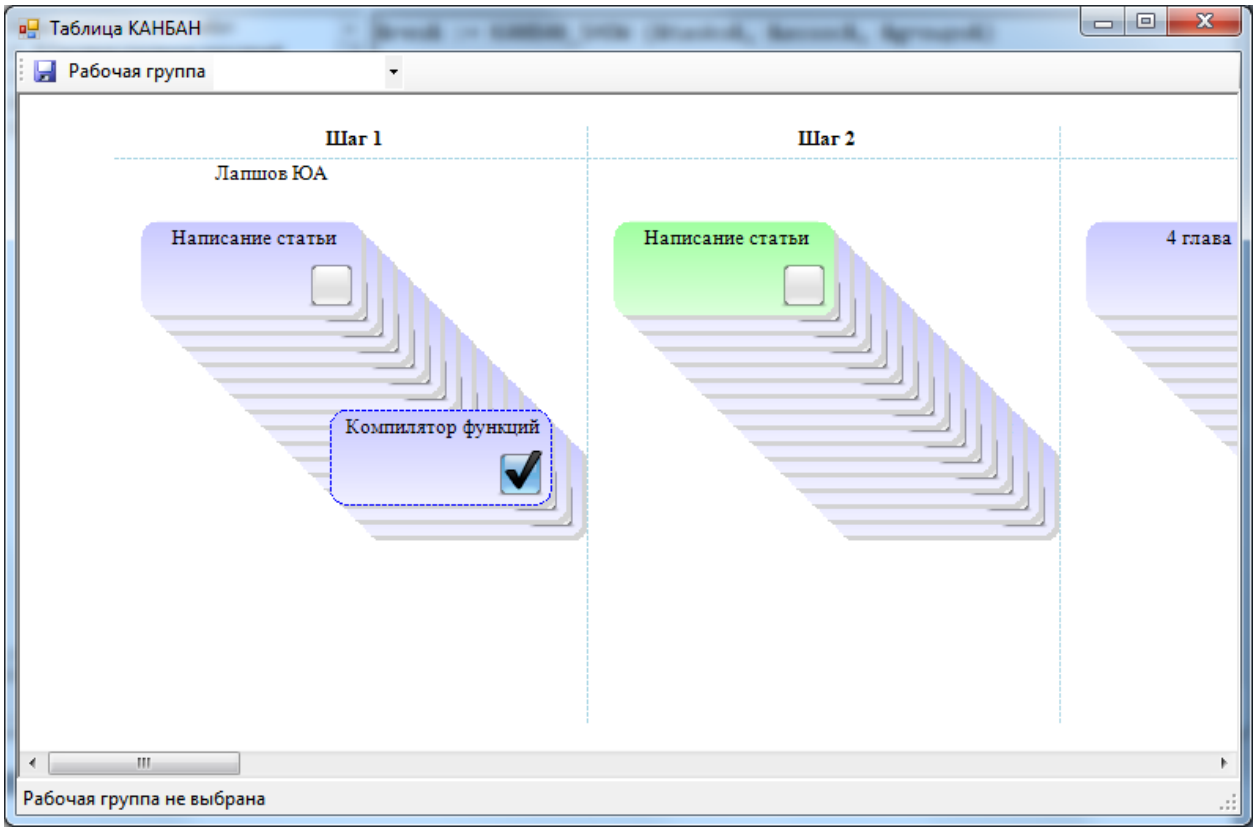


Рис. 4.10. Флажок

При двойном щелчке мышью по карточке, карточка раскрывается, демонстрируя редактируемое детальное описание задачи (рисунок 4.11). На раскрытой карточке также имеется возможность установки или снятия флажка отметки. В нижней части карточки около флажка имеются кнопки «Перевернуть карточку» и «Открыть интерпретатор». При нажатии на кнопку «Перевернуть карточку» отображается её обратная сторона, на которой находятся ассоциированные с задачей элементы управления. При нажатии «Открыть интерпретатор» окно доски KANBAN закрывается и открывается ещё один экземпляр интерпретатора псевдокода на задаче, соответствующей выбранной карточке (рисунок 4.12).

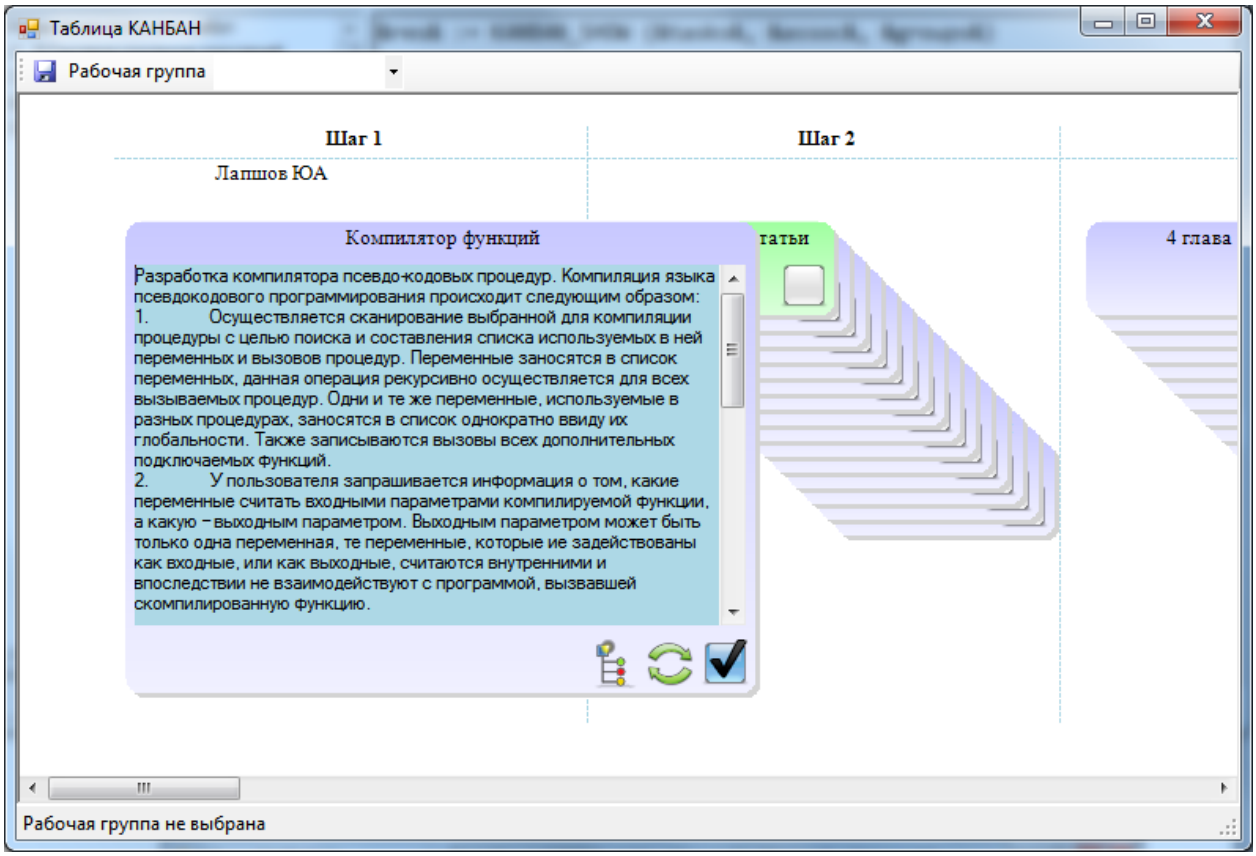


Рис. 4.11. Раскрытая карточка

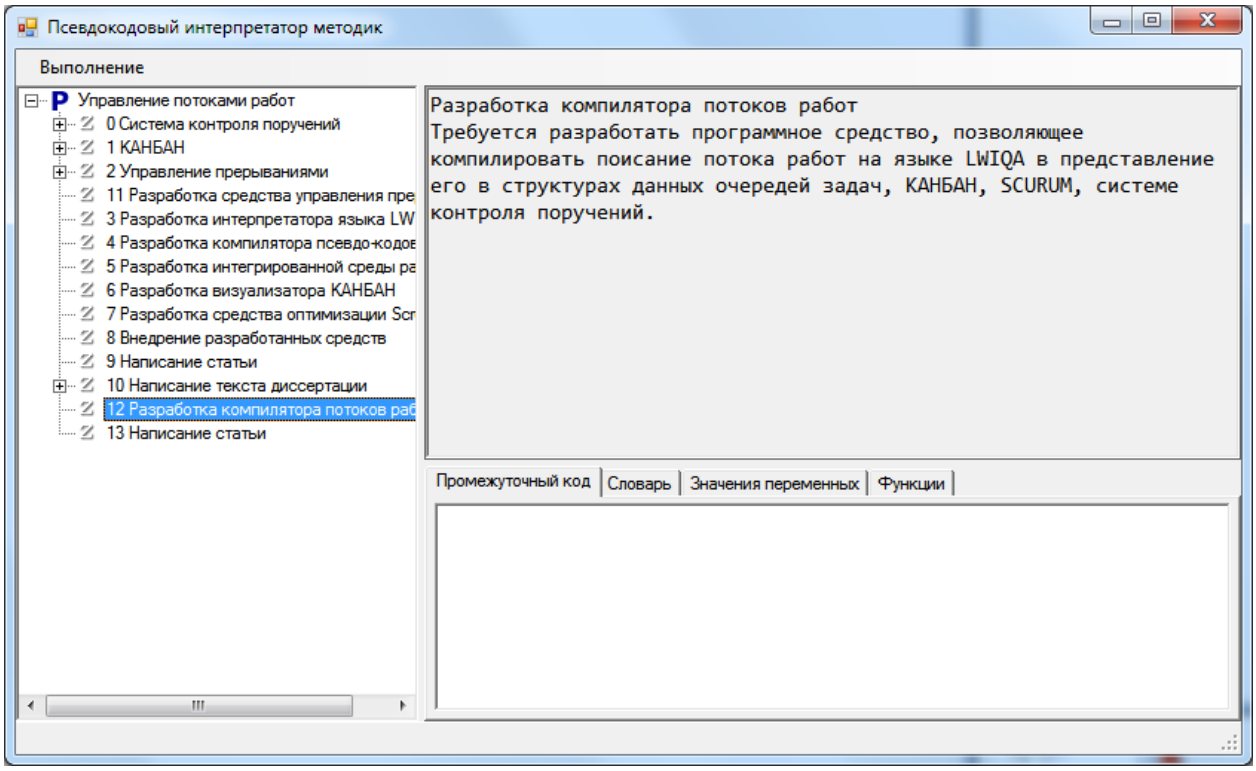


Рис. 4.12. Интерпретатор открывается на выбранной задаче

При щелчке мышью по колонке карточек открывается контекстное меню,

позволяющее выбрать необходимую карточку (рисунок 4.13)

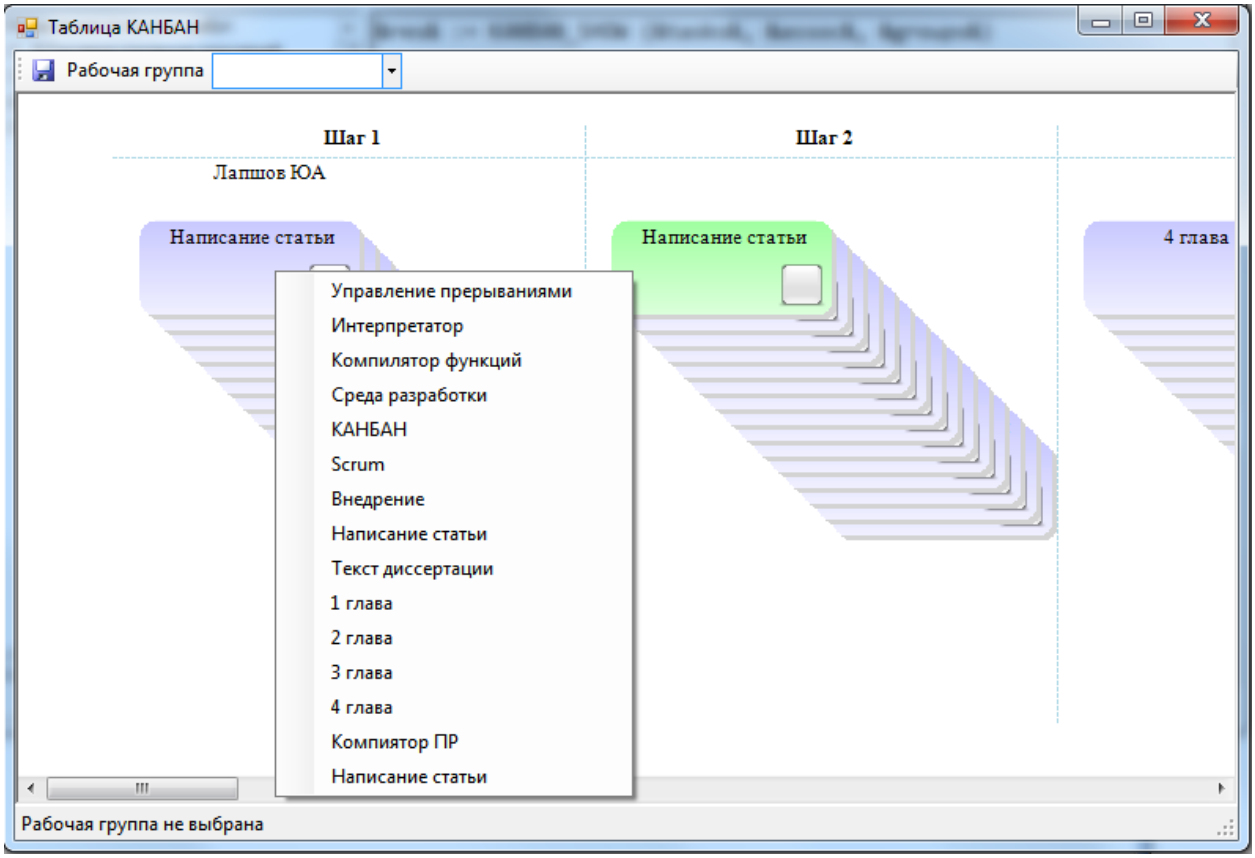


Рис. 4.13. Контекстное меню стопки карточек

При щелчке на пустом поле около стопки карточек открывается контекстное меню, предлагающее действия над всей стопкой карточек (рисунок 4.13). Для экспорта информации об отмеченных задачах в Microsoft Word необходимо нажать кнопку с изображением дискеты. Откроется стандартное диалоговое окно сохранения файла и при указании корректного имени файла после нажатия кнопки «Сохранить» будет произведен экспорт в формат Microsoft Word.

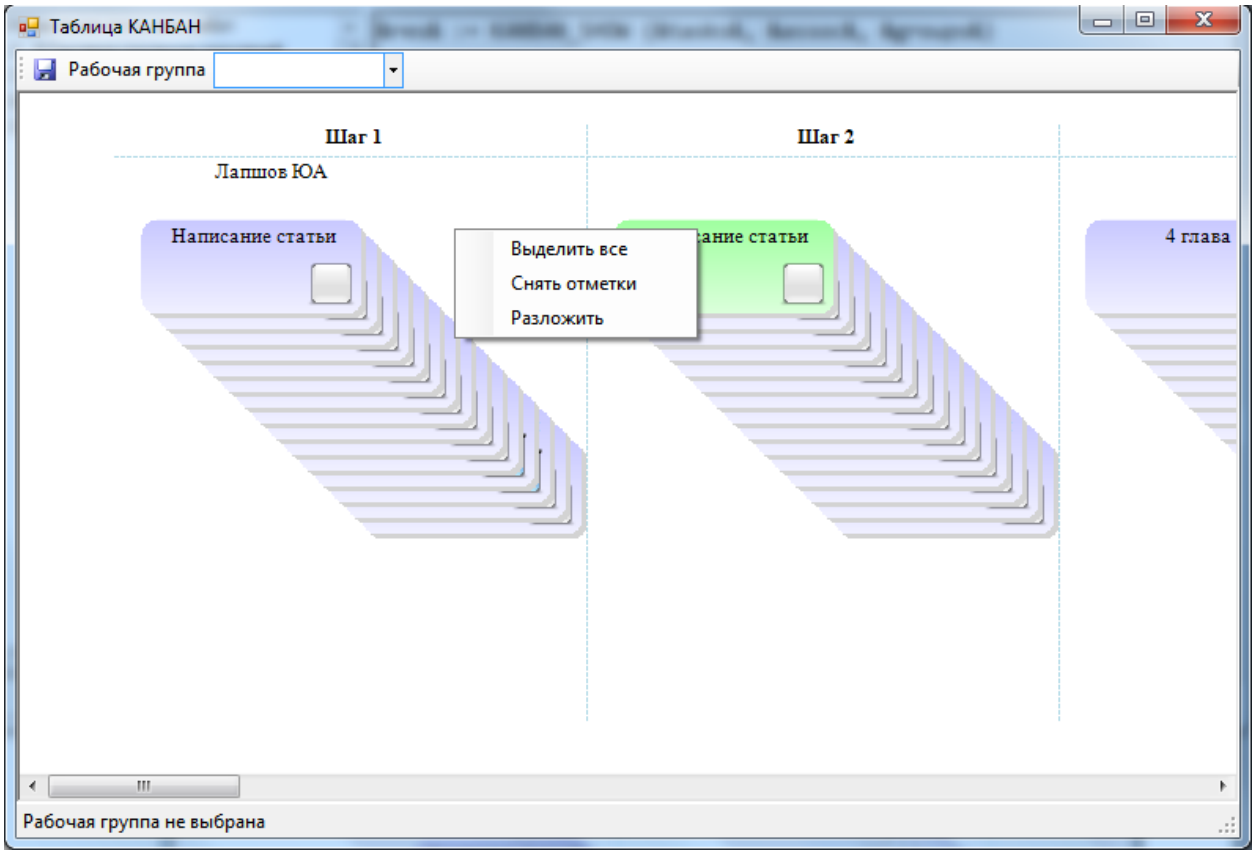


Рис. 4.14. Контекстное меню групповых операций

4.3. Экспериментальное исследование

Экспериментальное исследование данной работы включает в себя выполнение следующих проектов с использованием средств ПКУ:

- Проект «Создание единого интегрированного справочника», выполненный на предприятии ФНПЦ ОАО НПО «Марс»;
- Проект «Образовательная Web-система», выполненный на предприятии ООО «Центр технического творчества»;
- Проект, включающий в себя ряд задач, выполненных в рамках данного диссертационного исследования.

Также была экспериментальным путем проверена возможность использования средств ПКУ в качестве средства для сравнения результатов выполнения тестовых заданий на основе задач студенческой олимпиады, выполненной на кафедре «Вычислительная техника» УлГТУ.

Целью эксперимента, выполненного на проекте «Создание единого интегрированного справочника» является проверка следующей гипотезы:

H1. Применение ПКУ в проектировании позволяет рационализировать оперативное планирование этапов проектной работы.

Рассмотрим **условия проведения эксперимента**. Одним из потоков работ, над которыми выполнялись экспериментальные исследования в рамках данной диссертационной работы, является поток работ, выполняемый на предприятии **ФНПЦ ОАО НПО «Марс»** в рамках проекта **«Создание единого интегрированного справочника»** сотрудниками подразделения «НИЛАБ-111» предприятия. Эксперимент продолжался в период **с 1.10.2014 по 30.12.2014**. Задачи выполнял коллектив из 5 человек и руководителя:

Таблица 4.10. Проектная группа

Сотрудник	**Роль**	**Обозначение**
Михеев Петр Сергеевич	Руководитель	D1
Бородаенко Алексей Михайлович	Разработчик	D2
Гришин Артур Павлович	Разработчик	D3
Гайнуллин Дамир Каримович	Разработчик	D4
Лазарева Светлана Николаевна	Разработчик	D5
Чилипенко Алексей Алексеевич	Разработчик	D6

Имена участвующих в выполнении эксперимента людей изменены.

В качестве **входных параметров** выступают запланированные **сроки выполнения** задач проекта, а в качестве выходных параметров – **метрические характеристики** этапов проектной работы, выполненных в течение времени проведения эксперимента.

В качестве **плана эксперимента** выступает следующая методика:

1. Погрузить задачи и подзадачи проекта в среду WIQA.Net – *D1*;
2. Поместить в проект комплекс задач ПКУ – *D1*;
3. С помощью инструментария организационной структуры установить ассоциации между сотрудниками и исполняемыми ими задачами проекта – *D1*;

4. Создать для каждой задачи поручение, указав примерное время окончания работы над данной задачей – *D1*;
5. Сформировать первый спринт, в бэклог которого отобрать задачи для выполнения их в течение октября – *D1*;
6. Выполнить спринт;
   6.1. Перед началом рабочего дня провести 15-минутное совещание в соответствии с методологией Scrum, в течение которого выполнить следующие действия;
      6.1.1. Установить в диаграмме выгорания точку, соответствующую выполненной в течение предыдущего дня работе – *D1-D6*;
      6.1.2. Провести обсуждение планов работ на предстоящий рабочий день – *D1-D6*;
7. Рассчитать метрики и произвести оценивание планирования спринта по 5-балльной шкале – *D1*;
   7.1. Вычислить коэффициент x или y для Scrum-метрики TD в зависимости от того, были ли выполнены все запланированные задачи в течение времени выполнения спринта – *D1*;
8. Сформировать второй спринт для выполнения задач в течение ноября, учитывая скорость работы команды, определенную по результатам выполнения первого спринта – *D1, D2-D6*;
9. Выполнить второй спринт и вычислить метрики его выполнения – *D1, D2-D6*;
10. Сформировать третий спринт для выполнения задач в течение декабря, учитывая скорость работы команды, определенную по результатам выполнения первого и второго спринта – *D1, D2-D6*;
11. Выполнить третий спринт и вычислить метрики его выполнения – *D1, D2-D6*;
12. Провести сравнение эффективности планирования трех спринтов и

сделать выводы – *автор данной диссертационной работы*.

