

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Пензенский государственный университет» (ПГУ)

И. Ю. Балашова, Д. В. Такташкин

Современные информационные технологии в проектировании программных систем и комплексов

Учебное пособие

Под редакцией П. П. Макарычева

Пенза
Издательство ПГУ
2019

УДК 004.9

Б20

Р е ц е н з е н т ы :

доктор технических наук,
профессор кафедры «Вычислительная техника»
Пензенского государственного университета

С. А. Зинкин;

кандидат технических наук,
начальник отдела управления системных исследований
НТП «Криптософт» (г. Пенза)

А. Ю. Афонин

Балашова, И. Ю.

Б20

Современные информационные технологии в проектировании программных систем и комплексов : учеб. пособие / И. Ю. Балашова, Д. В. Такташкин ; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с.

ISBN 978-5-907185-99-9

Рассматриваются основные теоретические вопросы и терминологический аппарат современных информационных технологий в сфере проектирования программных систем и комплексов. Анализируются теоретические аспекты построения моделей информационных систем и комплексов, а также технологический процесс разработки программных систем. Приведены структура, базовые виды информационных технологий для процесса разработки программного обеспечения. Отдельно выделены информационные технологии IT-разработчика с раскрытием основных инструментальных средств проектирования и разработки информационных систем.

Издание подготовлено на кафедре «Математическое обеспечение и применение ЭВМ» ПГУ и предназначено для обучающихся по направлениям подготовки: 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика», 09.03.04 «Программная инженерия».

УДК 004.9

ISBN 978-5-907185-99-9

© Пензенский государственный
университет, 2019

ОГЛАВЛЕНИЕ

Введение	6
Г л а в а 1. Основные понятия информационных технологий	7
1.1. Основные понятия и определения.....	7
1.2. Исторические аспекты развития информационных технологий для проектирования программных систем и комплексов	9
Г л а в а 2. Разработка функциональных требований к информационным системам	12
2.1. Назначение диаграммы вариантов использования	12
2.2. Контекст системы	13
2.3. Актеры	13
2.4. Варианты использования	14
2.5. Интерфейсы.....	15
2.6. Примечания	16
2.7. Отношения	17
2.8. Отношение ассоциации.....	17
2.9. Отношение расширения	18
2.10. Отношение включения.....	19
2.11. Отношение обобщения.....	20
2.12. Программное обеспечение для анализа функциональных требований	22
Контрольные вопросы.....	24
Задания для самостоятельного выполнения.....	24
Г л а в а 3. Разработка алгоритмов работы программных модулей информационной системы.....	26
3.1. Назначение диаграммы деятельности.....	26
3.2. Начальное состояние	26
3.3. Конечное состояние.....	27
3.4. Состояние действия	27
3.5. Простой переход	28
3.6. Событие	29
3.7. Сторожевое условие	30
3.8. Выражение действия	30
3.9. Ветвление	31
3.10. Параллельные процессы	32
3.11. Дорожки.....	32
3.12. Объекты	33

3.13. Программное обеспечение для проектирования алгоритмов работы информационной системы.....	35
Контрольные вопросы	38
Задания для самостоятельного выполнения	38
Г л а в а 4. Системный подход к проектированию информационных систем	39
4.1. Концепция IDEF0.....	39
4.2. Принципы построения модели IDEF0.....	40
4.3. Синтаксис IDEF0.....	41
4.4. Функциональный блок	42
4.5. Интерфейсная дуга.....	42
4.6. Семантика функциональных блоков и интерфейсных дуг.....	44
4.7. Имена и метки	45
4.8. Диаграммы IDEF0	47
4.9. Контекстная диаграмма верхнего уровня	47
4.10. Дочерняя диаграмма	48
4.11. Родительская диаграмма	49
4.12. Рекомендации по построению модели IDEF0	51
4.13. Программное обеспечение для построения диаграммы IDEF0	55
Контрольные вопросы	58
Задания для самостоятельного выполнения	58
Г л а в а 5. Проектирование баз данных с использованием методологии IDEF1X	59
5.1. Создание модели данных	59
5.2. Системный анализ предметной области	60
5.3. Инфологическое проектирование.....	61
5.4. Даталогическое проектирование	61
5.5. Сущность	62
5.6. Атрибут сущности	63
5.7. Ключ сущности	64
5.8. Связь.....	64
5.9. Физическое проектирование	66
5.10. Программное обеспечение для проектирования базы данных по методологии IDEF1X.....	67
Контрольные вопросы	70
Задания для самостоятельного выполнения	71