В процессе **выполнения данного эксперимента** для каждой задачи было сформировано поручение и установлена примерная дата выполнения этой задачи:

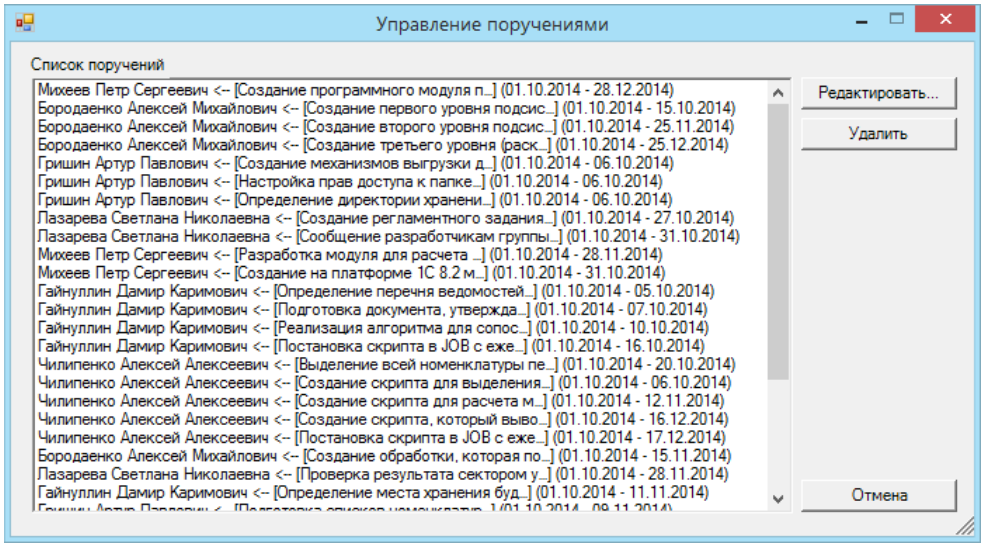


Рис. 4.15. Назначенные поручения

В процессе формирования поручений D0 был запрограммирован процесс параллельного выполнения этих проектных задач на языке $\boldsymbol{L^{WIQA}}$.

```
//Для каждой задачи было произведено назначение
&D1& = GetPersonId("Михеев Петр Сергеевич");
&Z1& = QA_GetQAId("Создание программного модуля плана");
SEIZE (&Z1&, &D1&);
```

Далее представлена таблица, содержащая набор задач проекта и обозначения их индексов на языке $\boldsymbol{L^{WIQA}}$.

Таблица 4.11. Задачи проекта

Задача	**Обозначение**
Анализ и сбор требований	&Z0&
Создание программного модуля плана выпуска технических средств с производства	&Z1&
Разработка модуля для расчета себестоимости на технические средства из плана производства	&Z2&
Определение места хранения будущего эталонного справочника	&Z3&
Подготовка списков номенклатуры по видам для проверки соответствия наименования требованиям ГОСТ	&Z4&
Создание регламента работы с будущим эталонным справочником	&Z5&
Определение перечня номенклатуры, которая сопоставлена с системой бухгалтерского учета и с ERP-системой	&Z6&
Обработка номенклатуры для сопоставления	&Z7&
Определение переченя ответственных лиц, которые будут устанавливать соответствия и установка соответствий	&Z8&

Выполнение проверки результата процедуры установления соответствий отделом снабжений	&Z9&
Загрузка полученных результатов в таблицу соответствий	&Z10&
Загрузить результаты проверки наименований в отдельную таблицу с сохранением всех внешних ключей	&Z11&
Проработка вопроса с загрузкой проверенной и сопоставленной информации в SWE PDM	&Z12&
Выполнить загрузку в справочники SWE PDM	&Z13&
Конструкторам сформировать дерево спецификации на примере прибора 108К с использованием номенклатуры, которая была загружена ранее	&Z14&

Задача ***Z0*** должна была выполниться самой первой из задач, до её выполнения нельзя приступать к выполнению других задач. После этого можно приступать к выполнению задачи ***Z1***, параллельно с которой можно выполнять задачи ***Z4***, ***Z5*** и ***Z6*** (шаблон ***Parallel Split*** потоков работ). После выполнения задачи ***Z1*** можно приступить к выполнению задачи Z2, после выполнения которой – к ***Z3*** (Шаблон ***Sequence***). После того, как задачи ***Z4***, ***Z5*** и ***Z6*** будут выполнены (Шаблон ***Synchronization***), можно приступить к последовательному выполнению оставшихся задач ***Z7*** – ***Z14*** в соответствии с шаблоном ***Sequence.***

Далее представим псевдокодовую программу данного потока работ, размещенную в поле «Условие» на карточках задач:

```
    //Для задачи Z0 условие отсутствует – она по умолчанию
выполнима всегда. Эти условия прописываются в поле Условие для
каждой карточки
    //Для задач Z1, Z4, Z5, Z6 условие выполнения зависит от
решения задачи Z0 - распараллеливание
    IF TaskFinished(&Z0&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z1
    IF TaskFinished(&Z0&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z4
    IF TaskFinished(&Z0&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z5
    IF TaskFinished(&Z0&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z6
    //Последовательное выполнение задач Z1 -> Z2 -> Z3
    IF TaskFinished(&Z1&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z2
    IF TaskFinished(&Z2&) THEN TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z3
    //Синхронизация перед выполнением задачи Z7 – в условии
    //её выполнения
    IF TaskFinished(&Z4&) && TaskFinished(&Z5&) &&
```

```
TaskFinished(&Z6&) THEN &TaskEnabled& = 1 ELSE
&Z7&.&TaskEnabled& = 0 //Условие на карточке задачи Z7
    //Далее – условия для задач Z8 – Z14, отвечающих за их
    //последовательное выполнение после задачи Z7
    IF TaskFinished(&Z7&) THEN &TaskEnabled& = 1 ELSE
&Z8&.&TaskEnabled& = 0 //Условие на карточке задачи Z8
    IF TaskFinished(&Z8&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z9
    IF TaskFinished(&Z9&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z10
    IF TaskFinished(&Z10&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z11
    IF TaskFinished(&Z11&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z12
    IF TaskFinished(&Z12&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z13
    IF TaskFinished(&Z13&) THEN &TaskEnabled& = 1 ELSE
&TaskEnabled& = 0 //Условие на карточке задачи Z14
```

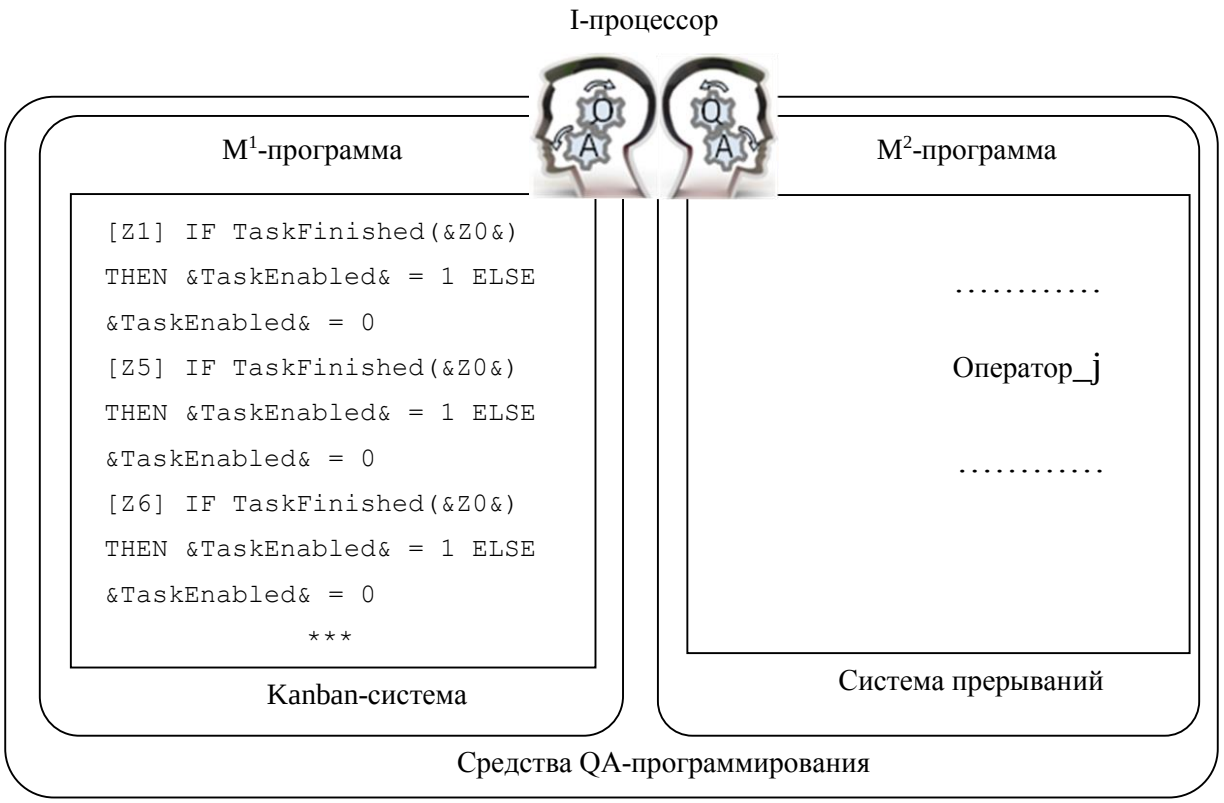


Рис. 4.16. Программирование параллелизма в потоках работ

В данных условиях используется функция ***TaskFinished***, осуществляющая проверку отметки о выполнении на всех карточках задачи по заданному индексу в переменной ***&TaskId&***. Далее представлена псевдокодовая реализация ***TaskFinished***:

```
    &dbqaid& := QA_GetQAId(&current_project&, "БД_KANBAN")
    &dbid& := OpenDB(&dbqaid&)
    &query& := "SELECT Состояние FROM Ассоциации WHERE id_задачи
= "
    &query& := STRCAT_INT(&query&, &TaskId&)
    &res& := DB_QUERY(&dbid&, &query&)
    &pos& := 0
    &max& := DB_GETROWCOUNT(&res&)
    &result& := 1
    IF &max& == 0 THEN GOTO &L2& //Не найдена задача
    LABEL &l1&
        &done& := DB_GETCELLINT(&res&, &pos&, 0)
        &pos& := &pos& + 1
        IF &done& == 0 THEN &result& := 0
        IF &result& == 0 THEN GOTO &l2&
        IF &pos& < &max& THEN GOTO &L1&
    LABEL &L2&
    FINISH //Результат – переменная &result&
```

Объекты «Карточка» хранятся в виде записей в таблице «Ассоциации» базы данных «БД_KANBAN» инструментария Kanban. Доступ к карточке осуществляется путем вызова SQL-оператора SELECT, в который передаётся один из параметров, в данном случае – поле «id_задачи», представляющее собой идентификатор задачи в БД Kanban.

С целью проверки условий выполнения, представленных на карточках задач, был разработан алгоритм и представлен на языке $\boldsymbol{L^{WIQA}}$. Для выполнения условий используется стандартная функция ***ExecuteString***, осуществляющая выполнение переданной в неё строки псевдокода, при этом все переменные, доступные в точке её запуска, оказываются доступными и для исполняемого этой функцией кода. В условиях на карточках происходит установка значения переменной ***TaskEnabled***, отвечающей за разрешение выполнения задачи.

```
    &dbqaid& := QA_GetQAId(&current_project&, "БД_KANBAN")
    &dbid& := OpenDB(&dbqaid&)
    &query& := "SELECT id, Условие FROM Ассоциации"
    &res& := DB_QUERY(&dbid&, &query&)
    &pos& := 0
    &max& := DB_GETROWCOUNT(&res&)
    IF &max& == 0 THEN GOTO &L2& //Нет задач
    LABEL &l1&
        &TaskEnabled& := 0; //Переменная, разрешающая выполнение
        &Condition& := DB_GETCELLSTRING(&res&, &pos&, 1)
        IF STRCMP(&Condition&, "") == 1 THEN &TaskEnabled& := 1
```

```
ELSE ExecuteString(&Condition&)
        &TaskId& := DB_GETCELLINT(&res&, &pos&, 0)
        &nquery& := "UPDATE Ассоциации SET Разрешено = "
        &nquery& := STRCAT_INT(&nquery&, &TaaskEnabled&)
        &nquery& := STRCAT(&nquery&, " WHERE id_задачи = ")
        &nquery& := STRCAT_INT(&nquery&, &TaaskId&)
        &res& := DB_QUERY(&dbid&, &nquery&)
        &pos& := &pos& + 1
        IF &pos& < &max& THEN GOTO &L1&
    LABEL &L2&
    FINISH
```

Затем из этих задач был отобран бэклог для выполнения в течение октября 2014 года и сформирован спринт:

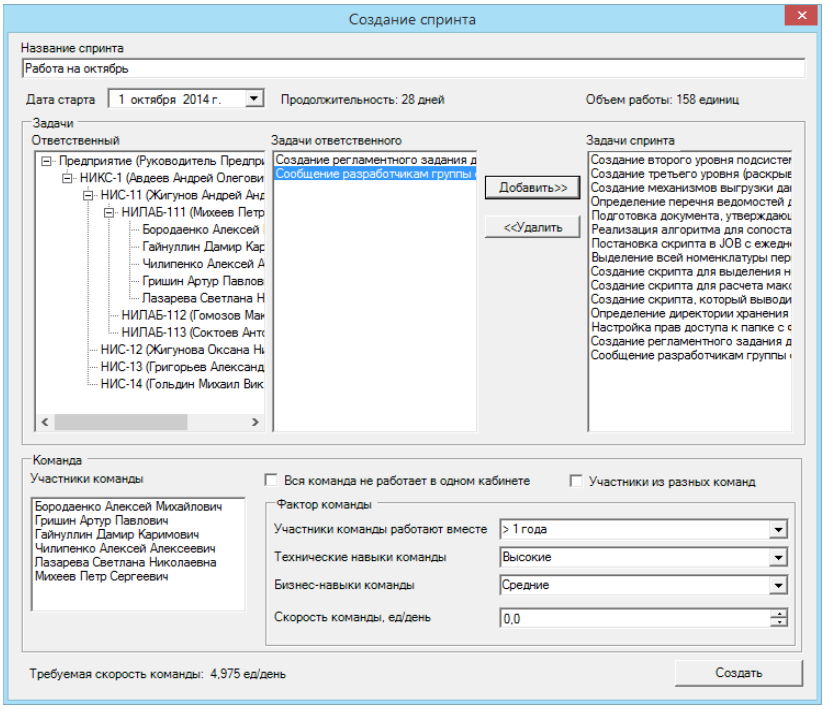


Рис. 4.17. Работы на октябрь

Для каждой подзадачи в процессе оперативного планирования участником D0 были описаны псевдокодовые условия их выполнения, объединяемые через логическое «И» с унаследованными от родительских задач условиями, представленными в программе потока работ.

В процессе выполнения спринта проектировщики псевдопараллельно выполняли возложенные на них задачи. Рассмотрим выполнение задач участником ***D4*** на 5 день спринта. В этот день очередь его задач состояла из четырех элементов – обозначим их Z4.1, Z4.2, Z4.3 и Z4.4

Эти задачи требовалось выполнять параллельно, причем по необходимости временных затрат они подразделялись на следующие категории, каждой из которых были установлены численные значения объемности:

Таблица 4.12. Объемность задач

Объемность	**Численное значение объемности**
Объемная	500
Средняя	300
Небольшая	100

Для каждой задачи была выбрана её характеристика объёмности:

Таблица 4.13. Численное выражение объемности

Задача	**Объемность**
Z4.1	Небольшая
Z4.2	Объемная
Z4.3	Средняя
Z4.4	Небольшая

В качестве кванта времени, соответствующего интервалу времени выполнения задачи с небольшой объемностью было выбрано 20 минут времени.

В процессе выполнения работы раз в минуту происходил пересчет критерия предпочтительности текущей задачи, и при достижении им нулевого значения происходило переключение к следующей задаче в очереди, а прерванная задача оказывалась в её конце.

Вычисление оставшихся единиц выполнения задачи происходило по следующей формуле:

$$Xz = Vz - (\Delta t * (Vz^{min} / Q)),$$

где Vz: характеристика объемности задач,

Δt – время, прошедшее от переключения к её выполнению,

Vz^{min} – минимальное численное значение объемности,

Q – квант времени, выделенный на задачу с минимальной объемностью.

В процессе работы проектировщика раз в минуту выполнялся демон,

рассчитывающий оставшиеся единицы времени для каждой задачи. Формула эта была запрограммирована на языке псевдокодового программирования для задач следующим образом:

Таблица 4.14. Вычисление критерия предпочтительности

Задача	**Вычисление**
Z4.1	&Xz& := 100 - (&tTime& * 5)
Z4.2	&Xz& := 500 - (&tTime& * 5)
Z4.3	&Xz& := 300 - (&tTime& * 5)
Z4.4	&Xz& := 100 - (&tTime& * 5)

Данные формулы были записаны в поле «Условия» на карточках задач, таким образом, было запрограммировано распараллеливание на уровне проектировщика.

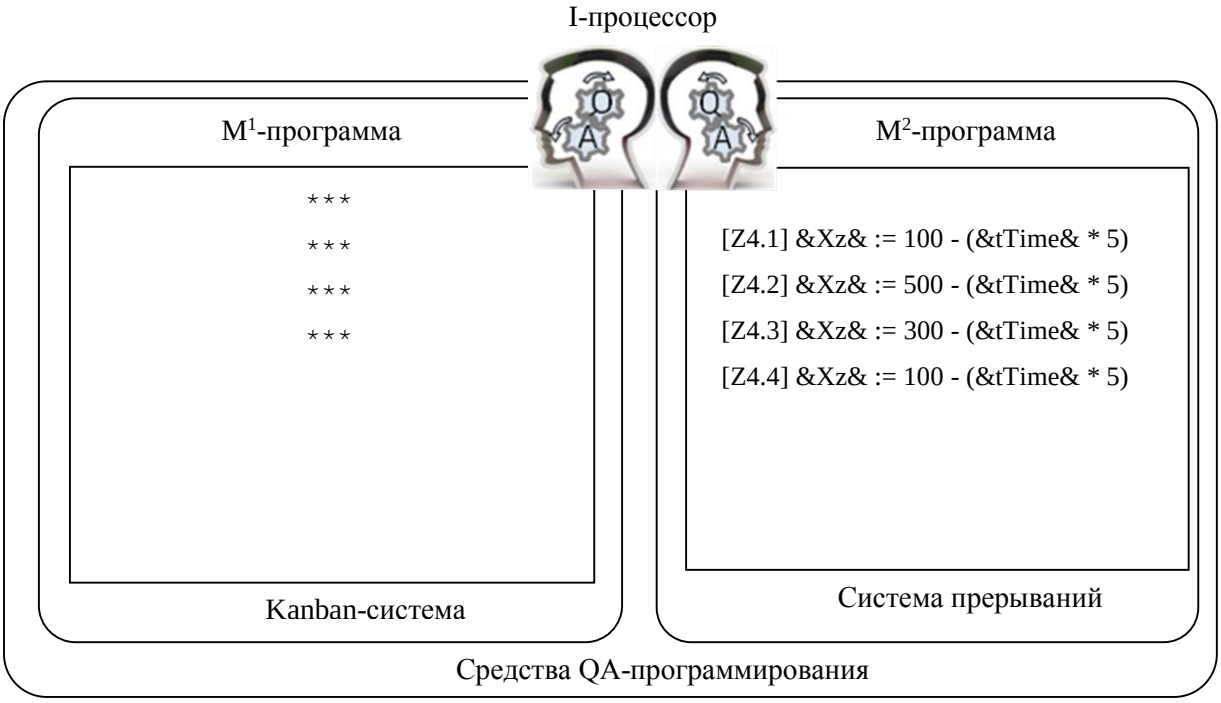


Рис. 4.18. Программирование параллелизма в задачах проектировщика

Когда единицы выполнения текущей задачи оказывались исчерпанными, её численное значение приоритета устанавливалось в 0, тем самым, задача оказывалась в конце очереди, а приоритеты остальных задач увеличивались на значение 100. Таким образом, происходило циклическое переключение между задачами. Вычисление этих действие происходило с использованием демона,

реализующего следующий код:

```
    &dbqaid& := QA_GetQAId(&current_project&, "БД_KANBAN")
    &dbid& := OpenDB(&dbqaid&)
    &query& := "SELECT id, Условие, Приоритет, Объемность FROM
Ассоциации WHERE id_колонки = "
    &query& := STRCAT_INT(&query&, &current_col&)
    &query& := STRCAT(&query&, "ORDER BY Приоритет")

    &res& := DB_QUERY(&dbid&, &query&)
    &vol& := DB_GETCELLINT(&res&, 0, 3)
    &Condition& := DB_GETCELLSTRING(&res&, &pos&, 1)
    ExecuteString(&Condition&)
    IF &Xz& >= 0 THEN GOTO &L1&

    &tId& := DB_GETCELLINT(&res&, 0, 0)
    &query& := "UPDATE Ассоциации SET Приоритет = 0 WHERE
id_задачи = "
    &query& := STRCAT_INT(&query&, &tId&)
    DB_QUERY(&dbid&, &query&)
    &tTime& := 0
    &pos& := 1
    &max& := DB_GETROWCOUNT(&res&)
    LABEL &lc&
        &tId& := DB_GETCELLINT(&res&, 0, pos)
        &tPrior& := DB_GETCELLINT(&res&, 0, 2) + 100
        &query& := "UPDATE Ассоциации SET Приоритет = ";
        &query& := STRCAT_INT(&query&, &tPrior&)
        &query& := STRCAT(&query&, "WHERE id_задачи = ")
        &query& := STRCAT_INT(&query&, &tId&)
        DB_QUERY(&dbid&, &query&)
        &pos& := &pos& + 1
        IF &pos& < &max& THEN GOTO &Lc&
    GOTO &LF&
    LABEL &L1&
    &tTime& := &tTime& + 1
    LABEL &LF&
```

Для оценивания трудоемкости задач использовался покер планирования. На следующем рисунке представлена картотечная визуализация выполненных задач:

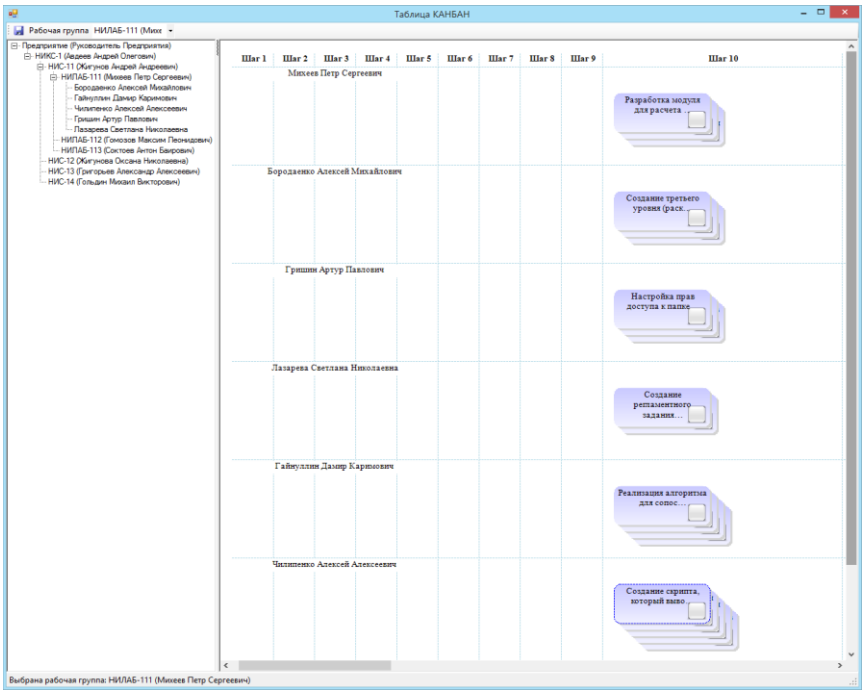


Рис. 4.19. Картотечная визуализация задач спринта

В процессе выполнения спринта была построена диаграмма выгорания, представленная на следующем рисунке:

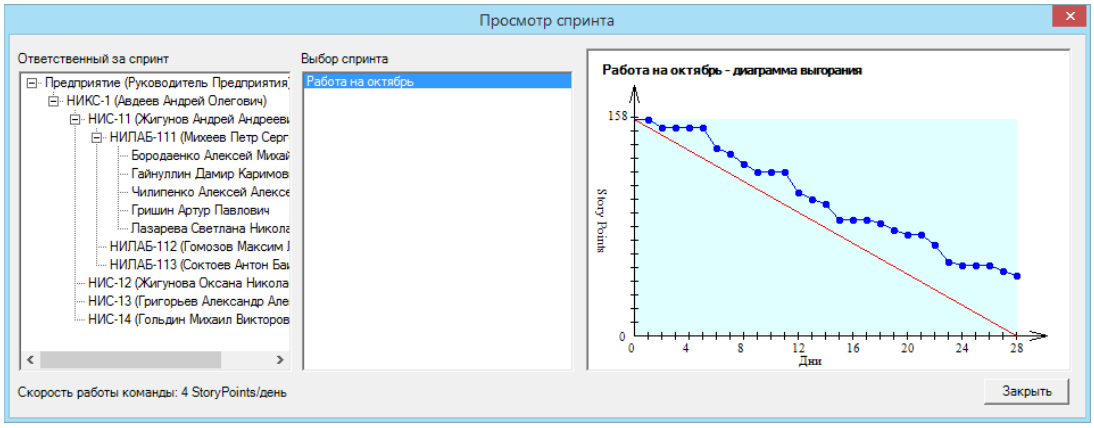


Рис. 4.20. Выполнение первого спринта

По результатам выполнения данного спринта были вычислены следующие метрики:

Таблица 4.15. Метрики первого спринта

Метрики Kanban	**Метрики Scrum**
• Cycle Time: 8,4 (сут); • WIP: 4,2 (задач); • Lead Time: 14 (сут); • Wasted Time: 5,6 (сут); • Effectiveness: 60%;	• TV: 4,07 (StoryPoints/день) – скорость работы команды в StoryPoints; • TD (Оценка планирования). 3 балла; • x = 0,09 (баллов за оставшийся StoryPoint).

• Throughput: 0,7 (задач/сут).	

Как видно из диаграммы выгорания, представленной на рисунке 4.20, сотрудники не успели выполнить в срок запланированные задачи. Руководитель команды оценил планирование данного спринта на 3 балла по 5-балльной шкале.

После выполнения второго спринта была получена следующая диаграмма выгорания:

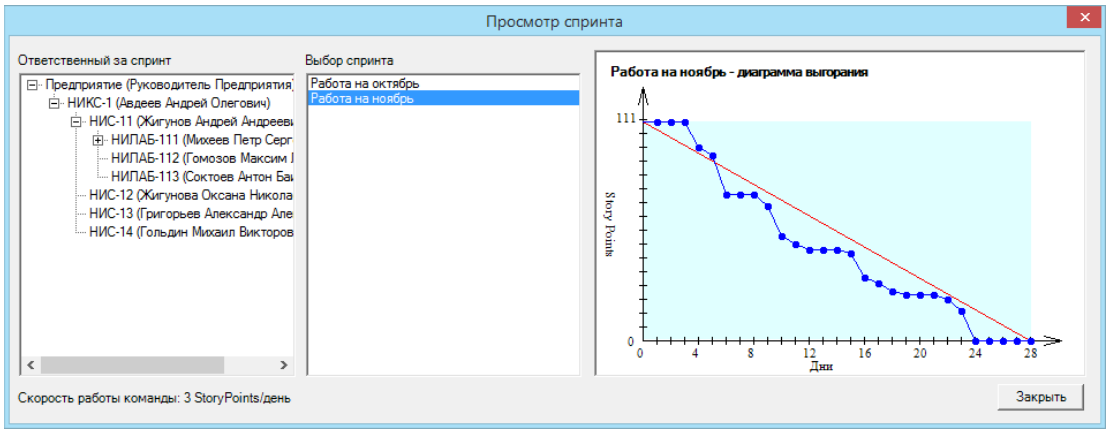


Рис. 4.21. Выполнение второго спринта

Из полученной диаграммы видно, что команда справилась с задачами спринта раньше запланированного времени. Групповые Scrum-метрики выполнения данного спринта следующие:

- TV: 4,625;
- TD: была оценена руководителем группы на 4 балла;
- x = 0,25 (баллов за неистраченный день спринта).

Для третьего спринта диаграмма выгорания получилась следующей:

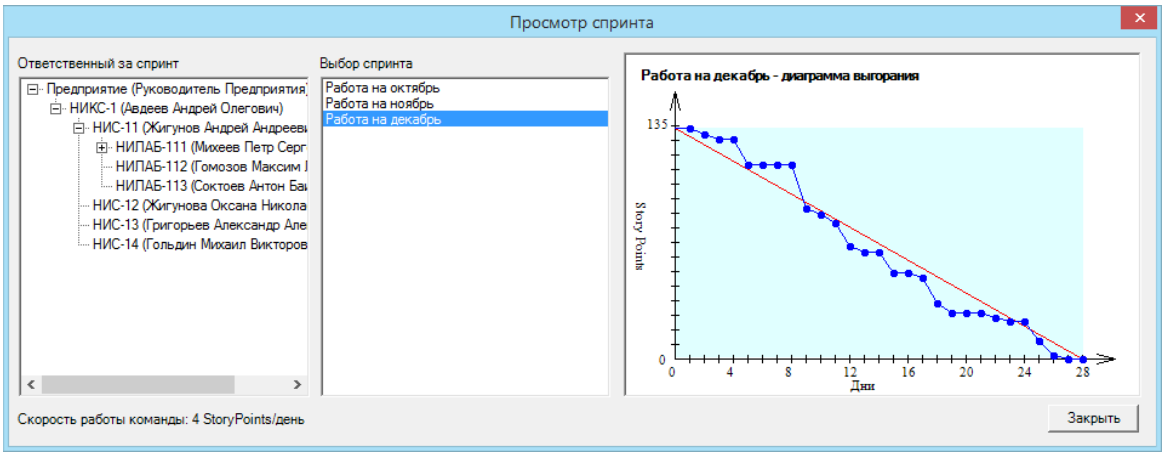


Рис. 4.22. Выполнение третьего спринта

Поскольку коэффициенты x и y для метрики оценки планирования TD были получены в результате выполнения предыдущих спринтов, эта метрика стала доступной для автоматического вычисления. Далее представлены метрики выполнения данного спринта:

- TV: 4,82;
- TD: 4,75.

Команда разработчиков согласилась с оценкой TD, выставленной автоматически.

На создание поручений руководителем проектной группы было затрачено 2 часа времени. На планирование каждого спринта командой разработчиков было затрачено по 1,5 часа времени, на каждую ежесуточную встречу было затрачено 15 минут. Руководитель проектной группы оценил данные временные затраты как приемлемые.

Из выполнения данного эксперимента были сделаны следующие выводы:

1. Применение комплекса средств ПКУ позволило поэтапно рационализировать планирование проектной работы, что подтверждается растущей оценкой TD. Следовательно, **гипотеза H1 подтверждена**;
2. Эксперимент показал возможность программирования как параллелизма в потоке проектных работ, так и очереди задач

проектировщика в условиях псевдопараллельного выполнения задач;

3. В процессе выполнения трех спринтов команда показала рост скорости выполнения задач в StoryPoints. Это можно объяснить тем, что участники команды в процессе работы над проектом с каждым этапом глубже понимали особенности задач проекта;
4. Временные затраты на пользование средствами ПКУ оказались приемлемыми для данной группы проектировщиков.

Описание проведения экспериментов на остальных проектах представлено в приложении 2.

В результате выполнения экспериментальных исследований были сделаны следующие выводы:

1. Использование средств ПКУ позволяет рационализировать оперативное планирование этапа проектной работы;
2. Включение ПКУ в процесс проектирования не приводит к недопустимым для проектировщика временным затратам, связанным с использованием программных средств ПКУ;
3. На среднюю скорость работы команды влияют реализовавшиеся в процессе выполнения этапа проектной работы риски, что дает возможность учитывать их при оперативном планировании дальнейших этапов проектной работы;
4. Разработанные средства ПКУ можно использовать при выполнении персональной работы в целях самоконтроля и приоритетизации решаемых задач;
5. Разработанные средства ПКУ можно использовать и в альтернативных целях, таких, как вычисление метрик в целях сравнения результатов выполнения тестовых заданий;
6. Разработанные программные средства протестированы.

4.4. Настройка средств ПКУ на использование различных методологий проектного управления

Включение в картотечное управление механизмов псевдо кодовго программирования позволяет не только реализовать рациональное распараллеливание коллективных и персональных действитий проектировщиков, но и настроить представленные выше средства на реализацию либо технологии Kanban, либо Scrum или Scrum-ban.

Выбор подходящей технологии управления зависит от специфики разрабатываемого проекта, а значит работы по настройке придётся выполнять исполнителям проекта, разумеется освоившим программирование в среде WIQA и служебные задачи ПКУ, представленные выше. Отметим, что служебные задачи (как задачи используемой технологии) всраиваются в дерево задач проекта.

Одним из важнейших отличий между механизмами управления Kanban, Scrum и Scrum-ban является работа с очередями задач, которую несложно настроить (запрограммировать на псевдо коде) на любую из этих технологий или их модификацию.

Для настройки средств ПКУ на применение Kanban-методологии, необходимо выполнить адаптацию в соответствии со следующей методикой:

1. Установить количество колонок в Kanban-таблице соответствующим требованиям. К примеру, при использовании модели водопада работа выполняется в течение семи фаз.
2. Для каждой колонки задать название, соответствующее фазе проектирования.
3. Установить лимиты нахождения количества задач на каждой фазе
4. Запрограммировать функционирование задач в очередях в соответствии с особенностями Kanban-подхода.

Теперь рассмотрим функционирование очередей в случае Kanban-подхода.

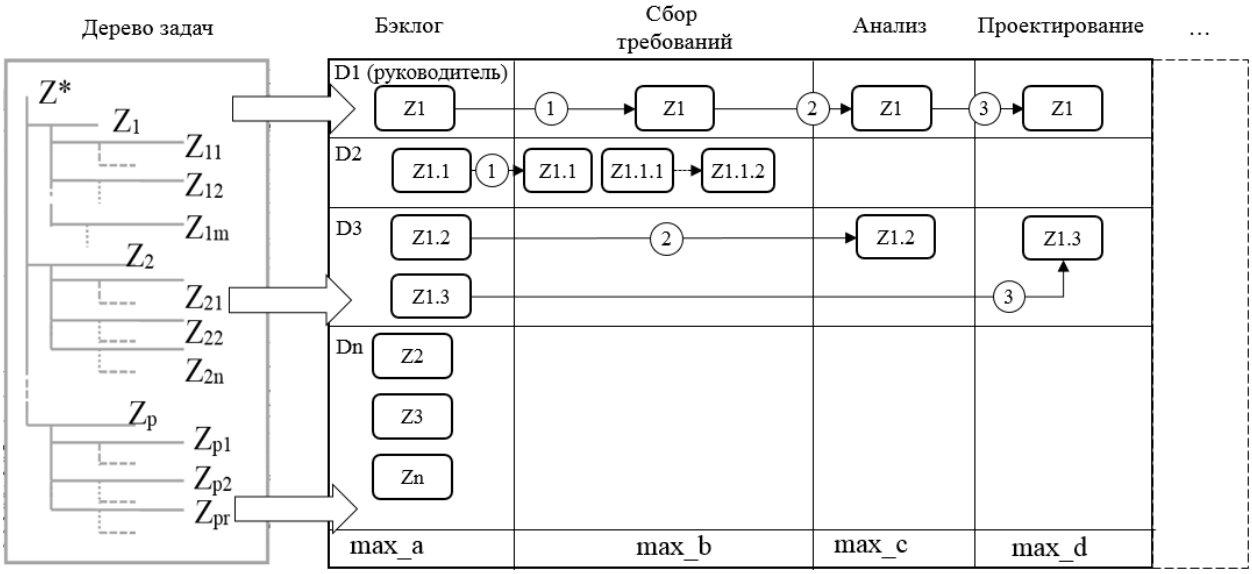


Рис. 4.23. Kanban в ПКУ

На рисунке 4.23 стрелками изображены переносы карточек из столбца в столбец в соответствии с выполняемой фазой проектирования. Для задачи требуется запрограммировать ряд условий, позволяющих перенести её карточку в следующий столбец при завершении фазы проектирования. При завершении работы над задачей карточка оказывается в последнем столбце. Эти условия могут различаться в зависимости от текущего положения карточки задач, в общем случае программа выглядит следующим образом:

```
IF &Z1&.Столбец == X THEN //Если карточка находится в столбце X
    IF &Z1.1&.Выполнено == 1 && &Z1.2&.Выполнено == 1 &&
&Z1.n&.Выполнено == 1 THEN
BEGIN
    &Z1&.Столбец := Y //Столбец может быть как следующим, в этом
случае можно увеличить это значение на единицу, так и другим, в
зависимости от особенностей задачи
    //Проверка лимита
    IF &Columns[Y]&.&Limit& < &Columns[Y]&.&Count& + &NSubTasks& THEN
    BEGIN
    //Подзадачи следует помещать в очереди с установкой условий
    //их выполнения и приоритетов
        QUEUE &Z1.A&, &Da& // Добавление подзадач, подлежащих
        QUEUE &Z1.B&, &Db& // выполнению на данном этапе,
     . . .                 // в очереди сотрудников
        QUEUE &Z1.M&, &Dq&
        ToColumn &Z1.A&, Y //
```

```
        ToColumn &Z1.M&, Y //Размещение подзадач в столбцах Kanban
     . . .                 //
        ToColumn &Z1.M&, Y //
    END
END
IF &Z1&.Столбец == Y THEN //Если карточка находится в столбце Y
. . .                     //И так далее, условия для каждого столбца
```

Теперь рассмотрим настройку разработанных инструментальных средств в случае использования методологии Scrum. В этом случае количество задач, выполняемых на этапе проектного процесса, именуемого спринтом, не является ограниченным, как в Kanban, вместо этого при планировании спринта учитывается трудоемкость выбранных задач.

Размещение карточек на Scrum-доске осуществляется следующим образом:

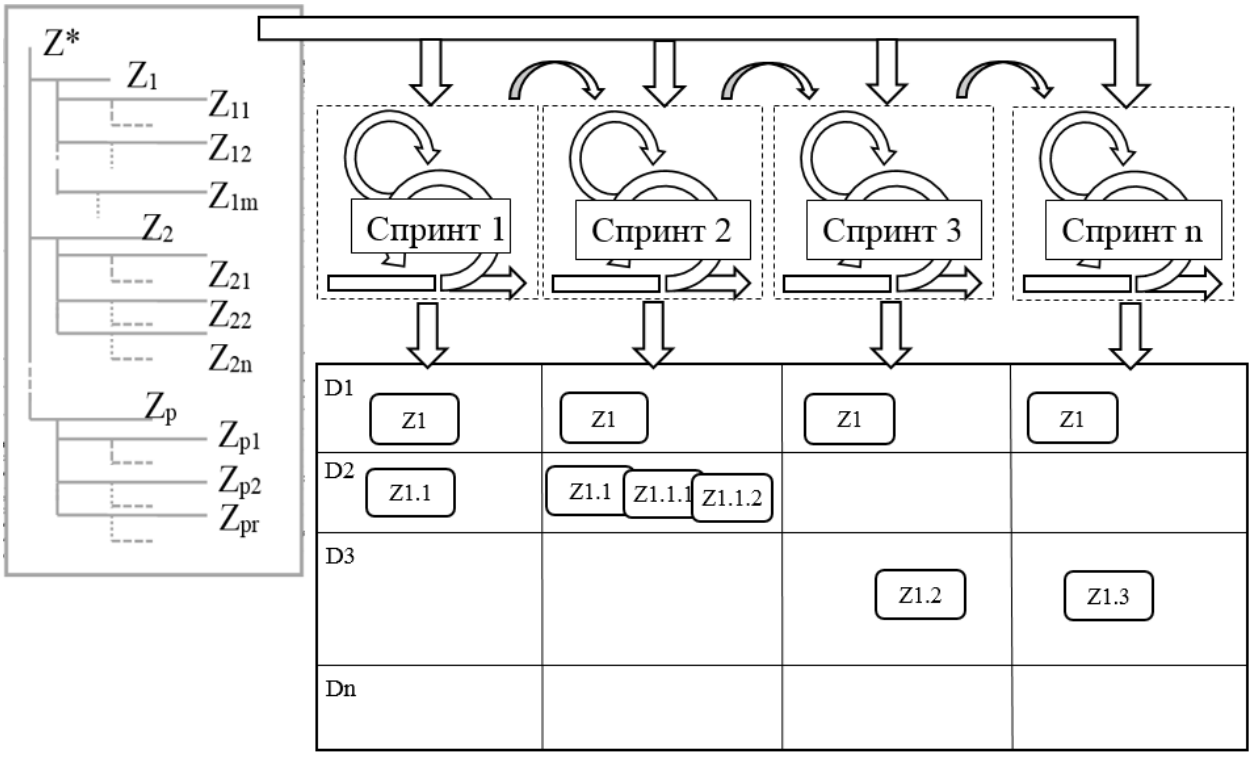


Рис. 4.24. Scrum в ПКУ

В Scrum-доске заполнение соответствующего столбца происходит при формировании нового спринта. Для настройки средств ПКУ на применение Scrum-методологии, необходимо выполнить адаптацию в соответствии со следующей методикой:

1. Установить количество колонок в таблице в соответствии запланированным количеством спринтов;
2. Установить отсутствие лимитов по колонкам;
3. Запрограммировать формирование очередей из подзадач на событие помещения задачи в очередь;
4. Запретить перемещение задачи из столбца в столбец – появление задачи в новых столбцах происходит при формировании нового спринта.

Программирование обработчика события помещения задачи в очередь осуществляется в общем виде следующим образом:

```
    QUEUE &Z1.A&, &Da& // Добавление подзадач, подлежащих
    QUEUE &Z1.B&, &Db& // выполнению на данном этапе,
. . .                  // в очереди сотрудников
    QUEUE &Z1.M&, &Dq&
    //Помещение карточек в колонку
    ToColumn &Z1.A&, &NSprint& //
    ToColumn &Z1.M&, &NSprint& //NSprint в данном случае -
. . .                          //порядковый номер спринта
    ToColumn &Z1.M&, &NSprint& //
```

ScrumBan-методология отличается совмещением как Kanban-, так и Scrum-подходов, при этом выполняется установка лимитов задач на столбец, кроме того, устанавливается небольшая длина спринта порядка 14 дней или меньше.

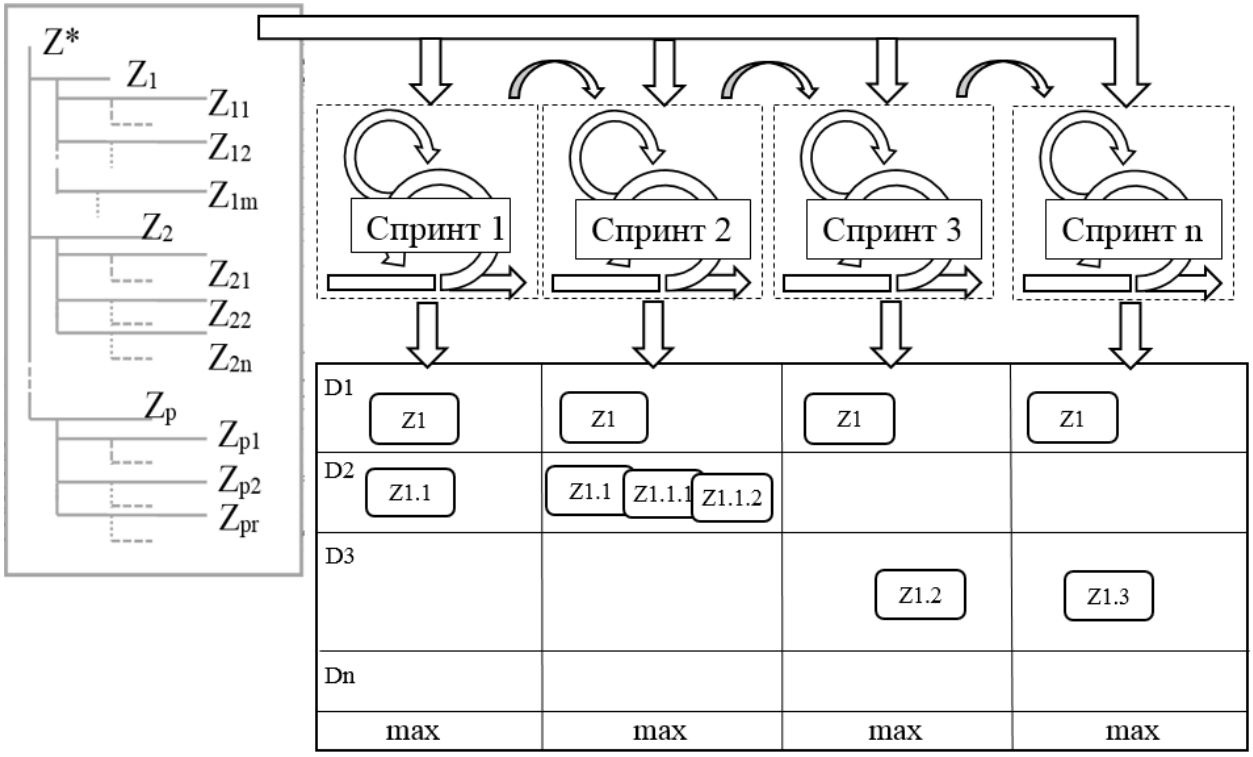


Рис. 4.25. ScrumBan в ПКУ

Для настройки средств ПКУ на применение Scrum-методологии, необходимо выполнить адаптацию в соответствии со следующей методикой:

1. Установить количество колонок в таблице в соответствии запланированным количеством спринтов;
2. Установить отсутствие лимитов по колонкам;
3. Запрограммировать формирование очередей из подзадач на событие помещения задачи в очередь;
4. Запретить перемещение задачи из столбца в столбец – появление задачи в новых столбцах происходит при формировании нового спринта.