Г л а в а 6. Планирование разработки информационной системы	72
6.1. Общие принципы управления проектом	72
6.2. Структурное планирование.....	74
6.3. Календарное планирование	79
6.4. Оперативное управление	85
6.5. Программное обеспечение для планирования разработки информационной системы	86
Контрольные вопросы.....	89
Задания для самостоятельного выполнения.....	90
Заключение.....	91
Библиографический список	92
П р и л о ж е н и е А. Примеры выполнения диаграмм.....	94
П р и л о ж е н и е Б. Примеры предметной области для выполнения самостоятельной работы.....	105

ВВЕДЕНИЕ

Информатизация как процесс перехода к информационному обществу сопровождается возникновением новых и интенсивным развитием существующих информационных технологий. Информация превращается в коммерческий ресурс, способствуя получению прибыли при внедрении информационных технологий во многие сферы человеческой деятельности. Возникают информационная экономика, новая информационная инфраструктура промышленности и социальной сферы, формируется информационная культура.

Зарубежный и российский опыт внедрения информационных технологий подтверждает высокую экономическую эффективность для многих сфер применения. Успешность и прибыльность компании полностью зависят в том числе и от уровня развития IT-технологий, скорости и качества обработки информации, обоснованности и взвешенности принимаемых решений. Перспективы развития информационных технологий связаны с подготовкой специалистов в данной сфере. В данном учебном пособии излагаются теоретические основы информационных технологий как комплексной научной и инженерной дисциплины.

Глава 1

ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

1.1. Основные понятия и определения

Понятие «информационная технология» базируется на таком основополагающем понятии, как «информация». Под информацией понимают сведения (сообщения, данные) независимо от формы их представления. Информация может существовать в форме сигнала (светового, звукового, электрического и других), информационного сообщения (текстового, графического, речевого, визуального, аудиовизуального и других), формализованных данных (символов, показателей, параметров и др.). Информация, зафиксированная на материальном носителе любым доступным человеку способом, является документированной.

Информационные технологии – это процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов. Основная цель автоматизированной информационной технологии – получать посредством переработки первичных данных информацию нового качества, на основе которой вырабатываются оптимальные управленческие решения. Основная цель информационной технологии достигается за счёт:

- интеграции информации;
- обеспечения актуальности и непротиворечивости данных;
- использования современных технических средств для организации процесса внедрения и функционирования качественно новых форм информационной поддержки деятельности аппарата управления.

Информационная технология справляется с существенным увеличением объёмов перерабатываемой информации, ведёт к сокращению сроков её обработки и является наиболее важной составляющей процесса использования информационных ресурсов в управлении.

Эффективность функционирования информационной технологии определяется её основными свойствами, представленными ниже.

1. *Целесообразность.* Состоит в повышении эффективности производства за счёт внедрения современных средств вычислительной техники, распределённых баз данных, различных вычислительных сетей, что позволяет обеспечить эффективную циркуляцию и переработку информации.

2. *Наличие компонентов и структуры.* В состав информационной технологии должны входить:

1) комплекс технических средств (КТС), состоящий из средств вычислительной, коммуникационной и организационной техники;

2) программные средства, состоящие из общего (системного), прикладного (программ для решения функциональных задач специалистов) и инструментального программного обеспечения (алгоритмических языков, систем программирования, языков спецификаций, технологии программирования и т.д.);

3) система организационно-методического обеспечения, включающая нормативно-методические и инструктивные материалы по организации работы управленческого и технического персонала конкретной ИТ.

3. *Взаимодействие с внешней средой.* Предполагает организацию взаимосвязи информационной технологии с объектами управления, внешними предприятиями, организациями, включая потребителей и поставщиков продукции, финансово-кредитные органы и т.д. Взаимодействие информационных технологий различных объектов организуется посредством программных и технических средств автоматизации.

4. *Целостность.* Информационная технология является целостной системой, способной решать задачи, не свойственные ни одному из её компонентов.

5. *Развитие во времени.* Это обеспечение динамичности развития информационной технологии, возможность её модернизации и модификации, изменение структуры, включение новых компонентов, возможность решения новых задач и т.д.

Информационная система (ИС) представляет собой совокупность информационных технологий и технических средств. Основные функции ИС:

- сбор и хранение больших объемов информации;
- обработка информации в ходе решения задач;
- отображение информации в виде, удобном для изучения и принятия решений.