Программирование обработчика события помещения задачи в очередь осуществляется в общем виде следующим образом:

```
    IF  &Columns[&NSprint&]&.&Limit&  <  &Columns[&NSprint&]&.&Count&  +
&NSubTasks& THEN
BEGIN
    QUEUE &Z1.A&, &Da& // Добавление подзадач, подлежащих
    QUEUE &Z1.B&, &Db& // выполнению на данном этапе,
 . . .                 // в очереди сотрудников
    QUEUE &Z1.M&, &Dq&
```

```
    //Помещение карточек в колонку
    ToColumn &Z1.A&, &NSprint& //
    ToColumn &Z1.M&, &NSprint& //NSprint в данном случае -
 . . .                         //порядковый номер спринта
    ToColumn &Z1.M&, &NSprint& //

END
```

Выводы по четвертой главе

1. Разработанные средства программно-картотечного управления реализованы полностью в среде WIQA как задача управления потоками работ, которую можно добавить в любой проект;
2. Существуют две версии WIQA – WIQA.Net, имеющая клиент-серверную архитектуру и OwnWIQA, предполагающая работу с ней одного проектировщика;
3. Эти версии WIQA имеют различия в организационной структуре, которые пришлось учесть при реализации средств ПКУ;
4. В версии OwnWIQA потребовалась реализация механизма синхронизации базы данных OwnWIQA с базами данных других проектировщиков;
5. Экспериментальное исследование подтвердило возможность применения разработанных средств для рационализации проектной деятельности;
6. В процессе выполнения экспериментов на языке LWIQA были запрограммированы как параллельные потоки работ проекта, так и механизмы распараллеливания выполнения задач, что подтвердило возможность применения данного языка для программного управления потоками проектных работ;
7. Экспериментальное исследование подтвердило возможность программного управления параллельным выполнением потока проектных работ группой проектировщиков;
8. Экспериментальное исследование подтвердило возможность

управления очередями выполнения задач как на основе разблокирования только одной из подлежащих параллельному выполнению задач в момент времени, так и на основе динамически вычисляемых приоритетов задач.

9. Экспериментальное исследование также подтвердило возможность применения разработанных средств для сравнительной оценки производительности выполняющих тестовое задание людей;
10. Нагрузочное тестирование показало устойчивую работу системы и приемлемую скорость работы программных средств при размещении в ней 1734 карточек задач.

Заключение

Подводя обобщающий итог диссертационному исследованию и практическим разработкам, реализованным на базе результатов исследований, можно утверждать следующее:

Цель исследований, направленная на совершенствование процессов управления потоками работ, способствующее предотвращению ошибок, обусловленных человеческим фактором, и обеспечивающее сокращение непроизводительных затрат времени в оперативной индивидуальной и коллективной работе проектировщиков за счет включения в совокупность средств управления проектной деятельностью дополнительных программируемых составляющих, **достигнута**.

Получены научные результаты:

1. **Программно-картотечная модель** гибкого управления в проектировании АС, определяющая как управление коллективной, так и персональной работой проектировщика. Эта модель обеспечивает рациональное планирование бэклога этапа проектной работы, а также обеспечивает механизмы параллельного и псевдопараллельного выполнения задач;

2. Подмножество **псевдокодового алгоритмического языка** программирования потоков работ, обеспечивающее оперативное выполнение проектных работ в соответствии с заданной программой;

3. **Совокупность методик программно-картотечного управления**, обеспечивающих формирование бэклога этапа проектной работы, установку ассоциаций между задачей и исполнителем, управление поручениями, управление очередями задач, вычисление метрических характеристик проектной работы как во время выполнения её этапа, так и по его окончанию;

4. **Библиотека псевдокодовых программ и программных моделей паттернов потоков работ** для использования их в программно-картотечном управление коллективным проектированием АС.

Практическую ценность работы составляет разработанный набор средств, обеспечивающий реализацию координирования потоков работ в проектировании АС, а также - библиотека шаблонов потоков работ и комплекс методик по координированию потоков работ в процессе проектирования АС

Литература

1. Акофф, Р.О целеустремленных системах / Р. Акофф, Ф.Эмери. –М.: Сов. радио, 1974. –272 с.
2. Гаврилова, Т.А., Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф.Хорошевский. –СПб: Питер, 2000. –384 с.
3. ГОСТ Р 52806–2007. Менеджмент рисков проектов. Общие положения [Электронный ресурс] URL: http://expert.gost.ru/ME/DOC/TXT_GOST_R_52806-2007.pdf (Дата обращения 15.01.2015)
4. ГОСТ Р 53892-2010. Руководство по оценке компетентности менеджеров проектов. Области компетентности и критерии профессионального соответствия [Электронный ресурс] URL: http://protect.gost.ru/ v.aspx? control=8&page=0&id=168699 (Дата обращения 15.01.2015)
5. ГОСТ Р 54869-2011 Проектный менеджмент. Требования к управлению проектом [Электронный ресурс] Режим доступа: http://docs.cntd.ru/ document/1200089604
6. ГОСТ Р ИСО 10006–2005. Системы менеджмента качества. Руководство

по менеджменту качества при проектировании [Электронный ресурс] URL: http://ohranatruda.ru/ot_biblio/normativ/data_normativ/46/46262/index.php (Дата обращения 15.01.2015)
7. Интерпретатор псевдокодовых программ в вопросно-ответной среде / Ю. А. Лапшов, С. А. Заболотнов, П. И. Соснин // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2010. – 531 с. С. 361-368
8. Кролл, П. RationalUnifiedProcess –это легко: Руководство по RUP для практиков / П. Кролл, Ф. Крачтен. –М.: КУДИЦ-Образ, 2004. –427 с.
9. Лапшов Ю. А. Псевдокодовое программирование управления прерываниями человека / Ю. А. Лапшов // Информатика и вычислительная техника: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск: УлГТУ, 2011 – 656 с. С.353-356
10. Лапшов Ю. А. Реализация компилятора функций вопросно-ответных псевдокодовых программ / Ю. А. Лапшов // Информатика и вычислительная техника: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск: УлГТУ, 2011 – 656 с. С.363-367
11. Лапшов Ю. А. Человеческие прерывания в потоках работ / Ю. А. Лапшов // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2010. – 531 с. С. 358-360
12. Лапшов Ю.А. Особенности реализации управления прерываниями в среде моделирования WIQA.Net / Ю.А. Лапшов // Информатика и вычислительная техника : сборник научных трудов / под ред. В.Н. Негоды. – Ульяновск, УлГТУ, 2010. – 677 с.316-317
13. Лапшов Ю.А. Приоритеты задач в коллективной среде проектирования / Ю. А. Лапшов // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2011. – 531 с. С.272-276
14. Лапшов Ю.А. Процесс работы в коллективной среде проектирования в условиях прерываний как система массового обслуживания / Ю.А. Лапшов // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2011. – 531 с. С.277-282
15. Лапшов Ю.А. Таксономия человеческих прерываний в корпоративных средах моделирования / Ю. А. Лапшов // Информатика и вычислительная техника : сборник научных трудов / под ред. В.Н. Негоды. – Ульяновск, УлГТУ, 2010. – 677 с.318-320
16. Лапшов Ю.А. Упаковка табличных данных в средах псевдокодового программирования / Ю.А. Лапшов // Информатика и вычислительная техника: сборник научных трудов Т.2/ под ред. Н. Н. Войта. – Ульяновск: УлГТУ, 2012 – 408 с. С.8-12

17. Лапшов Ю.А. Управление человеческими прерываниями в корпоративных средах моделирования / Ю. А. Лапшов // Информатика и вычислительная техника : сборник научных трудов / под ред. В.Н. Негоды. – Ульяновск, УлГТУ, 2010. – 677 с.321-326
18. Лапшов Ю.А., Маклаев В.А. Библиотека паттернов потоков работ / Ю.А. Лапшов, В.А. Маклаев // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2011. – 531 с. С.272-277
19. Лапшов Ю.А., Маклаев В.А., Ромодин М.Ю. Средства псевдокодового моделирования и программирования проектных решений с использованием баз данных / Ю.А. Лапшов, В.А. Маклаев, М.Ю. Ромодин // Информатика и вычислительная техника: сборник научных трудов Т.1/ под ред. Н. Н. Войта. – Ульяновск: УлГТУ, 2012 – 448 с. С.427-439
20. Лапшов Ю.А., Маклаев В.А., Соснин П.И. Объектно-ориентированное вопросно-ответное псевдокодовое программирование / Ю. А. Лапшов, В. А. Маклаев, П. И. Соснин // Информатика и вычислительная техника: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск: УлГТУ, 2011 – 656 с. С. 357-362
21. Лапшов Ю.А., Маклаев В.А., Соснин П.И. Учет сложности в управлении потоками работ в проектировании автоматизированных систем / Ю.А. Лапшов, В.А. Маклаев, П.И. Соснин // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2012. – 463 с. С. 15-23
22. Лапшов Ю.А., Ромодин М.Ю. Контроль ссылочной целостности в псевдо-кодовых базах данных / Ю.А. Лапшов, М.Ю. Ромодин // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск : УлГТУ, 2012. – 463 с. С. 266-271
23. Ларин, С.Н. Методологический базис конструкторско-технологических решений с позиций зрелости производственных процессов. // Автоматизация процессов управления. – № 4(26). –2011 г. – С. 55-65.
24. Ларичев, О.И. Теория и методы принятия решений / О.И. Ларичев. –М.: Логос, 2000. –296 с.
25. Лебедев В.Н. Введение в системы программирования/ В.Н. Лебедев. –М. : Статистика, 1975. –327 с.
26. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл. –М.: «Вильямс», 2002. –448 с.
27. Липаев В.В. Системное проектирование сложных программных средств для информационных систем / В.В. Липаев // М.: Синтег, 2002. –268 с.
28. Маклаев, В.А. Средства электронной коммуникации в корпоративной среде автоматизированного проектирования / В.А.Маклаев //Известия высших учебных заведений. Северо-Кавказский регион. Технические

науки. Приложение №4 – 2006. – С.12-16.
29. Маклаев, В.А. Инструментально-технологическая среда формирования и использования опыта проектной организации / В.А. Маклаев // Автоматизация процессов управления. – № 1(23) – 2010. – С.8-18.
30. Маклаев, В.А. Нормативы профессиональной зрелости процессов разработки автоматизированных систем / В.А. Маклаев, А. А. Перцев // Ульяновск : УлГТУ – 2012. – 343 с.
31. Марка, Д.А., Методология структурного анализа и проектирования SADT / Д.А. Марка. –М.: Метатехнология, 1993. –546 с.
32. Норенков, И.П. Основы автоматизированного проектирования: учебник для вузов/ И.П. Норенков. –М.: МГТУ им. Н.Э.Баумана, 2002. –336 с.
33. Поспелов, Д.А. Логико-лингвистические модели в системах управления / Д.А. Поспелов –М.: Энергоатомиздат, 1981. –231 с.
34. Поспелов, Д.А. Моделирование рассуждений. Опыт анализа мыслительных актов / Д.А. Поспелов–М.: Радио и связь, 1989.
35. Псевдокодовое программирование в концептуальном проектировании баз данных / М.Ю. Ромодин, Ю.А. Лапшов, П.И. Соснин // Научно-технический журнал «Автоматизация процессов управления». №2(32) 2013. Ульяновск: ФНПЦ ОАО НПО «Марс»
36. Рассел, С. Искусственный интеллект: современный подход: [пер. с англ.] / С. Рассел. –М.:ИД Вильямс, 2006. –408 с.
37. Робсон, М., Практическое руководство по реинжинирингу бизнес-процессов / Пер. с англ. под ред. Н. Д.Эриашвили. – М.: Аудит, ЮНИТИ, 1997. – 224 с.
38. Саати, Т. Принятие решений. Метод анализа иерархий. / Т. Саати. –М.: Радио и связь. 1993. –347 с.
39. Соммервилл, И. Инженерия программного обеспечения / И. Соммервилл. –М.: Вильямс, 2002. –624 с.
40. Соснин П.И. Инструментальные средства программного управления потоками работ в проектировании автоматизированных систем систем / П.И. Соснин, Ю. А. Лапшов, Маклаев В.А. // Научно-технический журнал «Автоматизация процессов управления». №4(30) 2012. Ульяновск: ФНПЦ ОАО НПО «Марс».
41. Соснин П.И. Псевдо-кодовое программное управление потоками работ в проектировании автоматизированных систем / П.И. Соснин, Ю. А. Лапшов, Маклаев В.А. // Научно-технический журнал «Автоматизация процессов управления». №3(29) 2012. Ульяновск: ФНПЦ ОАО НПО «Марс».
42. Соснин, П. И. Концептуальное моделирование компьютеризованных систем: учебное пособие/ П.И. Соснин. – Ульяновск: УлГТУ, 2008. – 198 с.
43. Соснин, П.И. Архитектурное моделирование автоматизированных систем: учебное пособие/ П.И. Соснин. –Ульяновск: УлГТУ, 2007. –146 с.

44. Соснин, П.И. Вопросно-ответное моделирование в разработке автоматизированных систем /П.И.Соснин. –Ульяновск:УлГТУ, 2007.– 333 с.
45. Соснин, П.И. Инструментальные средства для спецификации концептуализаций в проектировании автоматизированных систем/П.И.Соснин.// Онтология проектирования. № 2, 2012.
46. Соснин, П.И. Проблемно-ориентированные диалоговые среды/ П.И.Соснин // Саратов: СГУ,1995. – 100 с.
47. Соснин, П.И. Человеко-компьютерная диалогика / П.И. Соснин – Ульяновск: УлГТУ, 2000. –286 с.
48. Хантер, Р. Основные концепции компиляторов / Р. Хантер. – М.: «Вильямс», 2002.
49. Adamczyk P. D. A method, system, and tools for intelligent interruption management / P.D. Adamczyk, S.T. Iqbal, B.P. Bailey // Proceedings of the 4th International Workshop on Task Models and Diagrams (TAMODIA'05). – New York : ACM Press, 2005. – C. 123-126
50. Adamczyk P. D., Bailey B. P. If not now, when?: The effects of interruption at different moments within task execution / P.D. Adamczyk, B.P. Bailey // Human Factors in Computing Systems: Proceedings of CHI'04. – New York : ACM Press, 2004. – C. 271-278
51. Adler R. F., Benbunan-Fich R. Juggling on a high wire: Multitasking effects on performance / R.F. Adler, R. Benbunan-Fich // International Journal of Human-Computer Studies, 70 (2), 2012 – C. 156-168 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Adler-IntJHumComputStud12.pdf (Дата обращения: 10.08.2014)
52. Adler R. F., Benbunan-Fich R. Self-interruptions in discretionary multitasking / R.F. Adler, R. Benbunan-Fich // Computers in Human Behavior, 29 (4), 2013 – C. 1441-1449 [Электронный ресурс] Режим доступа: http://www. interruptions.net/literature/ Adler-ComputHumBehav13.pdf (Дата обращения: 11.08.2014)
53. Adler R. F., Benbunan-Fich R. The effects of task difficulty and multitasking on performance / R.F. Adler, R. Benbunan-Fich // Interacting with Computers, 2014 [Электронный ресурс] Режим доступа: http://www.interruptions.net /literature/Adler-InteractComput14.pdf (Дата обращения: 15.08.2014)
54. Altmann E. M. Momentary interruptions can derail the train of thought / E.M. Altmann, J.G. Trafton, D.Z. Hambrick // Momentary interruptions can derail the train of thought, Journal of Experimental Psychology: General, 143 (1). – C. 215-226 [Электронный ресурс] Режим доступа: http://www.interruptions.net /literature/Altmann-JExpPsycholGen14.pdf (Дата обращения: 18.09.2014)
55. Altmann E. M., Trafton J. G. Task interruption: Resumption lag and the role of cues / E.M. Altmann, J.G. Trafton // Proceedings of the 26th Annual Conference of the Cognitive Science Society, 2004 [Электронный ресурс] Режим доступа:

http://www.interruptions.net/literature/Altmann-CogSci04.pdf (Дата обращения: 18.09.2014)
56. Altmann E. M., Trafton J. G. Timecourse of recovery from task interruption: Data and a model / E.M. Altmann, J.G. Trafton // Psychonomic Bullletin & Review, 14 (6), 2007. – С. 1079-1084 [Электронный ресурс] Режим доступа: http://www.interruptions .net/literature/Altmann-PBR07.pdf (Дата обращения: 18.09.2014)
57. Anderson D. J, Concas J., Lunesu M. I., Marchesi M., Zhang H. A Comparative Study of Scrum and Kanban Approaches on a Real Case Study Using Simulation / D. J. Anderson, J. Concas, M. I. Lunesu, M. Marchesi, H. Zhang // C. Wohlin (Ed.): XP 2012, Lecture Notes in Business Information Processing, 2012, pp. 123–137.
58. Andrews A. E. The effect of alert type to an interruption on primary task resumption / A.E. Andrews, R.M. Ratwani, J.G. Trafton // Proceedings of the Human Factors and Ergonomics Society 53rd Annual Meeting (HFES'09), Santa Monica: Human Factors and Ergonomics Society, 2009 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Andrews-HFES09.pdf (Дата обращения: 15.09.2014)
59. Andrews P. Vying for your attention: Interruption management / P. Andrews // Executive Technology Report, 7. – С. 1-8 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Andrews-ETR04-G510-3939-00.pdf (Дата обращения: 16.09.2014)
60. Bangerter A. Managing third-party interruptions in conversations: Effects of duration and conversational role / Bangerter A., Chevalley E., Derouwaux S. // Journal of Language and Social Psychology, 29 (2), 2010. – С. 235-244 [Электронный ресурс] Режим доступа: http://www.interruptions.net/ literature/Bangerter-JLangSocPsychol10 .pdf (Дата обращения: 16.09.2014)
61. Barrett R. Usable autonomic computing systems: The administrator's perspective / P.P. Maglio, E. Kandogan, J. Bailey // Proceedings of International Conference on Autonomic Computing (ICAC 2004), Los Alamitos: IEEE Computer Society, – С. 18-25 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Barrett-ICAC04-01301342.pdf (Дата обращения: 21.10.2014)
62. Beeftink F.The effect of interruptions and breaks on insight and impasses: Do you need a break right now? / F. Beeftink, W. van Eerde, C.G. Rutte // Creativity Research Journal, 20 (4), 2008. – С. 358-364 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Beeftink-CreativResJ08.pdf (Дата обращения: 12.09.2014)
63. Benbunan-Fich R. Measuring multitasking behavior with activity-based metrics / R.F. Adler, T. Mavlanova // ACM Transactions on Computer-Human Interaction, 18 (2), Article 7, 2011 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Benbunan-Fich-TOCHI11.pdf (Дата

обращения: 18.11.2014)
64. Business Intelligence Maturity Models: An Overview/G.Lahrmann,F. Marx, R.Winter , F.Wortmann. - www.alexandria.unisg.ch/ Publikationen / 72444/L-en.
65. Charette, R.N. Why software falls/R.N. Charette//IEEE Spectrum, Vol. 42(9), 2005, pp. 36-43.
66. CMMI® for Development, Version 1.3 : CMMI Product Team. — 2010 — November [Электронный ресурс]. URL: http://www.sei.cmu.edu/reports/ 10tr033.pdf (Дата обращения 15.01.2015)
67. Cohn M. Agile Estimating and Planning, / M. Cohn // Prentice Hall PTR, Upper Saddle River, NJ, 2006.
68. Competency Maturity Model Wheel/M.Von Rosing, S.Moshiri, K.Gräslund, A.Rosenberg.- http:// www.valueteam.biz/downloads/ model_cmm_wheel.pdf.
69. Cooke-Davies, T.J.The maturity of project management in different industries - An investigation into variations between project management models/T.J.Cooke-Davies, A.Arzymanow //International Journal of Project Management, Vol. 21(6), 2003 , pp. 471-479.
70. Coraggio L. Deleterious Effects of Intermittent Interruptions on the Task Performance of Knowledge Workers: A Laboratory Investigation / L. Coraggio // PhD Dissertation, University of Arizona, College of Business and Public Administration, 1990 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Coraggio-PhD.pdf (Дата обращения: 18.09.2014)
71. Curtis, B. People Capability Maturity Model (P-CMM) Version 2.0 / B.Curtis,B. Hefley, S.Miller // Technical Report CMU/SEI-2009-TR-003, 2009.
72. Dijkstra, E. Programming Considered as a Human Activity/E.Dijkstra. - http://hp.fciencias.unam.mx/~jloa/CC/dijkstra1i.html.
73. Essence – Kernel and Language for Software Engineering Methods : сайт SEMAT — 2012 — August 13 [Электронный ресурс] URL: http://semat.org/wp-content/ uploads/ 2012/ 02/ 12- 08-15.pdf (Дата обращения 15.01.2015)
74. Ghosh S. Enhance PMBOK® by Comparing it with P2M, ICB, PRINCE2, APM and Scrum Project Management Standards / Ghosh S., Forrest D., DiNetta T. et al. // PM World Today - Jan 2012 - Vol. 14 Issue 1, Special section - p1-77.
75. Gluck, K.A. Modeling human behavior with integrated cognitive architectures: Comparison, evaluation, and validation/K. A. Gluck, R.Pew -Mahwah, NJ: Erlbaum.
76. Held, M. W. Structured collaborative workflow design, Future Generation Computer Systems/M.Held,W. Blochinger.-Vol. 25(6), 2009, pp. 638-653.
77. Henninger S. Tool Support for Experience-based Software Development Methodologies/S.Henninger // Advances in Computers, Vol. 59, 2003, pp. 29-82.

78. Hirschman, L. Natural language question answering: the view from here/L.Hirschman, R.Gaizauskas// Natural Language Engineering, 2001, pp. 767-876.
79. Human-Computer Interaction: Overview on State of the Art / F.Karray, M.Alemzadeh, J. A. Saleh, M. N. Arab //Smart sensing and intelligent systems, Vol. 1(1), 2008, pp 138-159.
80. IEEE Std 1490-2011. IEEE Guide – Adoption of the Project Management Institute (PMI(R)) Standard A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) – Fourth Edition. [Электронный ресурс] URL: http://deltarobot.wikispaces.com/file/view/IEEE%20Std%201490.pdf (Дата обращения 15.01.2015)
81. Ikonen M., Kettunen P., Oza N., Abrahamsson P. Exploring the Sources of Waste in Kanban Software Development Projects / M. Ikonen, P. Kettunen, N. Oza, P. Abrahamsson //Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications, 2010. С. 376–381
82. ISO 21500:20102. Guidance on project Management. [Электронный ресурс] Режим доступа: http://www.vanharen.net/Samplefiles/9789087538095 SMPL.pdf (Дата обращения 12.11.2014)
83. ISO 9004:2009. Менеджмент в целях достижения устойчивого успеха организации. Подход на основе менеджмента качества [Электронный ресурс] Режим доступа: http://www.sstu.ru/ upload/medialibrary /e5c/iso_9004_2009_pqm_.pdf (Дата обращения: 14.12.2014)
84. Johnson P. Where’s the Theory for Software Engineering? / Johnson P., Ekstedt M., Jacobson I. // IEEE Software - September/October 2012 – P. 94-95.
85. Kieras D. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction/D.Kieras,D.E.Meyer // Human-Computer Interaction, Vol. 12, 1997, pp. 391-438.
86. Kroll P., Kruchten P. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP / P. Kroll and P. Kruchten // Boston: Addison Wesley, 2003.
87. Kroll P. The Rational Unified Process Made Easy: A Practitioners Guide to the RUP / P. Kroll, Ph.Kruchten- Addison-Wesley, 2003.
88. Kruchten P. The Rational Unified Process: An Introduction / P. Kruchten // Third ed. Boston: Addison Wesley, 2004.
89. Lapshov Y. A. Human interruptions management in corporate modeling environment WIQA.Net / Y. A. Lapshov // Interactive Systems and Technologies: the Problems of Human-Computer Interaction. Volume III. – Collection of scientific papers. – Ulyanovsk: UlSTU, 2009. – 468 p. с.229-234
90. Lapshov Y., Maklaev V. Implementation of Compiler for Functions of Question-and-Answer Pseudo-Code Programs / Y. Lapshov, V. Maklaev // Interactive Systems and Technologies: the Problems of Human-Computer Inter-action. Volume III. – Collection of scientific papers. – Ulyanovsk: UlSTU, 2011. – 435 p С.275-279

91. Lapshov Y., Maklaev V., Sosnin P. Object-Oriented Question-and-Answer Pseudo-Code Programming / Y. Lapshov, V. Maklaev, P. Sosnin // Interactive Systems and Technologies: the Problems of Human-Computer Inter-action. Volume III. – Col-lection of scientific papers. – Ulyanovsk: UlSTU, 2011. – 435 p C.248-252
92. Larman C., Basili V. A History of Iterative and Incremental Development / C. Larman, V. Basili // IEEE Computer, vol. 36, no. 6, June 2003. – C. 47-56
93. Larman C. Agile and Iterative Development: A Manager's Guide / C. Larman // Boston: Addison Wesley, 2004.
94. Mahnic V. Applying Kanban Principles to Software Development / V. Mahnic // Proceedings of the 16th International Conference on Information Technology for Practice, Ostrava, Czech Republic, October 10-11, 2013. C. 89-96
95. McFarlane D. C. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction / D. C. McFarlane // Human-Computer Interaction, 17 (1), 2002. – C. 63-139 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/McFarlane-HCI02_2.pdf (Дата обращения: 19.09.2014)
96. McFarlane D. C. Coordinating the interruption of people in human-computer interaction. / D.C. McFarlane // Proceedings of Human-Computer Interaction (INTERACT'99), Amsterdam: IOS Press, 1999. – C. 295-303 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/McFarlane-Interact99-Coordinating.pdf (Дата обращения: 18.09.2014)
97. McFarlane D. C. Interruption of People in Human-Computer Interaction / D.C. McFarlane // Doctoral Dissertation, George Washington University, Washington, 1998 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/ McFarlane-NRL-97.pdf (Дата обращения: 18.09.2014)
98. McFarlane D. C. Interruption of People in Human-Computer Interaction: A General Unifying Definition of Human Interruption and Taxonomy / D.C. McFarlane // NRL Formal Report, Washington: US Naval Research Laboratory, 1997 [Электронный ресурс] Режим доступа: http://www.interruptions.net/ literature/McFarlane-NRL-97.pdf (Дата обращения: 18.09.2014)
99. McFarlane D. C., Latorella K. A. The scope and importance of human interruption in human-computer interaction design / D.C. McFarlane, K.A. Latorella // Human-Computer Interaction, 17 (1), 2002. – C. 1-61 [Электронный ресурс] Режим доступа: http://www.interruptions.net/ literature/McFarlane-HCI02_1.pdf (Дата обращения: 25.09.2014)
100. Mettler T. Maturity Assessment Models: A Design Science Research Approach / T.Mettler // International Journal of Society Systems Science Vol. 3(1.2), 2011, pp. 81-98
101. Murray A. PRINCE2® in one thousand words // Axelos Global Best Practice — 2011 [Электронный ресурс] URL: http://www.best-management-

practice.com/gempdf/prince2_ in _one_thousand_words.pdf (Дата обращения 15.01.2015)
102. Nagappan N., Williams L., Vouk M. Towards a Metric Suite for Early Software Reliability Assessment / N. Nagappan, L. Williams, M. A. Vouk // International Symposium on Software Reliability Engineering Fast Abstract. – Denver, CO. – 2003
103. Organization project management maturity model (OPM3®) - Third Edition : Knowledge Foundation // Project Management Institute — 2003 [Электронный ресурс] http://strelaconsult.com/upload/page/files/4_ Organizational_Project_Management_Maturity_Model_%28OPM3%29.pdf (Дата обращения 15.01.2015)
104. P3M3 Portfolio Model // AXELOS Global Best Practice — 2013 [Электронный ресурс] URL: https://www.axelos.com/Corporate/media/ Files/P3M3%20Model/P3M3_Portfolio_Model.pdf (Дата обращения 15.01.2015)
105. P3M3 Programme Model // AXELOS Global Best Practice — 2013 [Электронный ресурс] URL: https://www.axelos.com/ Corporate/media/ Files/P3M3%20 Model/ P3M3_ Programme_Model.pdf (Дата обращения 15.01.2015)
106. P3M3 Project Model // AXELOS Global Best Practice — 2013 [Электронный ресурс] URL: https://www.axelos.com/ Corporate/media/Files/ P3M3%20Model/P3M3_Project_ Model. pdf (Дата обращения 15.01.2015)
107. Palmer S., Felsing J. A Practical Guide to Feature-Driven Development. / S. R. Palmer and J. M. Felsing // Upper Saddle River, NJ. – Prentice Hall PTR. – 2002
108. Pew, R. W. Some history of human performance models/R.Pew//In W. Gray (Ed.), Integrated models of cognitive systems (pp. 29–47). New York: CambridgeUniversity Press, 2007.
109. Poppendieck M., Poppendieck T. Lean Software Development / M. Poppendieck and T. Poppendieck // Boston. – Addison Wesley. – 2003
110. Program management of workflows in conceptual designing of automated systems / Y.A. Lapshov, V.A. Maklaev, P.I. Sosnin // Открытые семантические технологии проектирова-ния интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS-2013): материалы III Междунар. научн.-техн. конф. (Минск, 21-23 февраля 2013г.) / редкол. : В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2013. – 592 с.
111. Pseudo-Code Programming of Workflows in Conceptual Designing of Software Intensive System / Y.A. Lapshov, V.A. Maklaev, P.I. Sosnin // nteractive Systems: Problems of Human – Computer Interaction. –Collection of scientific papers. - Ulya-novsk: UlSTU, 2013. – 355 p. C.40-52
112. Ramchurn S. D. (2004) Minimising intrusiveness in pervasive computing environments using multi-agent negotiation / S.D. Ramchurn, B. Deitch, M.K.

Thompson, D.C. De Roure, N.R. Jennings, M. Luck // Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), Los Alamitos: IEEE Computer Society, 2004. – С. 364-372 [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Ramchurn-MobiQuitous04-01331743.pdf (Дата обращения: 25.09.2014)

113. Ras, E. Knowledge services for experience factories / E.Ras, J.Rech, S.Weber // In Proc. of the 5th Conference on Professional Knowledge Management, 2009, pp. 232-241.
114. Ratwani R. M. (2008) A Spatial Memory Mechanism for Guiding Primary Task Resumption / R. M. Ratwani // PhD Thesis. – George Mason University, Department of Psychology, 2008. [Электронный ресурс] Режим доступа: http://www.interruptions.net/literature/Ratwani-A_Spatial_Memory_ Mechanism_for_Guiding_Primary_Task_Resumption.pdf (Дата обращения: 15.09.2014)
115. Robertson, K. Project Management Maturity Model / K.Robertson. - http://www.klr.com/ white_papers /pdfs /pm_ maturity_ model.pdf.
116. Roglinger M. Maturity models in business process management / M.Roglinger, J.Poppelbuth, J.Becker // Business Process Management Journal, Vol. 18 (2), 2012, pp. 328 –346.
117. Schwaber K, Beedle M. Agile Software Development with SCRUM / K. Schwaber, M. Beedle // Upper Saddle River, NJ. – Prentice-Hall. – 2002
118. Sjoberg D. I. K., Johnsen A., Solberg J. Quantifying the Effect of Using Kanban Versus Scrum: A Case Study / D. I. K. Sjoberg, A. Johnsen, J. Solberg // IEEE Software, Vol. 29, No. 5, 2012. С. 47–53.
119. Sjoberg D.I. K. The Future of Empirical Methods in Software Engineering Research/D.I. K.Sjoberg, T. Dyba, M. Jorgensen// In Proceeding of the 2007 workshop Future of Software Engineering(FOSE '07),2007,- 358-378.
120. Sosnin, P. Means of question-answer interaction for collaborative development activity/P.I. Sosnin // Hindawi Publishing Corporation, Advances in Human-Computer Interaction , Volume 2009, Article ID 619405, USA, 2009, 18 pages.
121. Sosnin, P. Question-Answer Approach to Human-Computer Interaction in Collaborative Designing / P.I. Sosnin // Chapter in the book “Cognitively Informed Intelligent Interfaces: Systems Design and Development” Published IGI Global, 2012, pp. 157-176.
122. Sosnin, P. Question-Answer Expert System for Ship Collision Avoidance/P.I. Sosnin // In Proc of 51st International Symposium ELMAR – 2009, Zadar, Croatia, 2009. pp.185-188.
123. Sosnin, P. Question-Answer Means for Collaborative Development of Software Intensive Systems/Sosnin P.I. // Collection of scientific paper Complex Systems Concurrent Engineering, Part 3, Springer London, 2007, 151-

158.
124. Sosnin, P. Question-Answer Modeling in Conceptual Design of Automated Systems/P.I. Sosnin // In Proc. of the 13th IEEE Mediterranean Electrotechnical Conference – MELECON 2006, Malaga, Spain, 2006, pp.121-124.
125. Sosnin, P. Question-Answer Processor for Cooperative Work in Human-Computer Environment/Sosnin P.I. // In Proc. of the 2d International IEEE conference Intelligent System Varna, Bulgaria, 2004, pp. 452-456.
126. Sosnin, P. Question-answer programming and modeling in expert systems/P.I. Sosnin // In Proc. of Artificial Intelligence and Applications – AIA'2009, Vienne, Austria, pp.
127. Sosnin, P. Question-Answer Programming in Collaborative Development Environment/P.I. Sosnin // In Proc. of The third IEEE conference Cybernetics and Intelligent Systems – CIS-RUM'2010, Singapore, 2010, pp. 273-278.
128. Sosnin, P. Question-Answer Shell for Personal Expert System/P.I. Sosnin.- Chapter in the book "Expert Systems for Human, Materials and Automation." Published by Intech, 2011, pp. 51-74.
129. Sosnin, P.I. , Question-Answer System for Object-Oriented Analysis and Design/P.I.Sosnin,E.P. Sosnina // In Proc. of 22nd Conference on Infomatio Technology in Constructio, Dresden, Germany, CIB W78, 2005, pp.199-204.
130. Sosnin, P.I. Question–answer analysis in concurrent engineering/Sosnin P.I. // In Proc. of Concurrent Engineering: The Vision for the Future Generation in Research and Applications, Portugal, 2003, pp. 961-964.
131. Sosnin, P.I. Question-Answer Models of Decision-Making Tasks in Automated Designing/P.I. Sosnin // In Proc. of the 22nd European Conference on Modelling and Simulation – ECMS'2008, Nocosia, Cyprus, 2008, pp. 173-180.
132. Team FME. Project Management Skills. Project Management Principles, 2013. [Электронный ресурс] Режим доступа: www.free-management-ebooks.com.
133. The Standish group, Charting the Seas of Information Technology – Chaos, The Standish Group International, 1994.
134. Toward user-oriented software for collaborative modeling and simulation/S.Gorlatch, J.Muller-Iden, M.Alt, J.Dunnweber, H.Fujita Clayworks// Knowledge-Based Systems, Vol. 22 (3), 2006, pp. 209-215.
135. Turley F. An Introduction to PRINCE2®. // Project Smart – 2009 [Электронный ресурс] http://www.projectsmart.co.uk/docs/prince2-introduction-ps.pdf (Дата обращения 15.01.2015)
136. Vouk M, Rivers A.T. Construction of Reliable Software in Resource-Constrained Environments / M. Vouk, A.T. Rivers // in Case Studies in Reliability and Maintenance , W. R. Blischke and D. N. P. Murthy, Eds. Hoboken, NJ. – Wiley-Interscience, John Wiley and Sons, 2003, pp. 205-231.
137. Wang J. X . Kanban: Align Manufacturing Flow with Demand Pull / J. X

.Wang // Chapter in the book: Lean Manufacturing Business Bottom-Line Based, CRC Press, 2010, pp. 185-204.
138. Williams G. Prince2 Maturity Model // AXELOS Global Best Practice — 2013 [Электронный ресурс] URL: https://www.axelos.com/ Corporate/media/Files/P3M3%20Model/PRINCE2_Maturity_Model_P2MM. pdf (Дата обращения 15.01.2015)
139. Williams L, Cockburn A. Special Issue on Agile Methods / L. Williams and A. Cockburn // IEEE Computer, vol. 36, no. 3. – June 2003
140. Williams L. The XP Programmer: The Few Minutes Programmer / L. Williams // IEEE Software, vol. 20, no. 3, pp. 16-20. – 2003
141. Williams L., Kessler R. Pair Programming Illuminated. / L. Williams, R. Kessler // Reading, Massachusetts: – Addison Wesley. – 2003
142. Workflow patterns. Distributed and Parallel Databases/ Van der Aalst, A.H.M.Hofstede, B.Kiepuszewski, A.Barros. - 14, 2003, pp. 5–51.

Приложение 1. Шаблоны потоков работ

Ряд шаблонов управления потоками работ являются базовыми, другие шаблоны строятся на основе базовых, являясь их расширениями. Далее приведена таблица шаблонов потоков работ с описаниями.

Таблица П1. Шаблоны потоков работ

№	Шаблон потоков работ	Базовые шаблоны	Описание
1	Sequence	Sequence	Является базовым
2	Parallel Split	Parallel Split	Является базовым
3	Synchronization	Synchronization	Является базовым
4	Exclusive Choice	Exclusive Choice	Является базовым
5	Simple Merge	Simple Merge	Является базовым
6	Multi-Choice	Multi-Choice	Является базовым
7	Synchronizing Merge	Synchronizing Merge	Является базовым
8	Multi-Merge	Multi-Merge	Является базовым
9	Structured Discriminator	Discriminator	Реализация паттерна Discriminator
10	Blocking Discriminator	Discriminator	Реализация Discriminator, не позволяющая выполнить входящие в него задачи ещё раз до окончания выполнения исходящей задачи
11	Cancelling Discriminator	Discriminator, Cancel Activity	Реализация Discriminator, отменяющая выполнение всех входящих задач при окончании выполнения одной из них
12	Structured Partial Join	Multi-Merge, Discriminator	Реализация Discriminator, вызывающая исходящую задачу после окончания выполнения определенного числа любых входящих задач
13	Blocking Partial Join	Multi-Merge, Discriminator	Реализация Discriminator, вызывающая исходящую задачу после окончания выполнения определенного числа любых входящих задач, и не позволяющая запустить исходящую задачу, пока выполняется предыдущий её экземпляр
14	Cancelling Partial Join	Discriminator, Cancel Activity	Реализация Discriminator, отменяющая выполнение

			всех входящих задач при окончании выполнения любых двух или более из них
15	Generalized AND-Join	Synchronization	Synchronization, реализованный с учётом того, что какая-либо из входящих задач может выполниться несколько раз
16	Local Synchronizing Merge	Synchronizing Merge	Реализация Synchronizing Merge, выполняющая исходящую задачу при условии выполнения набора условий, определяемых в процессе выполнения задач на входящих ветвях, если все входящие задачи выполнены
17	General Synchronizing Merge	Synchronizing Merge	Реализация Synchronizing Merge, выполняющая исходящую задачу при условии выполнения набора условий, определяемых в процессе выполнения задач на входящих ветвях, если все входящие задачи выполнены или будут обязательно выполнены в будущем
18	Thread Merge	Sequence	Реализация Sequence таким образом, чтобы участник начал выполнять исходящую задачу после выполнения определенного количества входящих задач
19	Thread Split	Sequence	Реализация Sequence таким образом, чтобы участник начал выполнять заданный набор задач по окончанию выполнения входящей задачи
20	Multiple Instances without Synchronization	Multiple Instances without Synchronization	Является базовым
21	Multiple Instances with a Priori Design-Time	Multiple Instances with a Priori Design-Time	Является базовым

	Knowledge	Knowledge	
22	Multiple Instances with a Priori Run-Time Knowledge	Multiple Instances with a Priori Run-Time Knowledge	Является базовым
23	Multiple Instances without a Priori Run-Time Knowledge	Multiple Instances without a Priori Run-Time Knowledge	Является базовым
24	Static Partial Join for Multiple Instances		Реализация Sequence таким образом, чтобы участник начал выполнять исходящую задачу после выполнения определенного количества из заданных входящих задач
25	Cancelling Partial Join for Multiple Instances	Sequence	Реализация Sequence таким образом, чтобы участник начал выполнять исходящую задачу после выполнения определенного количества из заданных входящих задач с отменой выполнения оставшихся задач
26	Dynamic Partial Join for Multiple Instances	Sequence	Реализация Sequence таким образом, чтобы участник начал выполнять исходящую задачу после выполнения определяющегося в процессе выполнения количества из заданных входящих задач
27	Deferred Choice	Deferred Choice	Является базовым
28	Interleaved Parallel Routing	Interleaved Parallel Routing	Является базовым
29	Milestone	Milestone	Является базовым
30	Critical Section	Milestone	Набор задач, выполняемых участником, объявляется «критической секцией». При выполнении участником «критической секции», другие участники не могут начать другие «критические секции» до завершения выполнения её участником. Можно

			реализовать через Milestone.
31	Interleaved Routing	Milestone	Набор задач, для которых не допускается одновременное выполнение нескольких из них разными участниками. Можно реализовать через Milestone
32	Cancel Task	Cancel Task	Отмена выполнения задачи участником.
33	Cancel Case	Cancel Case	Является базовым
34	Cancel Region	Cancel Region	Отмена выполнения набора задач
35	Cancel Multiple Instance Activity	Cancel Task	Отмена выполнения нескольких экземпляров задачи участниками
36	Complete Multiple Instance Task		Отмена выполнения нескольких экземпляров задачи участником по условию
37	Arbitrary Cycles	Arbitrary Cycles	Является базовым
38	Structured Loop		Циклическое выполнение набора задач участником с предусловием или с постусловием
39	Recursion		Генерация задачей ситуации вызова самой себя
40	Implicit Termination	Implicit Termination	Является базовым
41	Explicit Termination	Simple Merge	Завершение работы по выполнению задач одним из участников
42	Transient Trigger		Выполнение задачи участником после ожидания выполнения условия. Условие может быть отменено
43	Persistent Trigger		Выполнение задачи участником после ожидания выполнения условия

Далее приведена запрограммированная на языке $\boldsymbol{L^{WIQA}}$ реализация базовых шаблонов потоков работ.

Q 3.10.1 Sequence – *Последовательное выполнение задач участником*
Q 3.10.1.1 &QueueId& := 1 - *Участник №1*
Q 3.10.1.2 &TaskId& := 1 – *выполняет задачу №1*

Q 3.10.1.3 CALL &AddTask& - *Добавление задачи на выполнение*
Q 3.10.1.4 &TaskId& := 2 – *Задача №2*
Q 3.10.1.5 CALL &AddTask& - *Добавляется на выполнение*
Q 3.10.1.6 ShowTable(&NSteps&, &Tasks&, &Steps&, &Queues&, &Priorities&, &States&) – *отображение таблицы KANBAN*
Q 3.10.1.7 FINISH – *завершение шаблона*

Q 3.10.2 Parallel Split
Q 3.10.2.1 &QueueId& := 1 – *Участник №1*
Q 3.10.2.2 &TaskId& := 1 *Задача №1*
Q 3.10.2.3 CALL &AddTask& *Добавление задачи на выполнение*
Q 3.10.2.4 &QueueId& := 2 – *Участник №2*
Q 3.10.2.5 &TaskId& := 2 – *выполняет задачу №2*
Q 3.10.2.6 CALL &AddTask& - *Добавление задачи на выполнение. Эти задачи участники выполняют одновременно*
Q 3.10.2.7 ShowTable(&NSteps&, &Tasks&, &Steps&, &Queues&, &Priorities&, &States&)
Q 3.10.2.8 FINISH

Q 3.10.3 Synchronization
Q 3.10.3.1 &TaskId& := 1 - *на этих двух шагах*
Q 3.10.3.2 CALL &GetTaskInfo& - *получение состояния выполнения задачи 1*
Q 3.10.3.3 IF &step& != 2 THEN GOTO &NOP& - *Если шаг рабочего процесса не равен 2*
Q 3.10.3.4 IF &state& != &done& THEN GOTO &NOP& - *или задача не выполнена, выполнять ничего не требуется*
Q 3.10.3.5 &TaskId& := 2 - *на этих двух шагах*
Q 3.10.3.6 CALL &GetTaskInfo& - *получение состояния выполнения задачи 2*
Q 3.10.3.7 IF &step& != 2 THEN GOTO &NOP& - *Если шаг рабочего процесса не равен 2*
Q 3.10.3.8 IF &state& != &done& THEN GOTO &NOP& - *или задача не выполнена, выполнять ничего не требуется*
Q 3.10.3.9 &TaskId&:= 3 - *Задача №3*
Q 3.10.3.10 &QueueId& := 1 - *для участника №1*
Q 3.10.3.11 CALL &AddTask& - *на выполнение*
Q 3.10.3.12 LABEL &NOP& - *Метка для перехода*
Q 3.10.3.13 FINISH

Q 3.10.4 Exclusive Choice
Q 3.10.4.1 IF &condition& THEN &TaskId& := 1 ELSE &TaskId& := *2* - *Если условие1, то выполняется задача №1, в противном случае – задача №2*
Q 3.10.4.2 &QueueId& := 1 – *задачу выполняет участник №1*
Q 3.10.4.3 CALL &AddTask&
Q 3.10.4.4 FINISH

Q 3.10.5 Simple Merge
Q 3.10.5.1 &TaskId& := 1
Q 3.10.5.2 CALL &GetTaskInfo&
Q 3.10.5.3 &Task1State& := &state& - *Сохранение состояния выполнения задачи*

№1

Q 3.10.5.4 &TaskId& := 2
Q 3.10.5.5 CALL &GetTaskInfo&
Q 3.10.5.6 IF (&state& == &done&) && (&Task1State& == &done&) THEN *BEGIN – Если обе задачи выполнены, то*
Q 3.10.5.6.1 &TaskId& := 3 - *Задача №3 на выполнение*
Q 3.10.5.6.2 &QueueId& := 1
Q 3.10.5.6.3 CALL &AddTask&
Q 3.10.5.6.4 END
Q 3.10.5.7 FINISH