Основными составляющими любой ИС являются, как минимум:

- база данных как совокупность взаимосвязанных упорядоченных определенным образом данных;

- программные модули, предназначенные для обработки данных;

- пользовательский интерфейс.

Проектирование информационных систем – это упорядоченная совокупность методологий и средств создания или модернизации информационных систем.

Жизненный цикл информационной системы – развитие рассматриваемой системы во времени, начиная от замысла и кончая списанием.

Архитектура информационных систем – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

Методология проектирования информационных систем – это совокупность принципов проектирования (моделирования), выраженная в определенной концепции.

1.2. Исторические аспекты развития информационных технологий для проектирования программных систем и комплексов

Середина прошлого столетия ознаменовалась началом активного развития информационных технологий. С появлением вычислительной техники обработка больших объемов информации и автоматизация основных производственных процессов и органов управления на всех уровнях становятся наиважнейшей задачей для обеспечения военного превосходства наиболее развитых государств и конкурентных преимуществ коммерческих компаний. Разработчики национальных и крупномасштабных информационных систем стали первыми осознавать необходимость создания специальных средств проектирования и моделирования информационных процессов, позволяющих сделать их работу более эффективной и сократить не только сроки создания информационных систем, но и минимизировать ошибки. Ошибки и неточности встречаются постоянно, чем раньше они диагностируются и локализуются, тем меньше стоимость переделки. Известно, что стоимость выявления и устранения ошибки на стадии проектирования в два раза обходится дороже, на стадии тестирования информационной системы – в десять раз, а на стадии эксплуатации – в сто раз, чем на стадии анализа информационных процессов и разработки технического задания.

При создании сложных информационных систем зачастую очень трудно понять требования заказчика. Они могут быть сформулированы некорректно, а в процессе анализа конкретных информационных процессов даже измениться. Поэтому появление

методологий современного проектирования и моделирования информационных систем было насущной задачей, над которой работали специалисты разных стран.

Появление автоматизированных систем управления (АСУ) в шестидесятых годах прошлого столетия определялось получением начальных знаний и опыта их разработки и внедрения. Анализировались все успехи и неудачи создания первых АСУ, но бесспорным было сокращение времени обработки информации, производственных и управленческих затрат и, как следствие, персонала. Опыт зарубежных компаний по разработке и внедрению корпоративных информационных систем свидетельствует о появлении программ, в первую очередь связанных с автоматизацией учетных функций бухгалтерий, отдела кадров и складов. И намного позже появляются другие автоматизированные системы управления производством, логистикой, взаимоотношениями с клиентами и поставщиками. На последнем этапе разрабатываются информационные системы управления всей компанией, позволяющие полностью перейти к электронному документообороту и автоматизировать все сферы деятельности фирмы. С появлением персональных компьютеров происходит децентрализация процессов управления, все чаще внедряются модули с распределенными системами обработки информации.

На следующем этапе, в семидесятых годах прошлого столетия, пришло понимание, что информация – стратегический ресурс любой компании, который необходимо грамотно использовать. В то же время главными потребителями информации являются руководители. Идеи использования распределенных систем не находят пока применения из-за отсутствия компактной вычислительной техники, которая появится позднее. В компаниях и организациях создаются информационные отделы и службы, вычислительные центры и лаборатории.

Восьмидесятые годы прошлого столетия характеризуются появлением специализированных методологий проектирования информационных систем и CASE-средств. На их основе разрабатываются первые программные средства, а персональные компьютеры позволяют приступить к созданию децентрализованных информационных систем. Различные персональные компьютеры объединяются в локальную сеть. Этот период характеризуется интеграцией информационных систем и появлением различных концепций управления ими на единой методологической основе.

Девяностые годы прошлого столетия стали триумфом персональных компьютеров. Невысокая стоимость и компактные размеры

сделали их чрезвычайно популярными и общедоступными для индивидуализации использования при решении управленческих задач. Разрабатываются корпоративные информационные системы, реализующие принципы распределенной обработки данных. Становится возможной автоматизация всех отделов и служб компаний, а не только бухгалтерии. Появляются системы электронного документооборота, в том числе для предприятий с развитой филиальной сетью в разных городах и регионах. Сокращаются сроки обработки данных, производственных, складских и прочих управленческих отчетов.