Q 3.10.6 Multi Choice
Q 3.10.6.1 IF &Task1Needed& == TRUE THEN BEGIN – *Если нужно выполнить задачу1, то задача1 на выполнение*
Q 3.10.6.1.1 &TaskId& := 1
Q 3.10.6.1.2 &QueueId& := 1
Q 3.10.6.1.3 CALL &AddTask&
Q 3.10.6.1.4 END
Q 3.10.6.2 IF &Task2Needed& == TRUE THEN BEGIN – *Если нужно выполнить задачу2, то задача2 на выполнение*
Q 3.10.6.2.1 &TaskId& := 2
Q 3.10.6.2.2 &QueueId& := 2
Q 3.10.6.2.3 CALL &AddTask&
Q 3.10.6.2.4 END
Q 3.10.6.3 FINISH

Q 3.10.7 Synchronizing Merge
Q 3.10.7.1 &TaskId& := 1
Q 3.10.7.2 CALL &GetTaskInfo&
Q 3.10.7.3 IF &state& == &working& THEN GOTO &NOP&
Q 3.10.7.4 &TaskId& := 2
Q 3.10.7.5 CALL &GetTaskInfo&
Q 3.10.7.6 IF &state& == &working& THEN GOTO &NOP&
Q 3.10.7.7 &TaskId& : = 3 - *задача 3 на выполнение только после того, как задачи 1 и 2 выполнены.*
Q 3.10.7.8 &QueueId& : = 1
Q 3.10.7.9 CALL &AddTask&
Q 3.10.7.10 FINISH

Q 3.10.8 Multi Merge
Q 3.10.8.1 &TaskId& := 1
Q 3.10.8.2 CALL &GetTaskInfo&
Q 3.10.8.3 IF &state& == &done& THEN BEGIN - *Если задача1 выполнена, то участник, выполнивший её, выполняет задачу 3*
Q 3.10.8.3.1 &TaskId& := 3
Q 3.10.8.3.2 CALL &AddTask&
Q 3.10.8.3.3 END
Q 3.10.8.4 &TaskId& := 2

Q 3.10.8.5 CALL &GetTaskInfo&
Q 3.10.8.6 IF &state& == &done& THEN BEGIN - *Если задача2 выполнена, то участник, выполнивший её, выполняет задачу 3*
Q 3.10.8.6.1 &TaskId& := 3
Q 3.10.8.6.2 CALL &AddTask&
Q 3.10.8.6.3 END
Q 3.10.8.7 FIINISH

Q 3.10.9 Discriminator
Q 3.10.9.1 &TaskId& := 1
Q 3.10.9.2 CALL &GetTaskInfo&
Q 3.10.9.3 IF &state& == &working& THEN GOTO &nexttask&
Q 3.10.9.4 TaskId := 2
Q 3.10.9.5 CALL &InterruptTask& - *прерывание задачи 2, если задача 1 уже выполнена*
Q 3.10.9.6 GOTO &DISC_END&
Q 3.10.9.7 LABEL &nexttask&
Q 3.10.9.8 &TaskId& := 2
Q 3.10.9.9 CALL &GetTaskInfo&
Q 3.10.9.10 IF &state& == &working& THEN GOTO &DISC_END&
Q 3.10.9.11 &TaskId& := 1
Q 3.10.9.12 CALL &InterruptTask& - *прерывание задачи 1, если задача 2 уже выполнена*
Q 3.10.9.13 LABEL &DISC_END&
Q 3.10.9.14 FINISH

Q 3.10.10 Muliple Instance without Synchronization
Q 3.10.10.1 &Person& := 0
Q 3.10.10.2 LABEL &L1&
Q 3.10.10.3 &TaskId& := 3
Q 3.10.10.4 &QueueId& := &Person&
Q 3.10.10.5 CALL &AddTask&
Q 3.10.10.6 &person& := &person& + 1
Q 3.10.10.8 IF &condition& THEN GOTO &L1& - пока выполняется условие, каждому участнику на выполнение задача3
Q 3.10.10.9 FINISH

Q 3.10.11 Multiple Instance with a priori Design-time Knowledge
Q 3.10.11.1 Установка начальных значений
Q 3.10.11.1.1 &Person& := 0
Q 3.10.11.1.2 &cnt& := 0
Q 3.10.11.2 LABEL &L1&
Q 3.10.11.3 &TaskId& := 3
Q 3.10.11.4 &QueueId& := &Person&
Q 3.10.11.5 CALL &AddTask&
Q 3.10.11.6 &person& := &person& + 1
Q 3.10.11.7 &cnt& := &cnt& +1
Q 3.10.11.8 IF &cnt& <= 5 THEN GOTO &L1& - *пока выполняется условие, каждому участнику на выполнение задача3. Во время проектирования паттерна условие известно*

Q 3.10.11.9 FINISH

Q 3.10.12 Multiple Instance with a priori Run-time Knowledge
Q 3.10.12.1 Установка начальных значений
Q 3.10.12.1.1 &Person& := 0
Q 3.10.12.1.2 &cnt& := 0
Q 3.10.12.2 LABEL &L1&
Q 3.10.12.3 &TaskId& := 3
Q 3.10.12.4 &QueueId& := &Person&
Q 3.10.12.5 CALL &AddTask&
Q 3.10.12.6 &person& := &person& + 1
Q 3.10.12.7 &cnt& := &cnt& +1
Q 3.10.12.8 IF &cnt& <= &needed& THEN GOTO &L1& - *пока выполняется условие, каждому участнику на выполнение задача3. Во время выполнения паттерна условие становится известно*
Q 3.10.12.9 FINISH

Q 3.10.13 Muliple Instance without a priori Run-time knowledge
Q 3.10.13.1 &Person& := 0
Q 3.10.13.2 LABEL &L1&
Q 3.10.13.3 &TaskId& := 3
Q 3.10.13.4 &QueueId& := &Person&
Q 3.10.13.5 CALL &AddTask&
Q 3.10.13.6 &person& := &person& + 1
Q 3.10.13.7 IF &person& >= &MaxQueues& THEN &person& := 0
Q 3.10.13.8 IF &NeededToContinue& == TRUE THEN GOTO &L1& - *пока выполняется условие, каждому участнику на выполнение задача3. Условие NeedToContinue формируется вне данного шаблона.*
Q 3.10.13.9 FINISH

Q 3.10.14 Deferred Choice
Q 3.10.14.1 LABEL &L1&
Q 3.10.14.2 IF &condition1& THEN BEGIN – *Если выполнено условие1, то задача 1 на выполнение*
Q 3.10.14.2.1 &TaskId& := 1
Q 3.10.14.2.2 CALL &AddTask&
Q 3.10.14.2.3 GOTO &pend& - *завершение работы шаблона*
Q 3.10.14.2.4 END
Q 3.10.14.3 IF &condition2& THEN BEGIN– *Если выполнено условие2, то задача 2 на выполнение*
Q 3.10.14.3.1 &TaskId& := 2
Q 3.10.14.3.2 CALL &AddTask&
Q 3.10.14.3.3 GOTO &pend& - *завершение работы шаблона*
Q 3.10.14.3.4 END
Q 3.10.14.4 GOTO &L1& - *повторение шаблона пока не будет выполнено любое из условий*
Q 3.10.14.5 LABEL &pend&
Q 3.10.14.6 FINISH

Q 3.10.15 Interleaved Parallel Routing
Q 3.10.15.1 LABEL &L1&
Q 3.10.15.2 IF &Task1NotRunned& == TRUE THEN BEGIN – *Если задача 1 не была запущена, запуск на выполнение задачи 1*
Q 3.10.15.2.1 &TaskId& := 1
Q 3.10.15.2.2 CALL &AddTask&
Q 3.10.15.2.3 &Task1NotRunned& := FALSE
Q 3.10.15.2.4 END
Q 3.10.15.3 &TaskId& := 2
Q 3.10.15.4 CALL &GetTaskInfo&
Q 3.10.15.5 IF &state& == &done& THEN BEGIN – *Если задача 2 выполнена, задача 3 на выполнение*
Q 3.10.15.5.1 &TaskId& := 3
Q 3.10.15.5.2 CALL &AddTask&
Q 3.10.15.5.3 END ELSE GOTO &L1& - *иначе повторение операций шаблона*
Q 3.10.15.6 FINISH

Q 3.10.16 Milestone
Q 3.10.16.1 LABEL &L1&
Q 3.10.16.2 &TaskId& := 3
Q 3.10.16.3 CALL &GetTaskInfo&
Q 3.10.16.4 IF &state& == &working& THEN GOTO &L1&
Q 3.10.16.5 &TaskId& := 7 – *задача 7 на выполнение только после того, как задача 3 выполнена. Шаблон дожидается выполнения задачи 3.*
Q 3.10.16.6 CALL &AddTask&
Q 3.10.16.7 FINISH

Q 3.10.17 Arbitrary Cycle
В зависимости от состояния на выполнение задача 1, 2 или 3
Q 3.10.17.1 if &pstate& == 1 THEN &TaskId& := 1
Q 3.10.17.2 if &pstate& == 2 THEN &TaskId& := 2
Q 3.10.17.3 if &pstate& == 3 THEN &TaskId& := 3
Q 3.10.17.4 CALL &AddTask&
Состояние определяется в зависимости от условий.
Q 3.10.17.5 IF &condition1& == True THEN &pstate& := 1
Q 3.10.17.6 IF &condition2& == True THEN &pstate& := 2
Q 3.10.17.7 IF &condition3& == True THEN &pstate& := 3
Q 3.10.17.8 FINISH

Q 3.10.18 Implicit Termination
Q 3.10.18.1 Процесс завершается автоматически когда очередь задач сотрудника оказывается пустой

Q 3.10.19 Cancel Activity
Q 3.10.19.1 &QueueId& := 1 – *участник №1*
Q 3.10.19.2 CALL &InterruptNow& - *прерывание выполняемой задачи*
Q 3.10.19.3 CALL &UnqueueTasks& - *освобождение очереди задач участника*
Q 3.10.19.4 FINISH

Q 3.10.20 Cancel Case – *прерывание потока задач нескольких участников*
Q 3.10.20.1 &QueueId& := 1– *участник №1*
Q 3.10.20.2 CALL &InterruptNow& - *прерывание выполняемой задачи*
Q 3.10.20.3 CALL &UnqueueTasks& - *освобождение очереди задач участника*
Q 3.10.20.4 &QueueId& := 2– *участник №2*
Q 3.10.20.5 CALL &InterruptNow&- *прерывание выполняемой задачи*
Q 3.10.20.6 CALL &UnqueueTasks& - *освобождение очереди задач участника*
Q 3.10.20.7 FINISH

Приложение 2. Эксперименты

Эксперимент на потоке работ проекта «Сервис профессиональной тренировки сотрудников»

Целью данного эксперимента также является проверка гипотезы ***H1***. Рассмотрим **условия проведения** данного эксперимента. Одним из потоков работ, над которыми выполнялись экспериментальные исследования в рамках данной диссертационной работы, является поток работ, выполняемый на предприятии **ООО «Центр технического творчества»** в рамках проекта **«Сервис профессиональной тренировки сотрудников».** Для выполнения эксперимента были выбраны задачи, выполняемые членами проектной группы в процессе работы над проектом. Здесь под словом «Проект» будем подразумевать ту часть проекта, которая была выполнена в сроки проведения эксперимента. Проект этот включает в себя в общей сложности **19** задач и подзадач. На выполнение его был выделен следующий временной интервал – с 15.04.2014 по 15.07.2014. Задачи выполнял коллектив из **6** человек и руководителя.

Таблица П2. Коллектив проектировщиков

Сотрудник	**Роль**	**Обозначение**
Святов К. В.	Руководитель	D1
Федотов А. А.	Разработчик	D2
Майоров А. И.	Разработчик	D3
Комаров И. Д.	Разработчик	D4
Солдатова Е. Н.	Разработчик	D5
Тимофеев С. Н.	Разработчик	D6
Черных Н.Ф.	Разработчик	D7

В качестве **плана** данного эксперимента также была использована методика, представленная в п. 4.3.

В процессе выполнения данного эксперимента для каждой задачи руководителем группы было составлено поручение, затем им были сформированы бэклоги спринтов, состоящих из задач, выполнение которых совпадало со временем проведения эксперимента.

Рассмотрим фрагмент процесса формирования набора задач проекта с позиции **программирования потока работ**.

D1 как руководитель группы назначил *D3* поручения для выполнения двух задач:

- Интерфейс с БД для тестирования;
- Интерфейс с БД для руководителей.

С точки зрения выполнения всей проектной работы эти задачи равноприоритетны, их выполнение не зависит друг от друга, но начало выполнения зависит от выполнения задачи «Разработка БД системы», выполняемой проектировщиком *D4*.

Таким образом, можно говорить о шаблоне ***Parallel Split*** потока работ, т.к. эти задачи можно выполнять параллельно.

В процессе создания поручения для этих задач было выставлено условие:

&TaskEnabled& := TaskFinishedByName("Разработка БД системы");

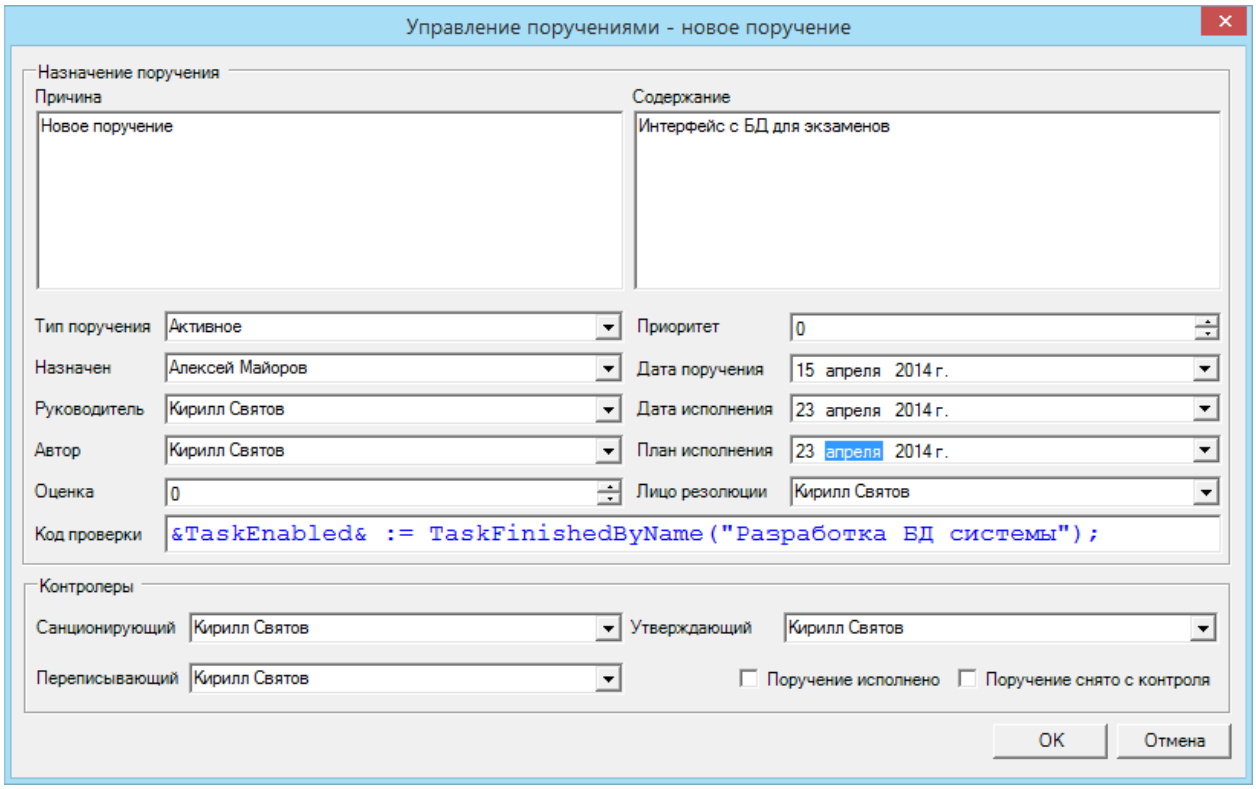


Рис. П1. Создание поручений

Это же условие было унаследовано и подзадачами. Для задачи «Интерфейс с БД для экзаменов» следующие:

- Разработать хранимые процедуры и триггеры;
- Разработать пользовательские интерфейсы для экзаменуемых;
- Разработать пользовательские интерфейсы для экзаменатора;
- Разработать интерфейс журнала;
- Разработать интерфейс статистики;

Далее, в процессе **оперативного планирования и формирования спринта**, включающего в себя данные задачи, условия их выполнения были дополнены. Важно отметить, что эти условия учитываются совместно с условиями, выставленными в процессе формирования поручения через логическую операцию «***И***», т.е. для начала выполнения задачи требуется соблюдение всех этих условий.

Итак, первой должна выполняться задача «Разработать хранимые процедуры и триггеры», после которой можно приступать к выполнению

других задач, таким образом, на данном этапе опять образуется конструкция, соответствующая шаблону ***Parallel Split***. Чтобы *D3* не пришлось выполнять остальные задачи в течение времени спринта псевдопараллельно, затрачивая время на переключение между задачами, он запланировал их последовательное выполнение, соответствующее паттерну ***Sequence***. Но при этом последовательность выполнения этих задач не является строгой, поэтому для их выполнения не требовалась установка дополнительных условий, было достаточно расстановки численных приоритетов. Для задачи «Интерфейс с БД для преподавателей» был установлен меньший приоритет. Приоритеты были расставлены следующие:

Таблица П3. Численные приоритеты задач

Задача	**Приоритет**
Разработать пользовательские интерфейсы для тестируемых сотрудников	500
Разработать пользовательские интерфейсы для руководителя	400
Разработать интерфейс журнала	300
Разработать интерфейс статистики	200
Интерфейс с БД для руководителя	100

После планирования спринта он был запущен на выполнение. В начале выполнения спринта для установленных задач были выставлены следующие условия:

Задача «Интерфейс с БД для тестируемых сотрудников»:

Таблица П4. Условия для задачи «Интерфейс с БД для тестируемых сотрудников»

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Не выполнено
Условие оперативного планирования	Нет	-

Задача «Разработать хранимые процедуры и триггеры»:

Таблица П5. Условия для задачи «разработать хранимые процедуры и триггеры»

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Не выполнено

Условие оперативного планирования	Нет	-

Для остальных рассмотренных выше задач *D3*:

Таблица П6. Условия для выполнения задач

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Не выполнено
Условие оперативного планирования	&TaskEnabled& := TaskFinishedByName("Разработать хранимые процедуры и триггеры")	Не выполнено

После того, как участник *D4* выполнил задачу «Разработка БД системы», условие планирования проектных работ для рассмотренных выше задач начало выполняться и *D3* приступил к решению задачи «Интерфейс с БД для экзаменов», начиная с подзадачи «Разработать хранимые процедуры и триггеры», для которой условия выполнения оказались следующими:

Таблица П7. Условия для выполнения задачи «Разработать хранимые процедуры и триггеры»

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Выполнено
Условие оперативного планирования	Нет	-

После выполнения этой задачи стали доступными для выполнения другие задачи, для которых были установлены численные приоритеты. *D3* выбрал из очереди задачу, обладающую наивысшим из них: «Разработать пользовательские интерфейсы для тестируемых». Её условия для выполнения:

Таблица П8. Условия для выполнения задачи «Разработать пользовательские интерфейсы для тестируемых сотрудников»

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Выполнено
Условие оперативного планирования	&TaskEnabled& := TaskFinishedByName("Разработать хранимые процедуры и триггеры")	Выполнено

В процессе решения задачи «Разработать интерфейс журнала» *D3*

обнаружил, что в базе данных не существует механизма для учета сотрудников, работающих по совместительству в разных отделах.

При этом возникла новая, имеющая максимальный приоритет задача для *D4*: «Разработать механизм учета совместительства сотрудников». *D4* произвел прерывание выполняемой им в тот момент задачи и приступил к решению новой.

D3 также прервал выполнение задачи «Разработать интерфейс журнала» и изменил её приоритет с 300 до 150. Также им было установлено динамическое условие, возникшее в процессе выполнения спринта:

Таблица П9. Условия для выполнения задачи «Разработать интерфейс журнала»

Вид условия	**Условие**	**Выполнение**
Условие планирования проектных работ	&TaskEnabled& := TaskFinishedByName("Разработка БД системы")	Выполнено
Условие оперативного планирования	&TaskEnabled& := TaskFinishedByName("Разработать хранимые процедуры и триггеры")	Выполнено
Динамическое условие	&TaskEnabled& := TaskFinishedByName("Разработать механизм перевода студенческих групп")	Не выполнено

После этого *D3* приступил к решению следующей в очереди задачи: «Разработать интерфейс статистики». В тот момент, когда он закончил работу над ней, все условия, включая динамическое, для выполнения задачи «Разработать интерфейс журнала» выполнялись, и он продолжил работу над ней.

Теперь рассмотрим решение задач сотрудником *D4* с позиции параллелизма. В спринте для выполнения ему были назначены следующие задачи:

Таблица П10. Задачи D4

Задача	**Обозначение**
Разработка БД системы	Z1
Разработка средств авторизации и аутентификации	Z2
Менеджер профилей пользователей	Z3

Задача Z1 включает в себя следующие подзадачи:

Таблица П11. Подзадачи Z1

Задача	**Обозначение**
Разработка БД на уровне диаграмм	Z1.1
Размещение таблиц в СУБД	Z1.2
Программирование базовых хранимых процедур и триггеров	Z1.3

Задача Z2 включает в себя следующие подзадачи:

Таблица П12. Подзадачи Z2

Задача	**Обозначение**
Разработка механизма авторизации и аутентификации	Z2.1
Разработка пользовательского интерфейса входа в систему	Z2.2
Разработка инструмента восстановления пароля	Z2.3

Задача Z3 включает в себя следующие подзадачи:

Таблица П13. Подзадачи Z3

Задача	**Обозначение**
Разработка личного кабинета пользователя	Z3.1
Разработка личного кабинета администратора	Z3.2

Разработчик D4 должен был начать свою работу над спринтом с выполнения задачи Z1, подзадачи которой могли выполняться только последовательно, затем после того, как задача Z1 завершена, D4 получил возможность приступить к выполнению задач Z2 и Z3, выполнением которых он занимался параллельно. Таким образом, задачи формируют последовательность шаблонов *Sequence* и *Parallel Split*.

Теперь рассмотрим псевдопараллелизм в выполнении задач разработчиком D4 в отрезке времени, соответствующем 11 и 12 дням спринта. Численный приоритет для этих задач был установлен в 100 единиц.

Таблица П14. Очереди задач

День спринта	**Очередь задач для выполнения**
11	Z2.1, Z2.2, Z2.3, Z3.1
12	Z2.2, Z2.3, Z3.1

В середине 12-го дня спринта появилась новая задача Zm, имеющая наивысший приоритет – «Разработать механизм учета совместительства

сотрудников». Численный приоритет для неё был установлен в 500 единиц. Вычисление динамического условия выполнения для задач Z спринта в 11-й день происходило в соответствии с выражением:

Таблица П15. Динамическое условие выполнения 11-го дня

Задача	**Динамическое условие для выполнения**
Z2.1	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 0
Z2.2	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 1
Z2.3	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 2
Z3.1	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 3

Здесь:

- &StTime& - время начала рабочего дня в секундах
- &quant& - квант времени в минутах на задачу, был установлен в 20 минут
- &nz& - количество параллельно выполняемых задач в очереди. Здесь &nz& = 4.

В 12-й день условия были установлены следующие:

Таблица П16. Динамическое условие выполнения 12-го дня

Задача	**Динамическое условие для выполнения**
Z2.2	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 0
Z2.3	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 1
Z3.1	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 2
Z3.2	&TaskEnabled& := ((&StTime& / 60)* &quant&) mod &nz& == 3

Для задачи Zm &TaskEnabled& был установлен в 1 без дополнительных условий.

При этом задачи Z1.1 – Z3.2 в течение этих дней выполнялись отрезками времени по 20 минут, после чего происходило прерывание – задача оказывалась заблокированной для выполнения в соответствии с динамическим условием. При этом разблокировалась следующая задача, и так далее, по очереди. Возникшая задача Zm в 12-й день получила больший приоритет, чем остальные задачи и была выполнена до конца вне зависимости от &StTime&.

В процессе выполнения спринта была получена следующая диаграмма

выгорания:

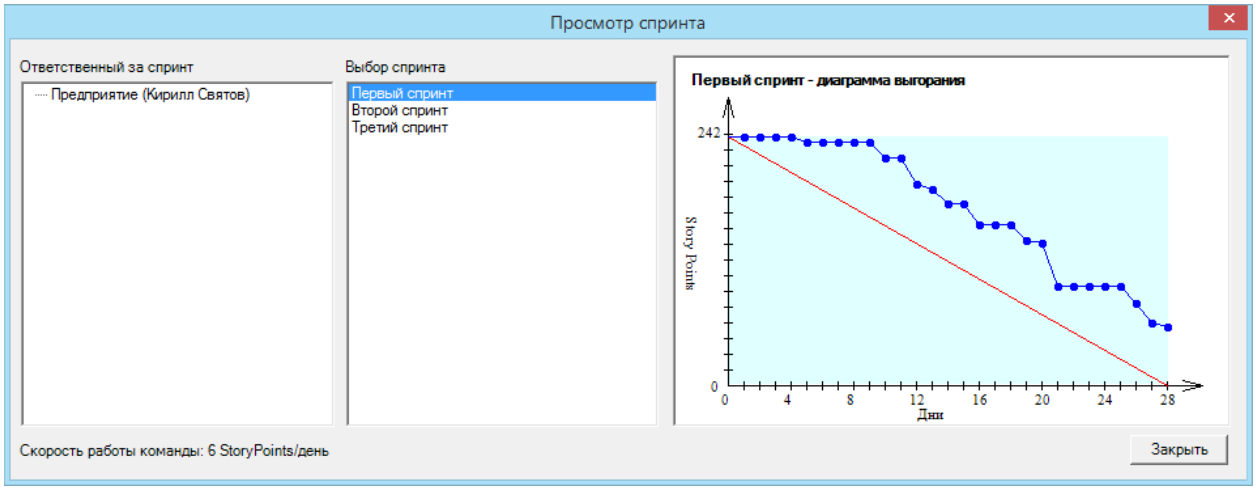


Рис. П2. Выполнение первого спринта

Во время выполнения первого спринта все запланированные задачи не были выполнены командой разработчиков, команда выполнила 185 из запланированных 242 StoryPoints. Руководитель команды оценил планирование данного спринта на 4 по 10-балльной шкале, соответственно:

- TV = 6,6 StoryPoints/сут.;
- TD = 4, соответственно:
- коэффициент x = 9,5

Второй спринт был запланирован с учетом результатов выполнения первого. Эти результаты показывают, что возможности команды позволяют выполнить со скоростью 6,6 StoryPoints/сут. не более 185 StoryPoints за 28 дней спринта.

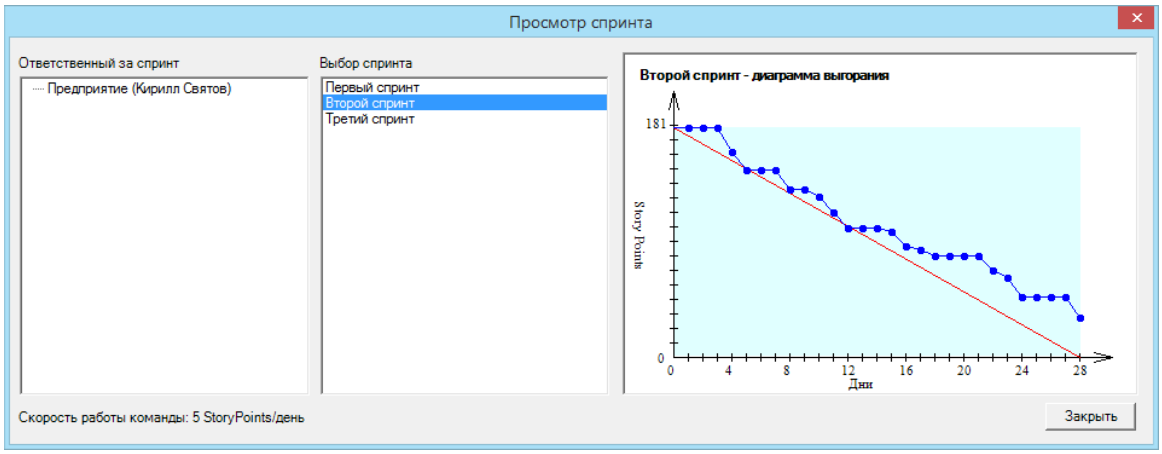


Рис. П3. Выполнение второго спринта

Выполнение второго спринта также показало, что возможности команды оказались переоценены, и она не полностью справилась с поставленными задачами. Команда выполнила 150 StoryPoints из 181. Метрики, вычисленные по результатам выполнения спринта, оказались следующими:

- TV = 5,4 StoryPoints/сут.;
- TD = 6,7 баллов.

Метрика ***TD*** в этом случае была вычислена с использованием коэффициента x, полученного в ходе расчета метрик первого спринта. Скорость команды в этом случае оказалась ниже, т.к. один из сотрудников (Елена Солдатова) в связи с временной нетрудоспособностью не участвовала в проектном процессе в течение пяти рабочих дней.

Третий спринт руководитель проектной группы планировал с учетом результатов выполнения первого и второго спринтов, опираясь на скорость команды.

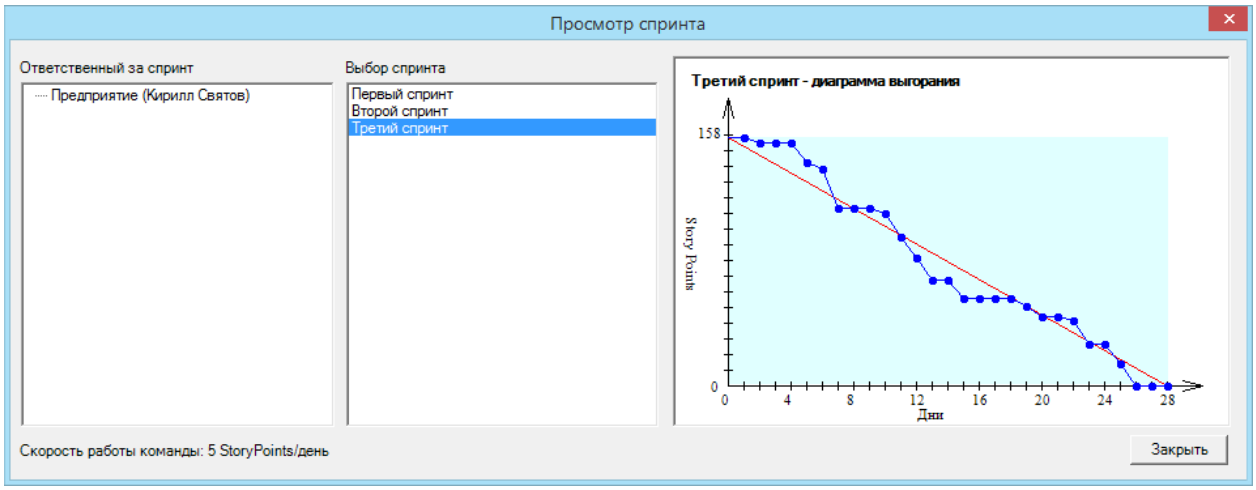


Рис. П4. Выполнение третьего спринта

В процессе его выполнения команда выполнила все запланированные проектные задачи на 3 дня раньше срока окончания спринта. Руководитель проекта оценил планирование данного спринта на 9 баллов по 10-балльной шкале. В результате его выполнения получены следующие метрики:

- TV = 6,1 StoryPoints/сут.;
- TD = 9 баллов.

Коэффициент **у** был вычислен как 0,33.

На создание поручений руководителем проектной группы было затрачено 1,5 часа времени. На планирование каждого спринта командой разработчиков было затрачено по 1 часу времени, на каждую ежесуточную встречу было затрачено 15 минут. Руководитель проектной группы оценил данные временные затраты как приемлемые.

На основании проведенного эксперимента были сделаны следующие **выводы**:

1. С каждым следующим этапом проектной работы оценка планирования повышается, что подтверждает гипотезу ***Н1***. Также её подтверждает оценка командой временных затрат на использование средств ПКУ как приемлемых.
2. Выполнение второго спринта показало реализацию одного из возможных рисков – временная утрата трудоспособности сотрудника, что отразилось на скорости работы команды. При выполнении достаточного для статистики количества реализаций подобных рисков появляется возможность их учета в процессе оперативного планирования этапа проектной работы, т.к. они повлияют на среднюю скорость работы команды.

Эксперимент на потоке работ данного диссертационного исследования

Одной из **целей** данного эксперимента является проверка следующих гипотез:

Н2. Применение ПКУ в персональной работе позволяет выполнить задачи самоконтроля.

Н3. Разработанные инструментальные средства ПКУ работают корректно.

Второй целью эксперимента является проверка корректности работы разработанных средств ПКУ.

В качестве **входных параметров** выступает набор задач данного диссертационного исследования, выполненных в течение 2014 года. В качестве **выходных параметров** выступает картотечная визуализация задач и их приоритетов, а также – временные затраты на работу со средствами ПКУ.

В качестве плана проведения данного эксперимента выступает следующая методика:

1. Добавить задачи диссертационного исследования в проект – *автор данной диссертационной работы.*
2. Для каждой задачи создать поручение с указанием времени примерного завершения работы над ней и её приоритета – *автор данной диссертационной работы*;
3. При возникновении новой задачи:
   3.1. Добавить новую задачу в проект – *автор данной диссертационной работы;*
   3.2. Для новой задачи создать поручение с указанием примерного времени завершения работы над ней, а также – её приоритета – *автор данной диссертационной работы*;
4. При завершении работы над задачей:
   4.1. Отметить задачу как завершенную;
   4.2. С использованием инструмента картотечной визуализации выбрать новую задачу для выполнения.

Выполнение эксперимента автором данной диссертационной работы было начато с формирования проекта, содержащего задачи работы над текстом диссертации:

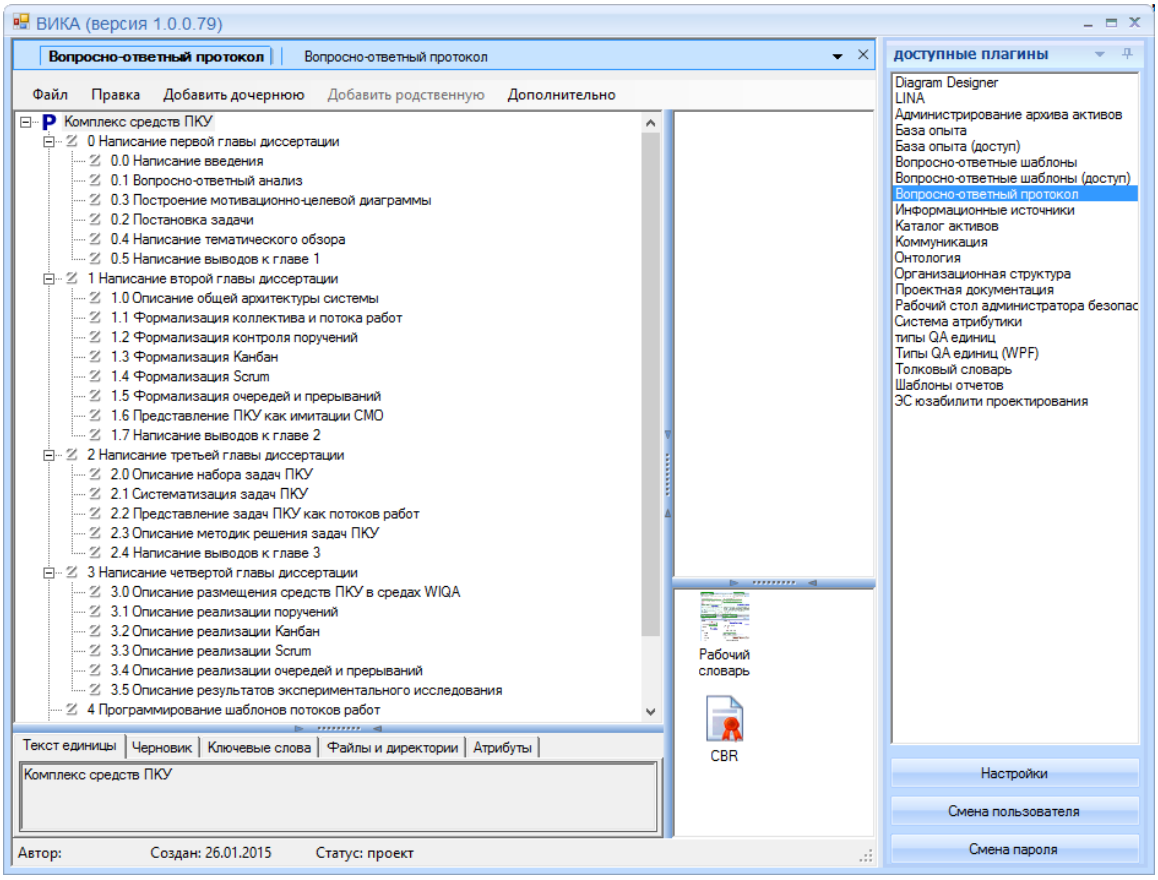


Рис. П5. Выполнение третьего спринта

На добавление задачи в проект в среднем было затрачено менее 1 минуты времени.

Затем в проект были включены задачи ПКУ и для каждой задачи диссертации было сформировано поручение:

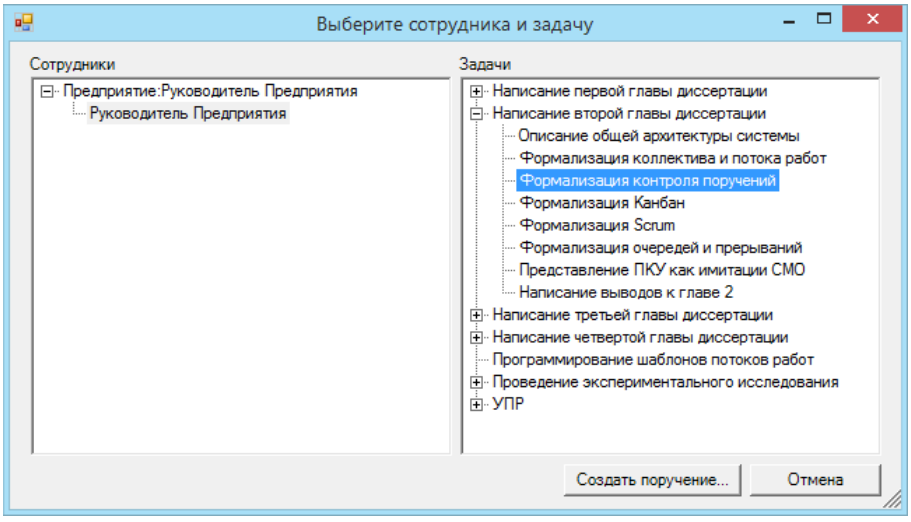


Рис. П6. Выбор задачи для создания поручения

На рисунке П6 представлен процесс выбора задачи для создания

поручения. Единственный сотрудник предприятия, обозначенный как руководитель предприятия – автор данной работы, выполняющий задачи проекта, организационная структура не создавалась.

Далее заполнение атрибутов поручения:

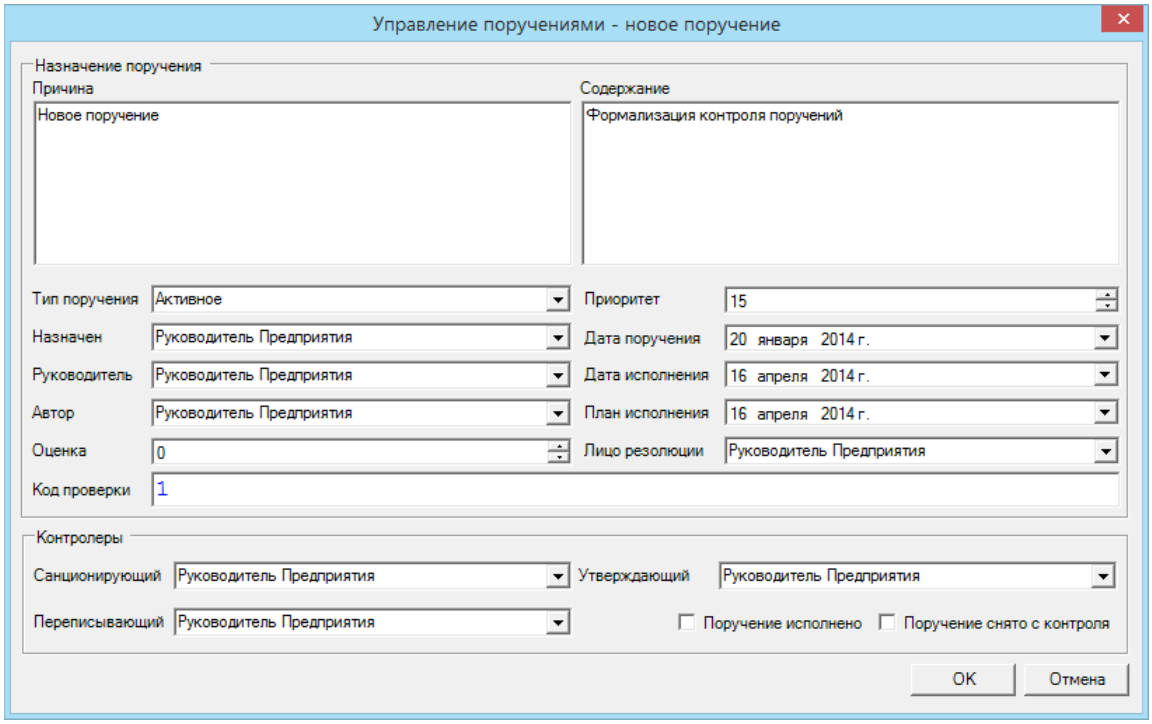


Рис. П7. Заполнение атрибутов поручения

На создание одного поручения также требуется около 1 минуты времени. После того, как все поручения были созданы, с использованием картотечной визуализации происходил выбор задач для дальнейшего выполнения и выполнение этих задач.

Задачи располагаются в порядке приоритетов для их выполнения, на выбор задачи требуется также менее 1 минуты времени. Временные затраты на использование данной системы были оценены автором как приемлемые.

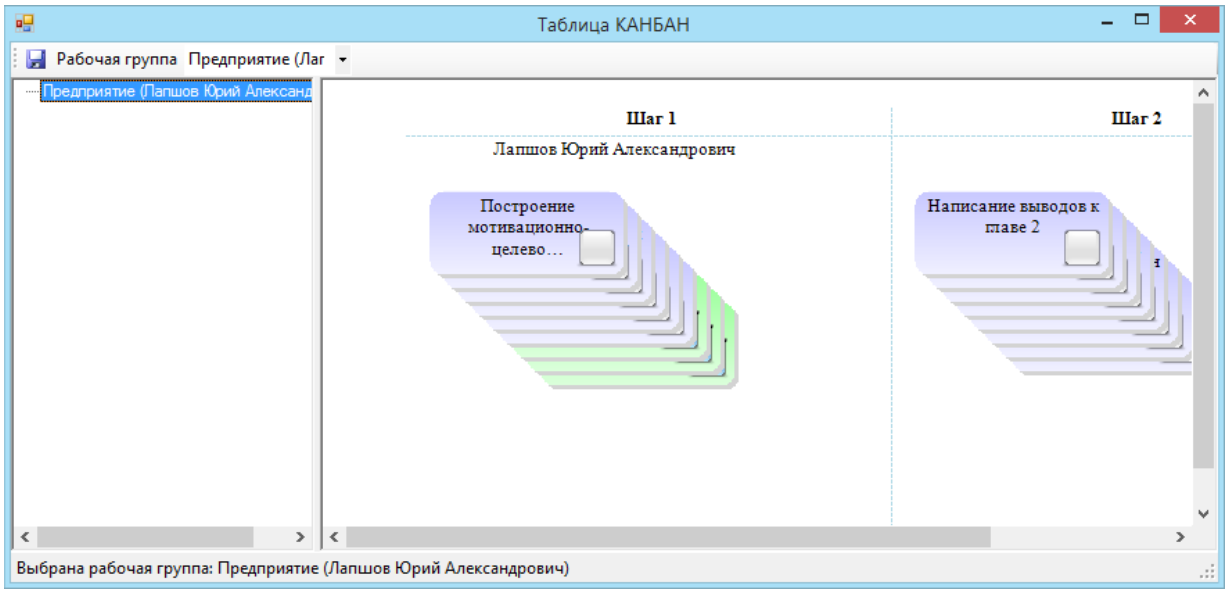


Рис. П8. Очередь задач в Kanban

После выполнения задачи она отмечается как выполненная. На постановку такой отметки также требуется менее 1 минуты времени. У выполненных задач приоритет устанавливается как минимально возможный, соответственно, они оказываются в нижней части «стопки» карточек, визуализирующих очередь задач для выполнения.

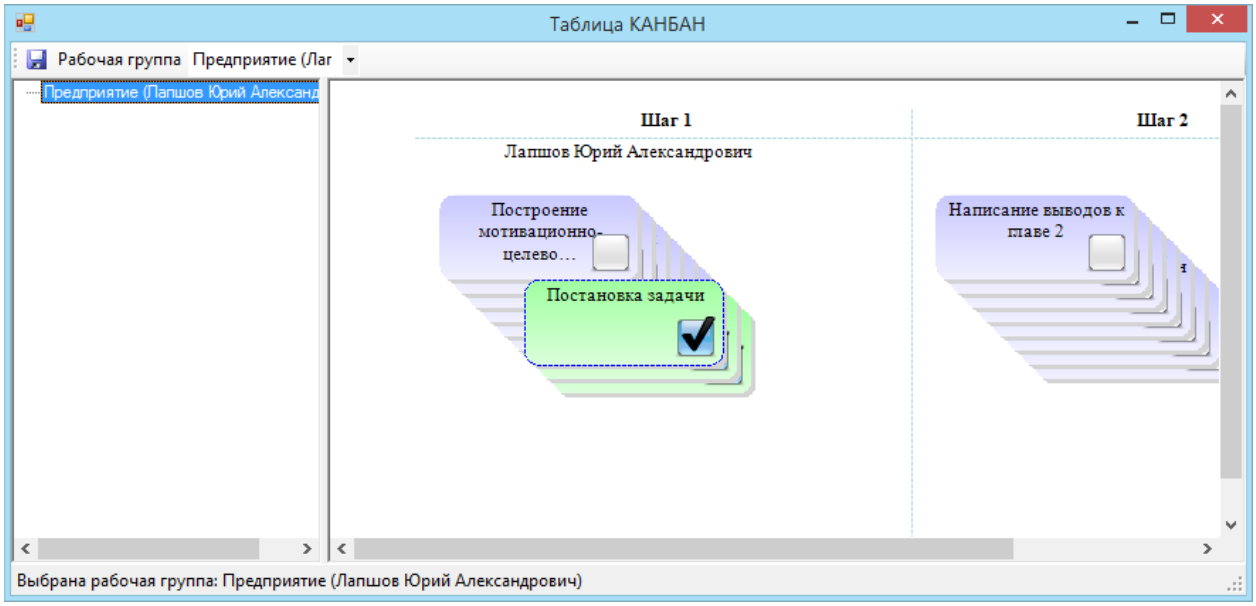


Рис. П9. Выполненная задача в Kanban

При возникновении новой задачи с максимальным приоритетом, она оказывается в верхней части «стопки» задач.

В данном случае возникла задача написания статьи на тему «Programmable Managing of Workflows in Development of Software-Intensive Systems». Для данной задачи было создано поручение с максимальным приоритетом, равным 100.

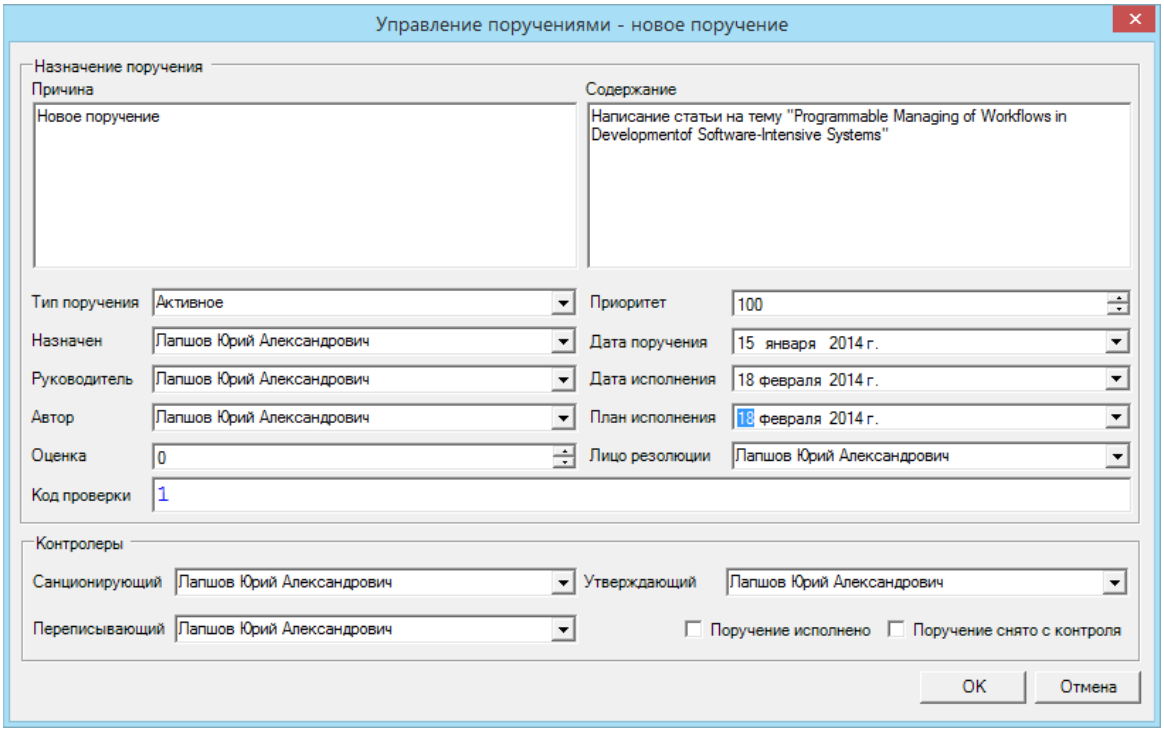


Рис. П10. Создание нового поручения

Поскольку новое поручение по времени затрагивает два месячных этапа работы, карточки появляются как в первой, так и во второй колонке:

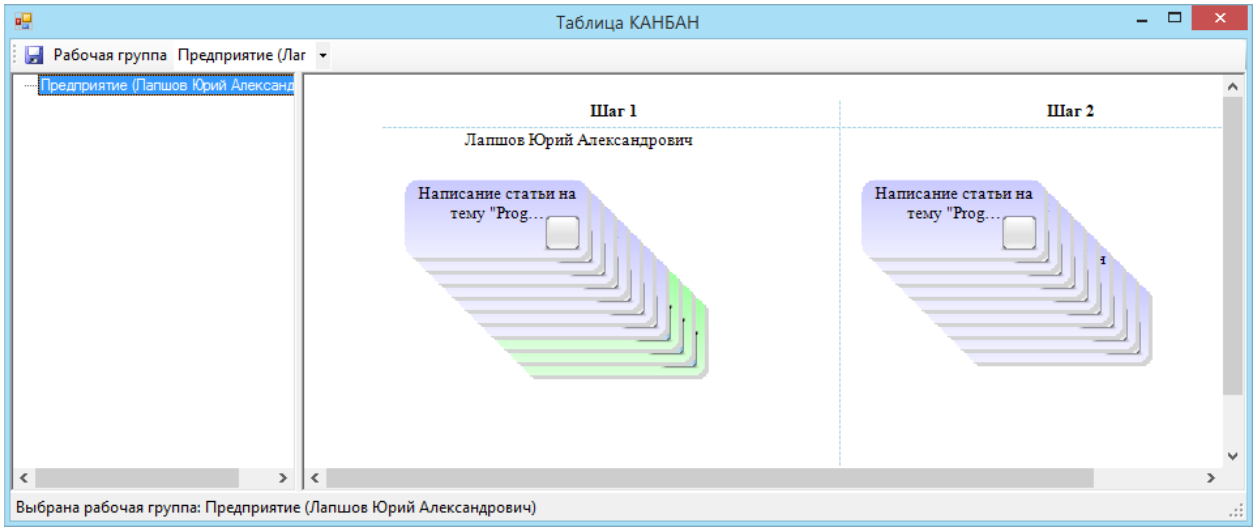


Рис. П11. Новая высокоприоритетная задача в Kanban

Также, в во время выполнения данных работ возникла задача, связанная с реализацией

средств ПКУ в инструментальной среде OwnWIQA. Эта задача:

- Реализация имитационной оргструктуры в среде OwnWIQA (***Z**im*).

Данная задача состоит из следующих подзадач:

- Разработка общей архитектуры имитационной оргструктуры (***Z**im1*);
- Разработка базы данных имитационной оргструктуры (***Z**im2*);
- Разработка экранных форм имитационной огрструктуры (***Z**im3*);
- Разработка псевдокодовых алгоритмов управления имитационной оргструктурой (***Z**im4*);
- Документирование разработанных средств (***Z**im5*).

Решение данной задачи должно начинаться с решения задачи ***Z**im1*. К задачам ***Z**im2* и (***Z**im3*) можно приступать только после завершения работы над задачей ***Z**im1*. Эти задачи допускают параллельное их выполнение.

Таким образом, порядок выполнения этих задач описывается шаблоном управления потоком работ ***Parallel Split***, представляющим собой разделение нити управления потоками работ.

Приступить к задаче ***Z**im4* можно только после завершения работы над задачами ***Z**im2* и ***Z**im3*. Данная ситуация описывается шаблоном управления потоком работ ***Synchronization***. И только после завершения работы над задачей ***Z**im4*; можно приступать к выполнению задачи ***Z**im5*, что описывается шаблоном Sequence.

Данные задачи были погружены в QA-память WIQA.

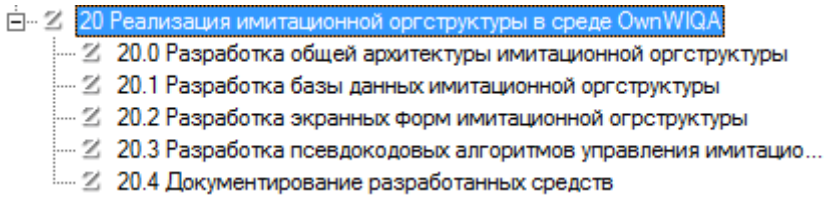


Рис. П12. Погруженные задачи имитационной оргструктуры

После этого для новых задач были сформированы поручения. Форма создания поручений содержит поле «Код проверки», предназначенный для вписывания строки псевдокода, позволяющей определить, можно ли начать выполнение данной задачи.

Для того, чтобы задача была доступной к выполнению, это поле должно содержать единицу, код, её возвращающий или присваивание значения «1» переменной &TaskEnabled&.

Задачи ***Z**im* и ***Z**im1* доступны для начала выполнения вне зависимости от результатов выполнения других задач. Код проверки для них был установлен в 1.

Для задач Z_{im2} и Z_{im3}, которые выполняются после завершения задачи был установлен следующий код:

&TaskEnabled& := TaskFinishedByName ("Разработка общей архитектуры имитационной")

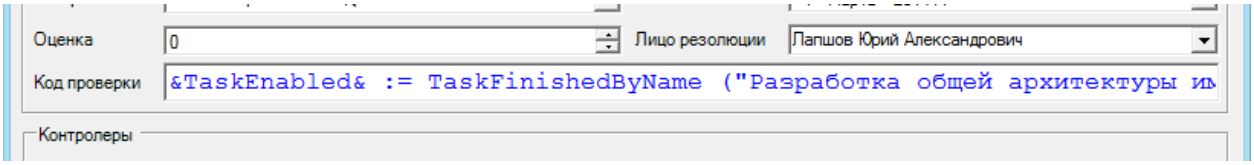


Рис. П13. Код проверки

Здесь была использована функция ***TaskFinishedByName***, определяющая, отмечена ли задача как завершенная в Kanban-таблице по началу её имени.

Для задачи Z_{im4} был установлен синхронизирующий выполнение задач Z_{im2} и Z_{im3} код:

IF TaskFinishedByName ("Разработка базы") == 1 AND TaskFinishedByName ("Разработка экранных") == 1 THEN &TaskEnabled& := 1 ELSE &TaskEnabled& := 0

Для начала выполнения задачи Z_{im5} необходимо проверить на выполнение задачу Z_{im4}:

IF TaskFinishedByName ("Разработка псевдокодовых алгоритмов") == 1 THEN &TaskEnabled& := 1

Задачи, недоступные для выполнения, отмечены в Kanban-таблице как «Выполнение: Не разрешено» и в меню выбора задачи отмечены как неактивные:

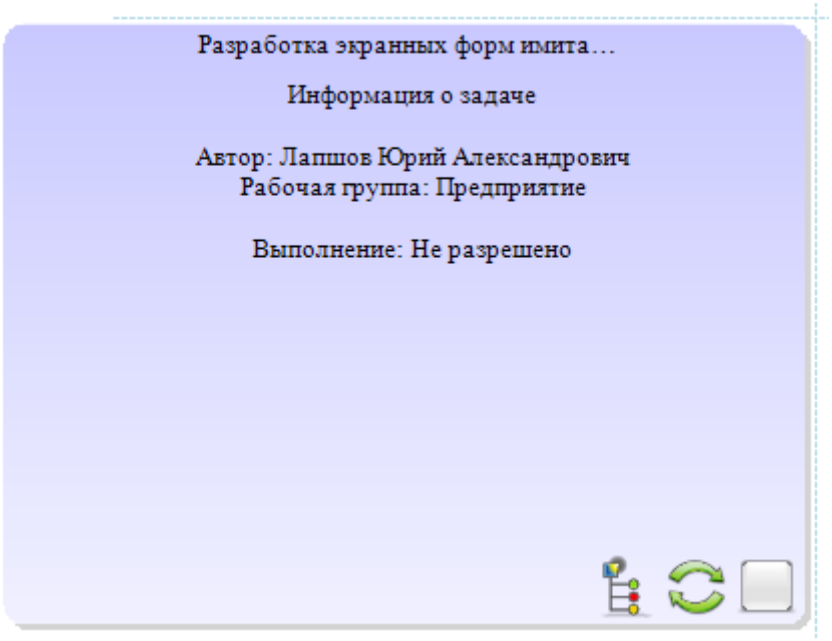


Рис. П14. Неразрешенная задача

Из выполнения данного эксперимента были сделаны следующие выводы:

1. Применение средств ПКУ в персональной работе позволяет визуально контролировать очередь собственных задач для выполнения, управлять их приоритетами. Следовательно, гипотеза ***H2*** подтверждена.
2. В процессе работы с инструментальными средствами ПКУ сбоев не происходило, следовательно, гипотеза ***H3*** подтверждена.

Эксперимент на потоках работ по решению задач олимпиады по программированию студентами

Цели данного эксперимента следующие:

1. Определить возможности альтернативного применения реализованных средств ПКУ, проверив следующую гипотезу:

H4. Разработанные инструментальные средства ПКУ можно использовать в целях получения сравнительных характеристик исполнителей тестового задания

2. Произвести их нагрузочное тестирование разработанных

программных средств.

В качестве входных параметров выступает коллектив из **102 студентов**, что много больше максимально рекомендуемого методологией Scrum размера проектной команды, составляющего 9 человек. Студенты выполняют задания в рамках турнира по алгоритмическому программированию, проходящего в течение **4 дней**, а также входным параметром выступает **17 задач**, решаемых студентами и **результаты их работы над задачами**.

В качестве **выходных параметров** выступает **время, затраченное на загрузку картотечной визуализации**, а также – метрическая характеристика скорости работы участника группы **IV**. В том случае, если оценивание трудоемкости задач проводится одним человеком, из оценки объема работ и скорости её выполнения исключается фактор индивидуального подхода к оценке объема работы каждым участником, что позволяет использовать метрику **IV** в качестве сравнительной характеристики.

В качестве плана проведения данного эксперимента выступает следующая методика, исполнителем которой является автор данной диссертационной работы:

1. Сформировать проект, включающий в себя набор задач турнира;
2. Добавить в проект задачи ПКУ;
3. Сформировать организационную структуру, состоящую из студенческих групп и студентов;
4. Сгенерировать для студентов одной студенческой группы поручения на выполнение задач, и, соответственно, Kanban-карточки этих задач;
5. Произвести замер времени, затрачиваемого на загрузку средства картотечной визуализации;
6. Сгенерировать для каждого оставшегося студента поручение на выполнение задачи, и, соответственно, Kanban-карточку;
7. Произвести замер времени, затрачиваемого на загрузку средства

картотечной визуализации;

8. Рассчитать Scrum-метрику **IV** для трех студентов, выполнивших различное количество задач;

Выполнение данного эксперимента было начато с создания проекта «Олимпиада ВТ УлГТУ», в который были помещены задачи олимпиады. Далее в проект были добавлены задачи олимпиады и задачи средств ПКУ:

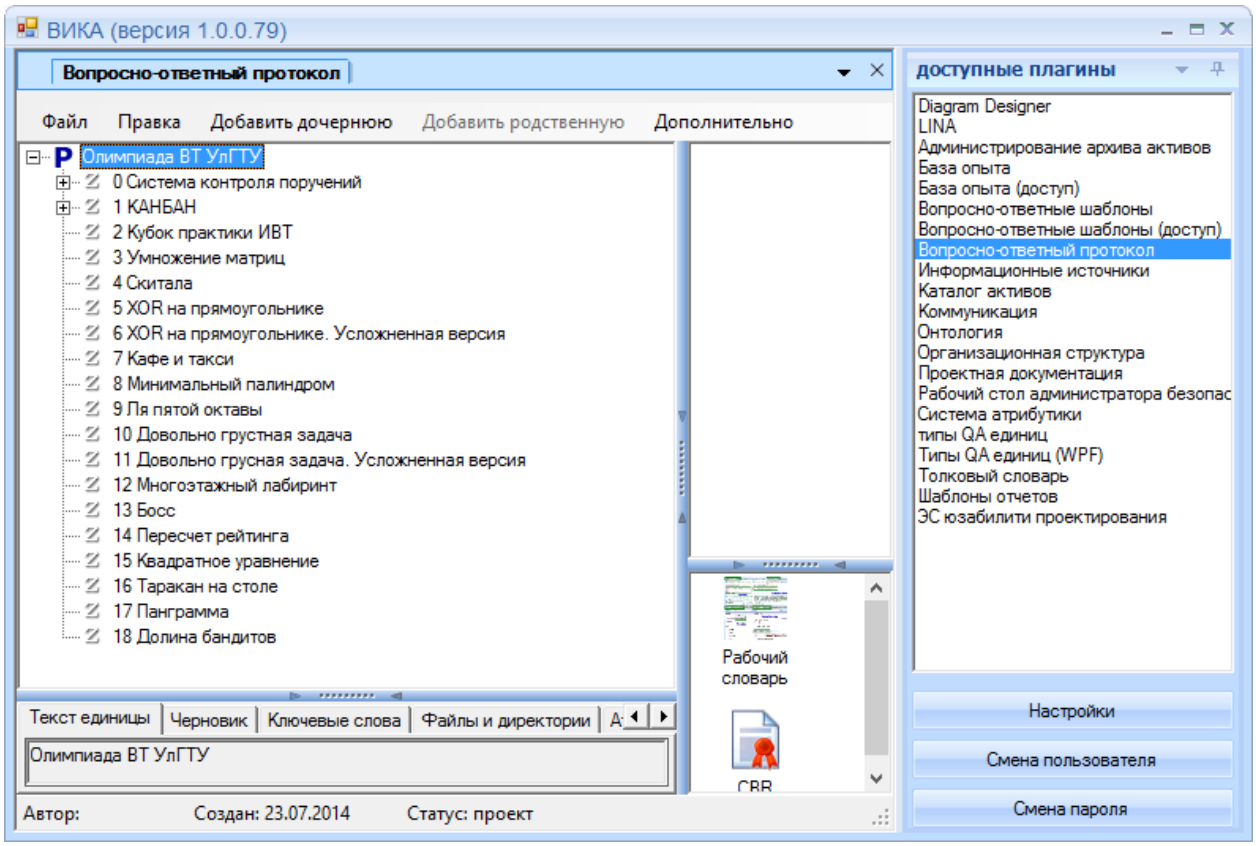


Рис. П15. Задачи олимпиады

Затем была сгенерирована организационная структура, включающая 102 студентов в 6 студенческих группах.

Для массового создания поручений и Kanban-карточек был написан псевдокодовый скрипт, автоматизирующий выполнение данных действий.

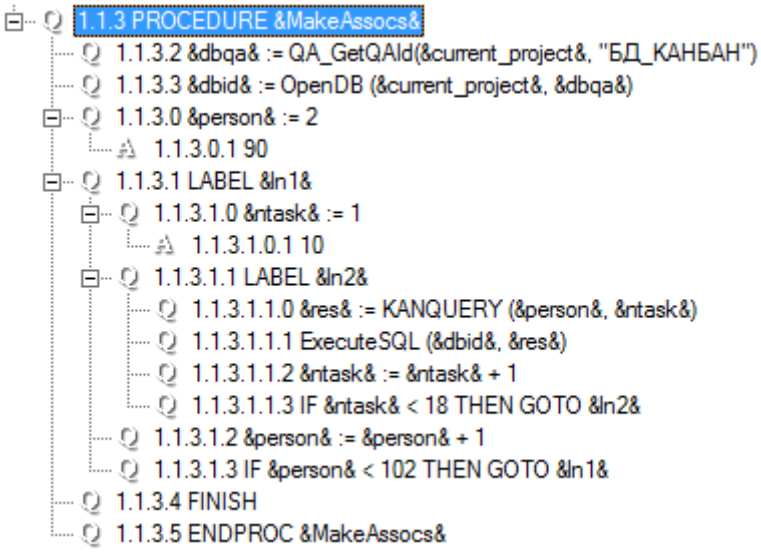


Рис. П16. Скрипт автоматизации создания карточек задач

Далее, после выполнения данный скрипта для добавления задач студентов одной студенческой группы, был произведен замер времени открытия средства картотечной визуализации. Для группы из **25** студентов было создано **425** карточек. Время загрузки средства визуализации t_1 составило **4** секунды.

После этого скрипт был выполнен для генерирования остальных карточек. После этого общее количество карточек составило **1734.** время загрузки средства визуализации t_2 составило **7** секунд, что является приемлемым результатом. Работа средств ПКУ при этом оставалась устойчивой.

Далее были определены трудоемкости задач. Каждая задача была оценена в **21** StoryPoints и суммарная трудоемкость составила **357** StoryPoints. Далее были рассчитаны следующие показатели:

Кондратьев Евгений Валерьевич – выполнил за **4** дня **17 задач из 17**. Его скорость IV составила **89,75** StoryPoints/сут.

Петров Виталий Глебович – выполнил за 4 дня **10 задач из 17**. Его скорость IV составила **52,5** StoryPoints/сут.

Сердитов Иван Александрович выполнил за **4** дня **6 задач из 17**. Его скорость IV составила **31,5** StoryPoints/сут.

Из выполнения данного эксперимента были сделаны следующие выводы:

1. Разработанные средства ПКУ можно использовать в целях сравнения производительности людей, выполняющих тестовые задания, рассчитав метрику индивидуальной скорости IV в том случае, если оценка задач для всех участников была константой. Следовательно, гипотеза H4 подтверждается.
2. В процессе нагрузочного тестирования в инструментальные средства ПКУ был погружен набор задач и исполняющий их коллектив, дающий в сумме 1734 карточки. При этом инструментальные средства показали устойчивую работу и приемлемое время загрузки. Следовательно, нагрузочное тестирование является пройденным.