В 1970–1980-х гг. информационные системы интегрировались в комплексные АСУ, решающие задачи автоматизированного проектирования новых изделий, технологической подготовки производства и автоматизации организационного управления предприятием. В США Дуглас Т. Росс (SoftTech, Inc.) разработал язык АПТ для работы станков с ЧПУ, который в середине шестидесятих годов прошлого столетия положил начало разработкам графических языков моделирования. А в 1969 г. им же предложена методология SADT (IDEF0) для моделирования информационных систем в рамках программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой департаментом Военно-Воздушных Сил США.

К концу XX в. было разработано несколько десятков методов моделирования сложных систем. Они все были разные по функциональным возможностям, но во многом имели схожие подходы к анализу и описанию предметной области. Возникла необходимость объединения удачных решений в одну методику, которая устраивала большую часть разработчиков информационных систем. В результате этих процессов был разработан унифицированный язык моделирования UML. В настоящее время методологии и средства моделирования информационных процессов, процессно-ориентированные методы анализа и проектирования информационных систем широко представлены как в России, так и в большинстве стран мира.

Глава 2

РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ К ИНФОРМАЦИОННЫМ СИСТЕМАМ

2.1. Назначение диаграммы вариантов использования

Диаграммы вариантов использования в UML описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

- определить общие границы и контекст моделируемой предметной области;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть диаграммы вариантов использования состоит в следующем. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования. При этом актером называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру. Диаграмма вариантов использования может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Моделирование вариантов использования выполняется по следующему алгоритму:

- устанавливаются границы моделируемой системы;
- выявляются актеры;
- выявляются варианты использования;
- предыдущие шаги повторяются, пока варианты использования, актеры и границы системы не стабилизируются.

Результатом этой деятельности является модель вариантов использования, в которой присутствуют четыре компонента:

- граница системы – прямоугольник, очерчивающий варианты использования для обозначения границ моделируемой системы (контекст системы);
- актер – это роль объекта вне системы, который напрямую взаимодействует с ее составными частями;
- варианты использования – то, что актеры могут делать с системой;
- отношения – значимые отношения между элементами моделируемой системы.

2.2. Контекст системы

Контекст системы отделяет систему от внешнего мира, т.е. определяет, что является частью системы (находится внутри границ системы) и что находится вне системы (вне ее границ). Контекст изображается в виде прямоугольника с именем системы. Актеры размещаются вне границ блока, а варианты использования – внутри. В начале моделирования вариантов использования имеется лишь предварительное представление о том, где находятся границы системы. По мере выявления актеров и вариантов использования контекст системы обретает все более четкие очертания.

2.3. Актеры

Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка человечка, под которой записывается имя актера (рис. 2.1).

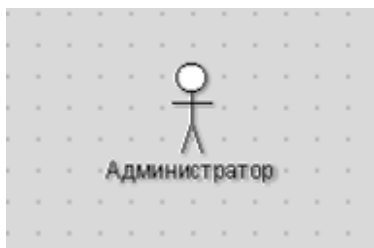


Рис. 2.1. Актер

В некоторых случаях актер может обозначаться в виде прямоугольника класса с ключевым словом «актер» и обычными составляющими элементами класса. Имена актеров должны записываться с заглавных букв и следовать рекомендациям использования имен для типов и классов модели. Примерами актеров могут быть: клиент банка, банковский служащий, продавец магазина, менеджер отдела продаж, пассажир авиарейса, водитель автомобиля, администратор гостиницы, сотовый телефон и другие сущности, имеющие отношение к концептуальной модели соответствующей предметной области.

Так как в общем случае актер всегда находится вне системы, его внутренняя структура никак не определяется. Для актера имеет значение только то, как он воспринимается со стороны системы. Актеры взаимодействуют с системой посредством обмена сообщениями с вариантами использования. Сообщение представляет собой запрос актером определенного сервиса системы и получение этого сервиса. Это взаимодействие может быть выражено посредством ассоциаций между отдельными актерами и вариантами использования или классами. Кроме этого, с актерами могут быть связаны интерфейсы, которые определяют, каким образом другие элементы модели взаимодействуют с этими актерами.

Два и более актера могут иметь общие свойства, т.е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.

2.4. Варианты использования

Вариант использования описывает поведение, демонстрируемое системой с целью получения значимого результата для одного или более актеров. Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия её внутренней структуры. В качестве такой сущности может выступать система или любой элемент модели, который обладает собственным поведением. Примерами вариантов использования могут являться следующие действия: проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение дополнительной информации о кредитоспособности клиента, отображение графической формы на экране монитора и другие действия.

Варианты использования всегда иницируются актером, и варианты использования всегда описываются с точки зрения актеров. Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами (рис. 2.2).

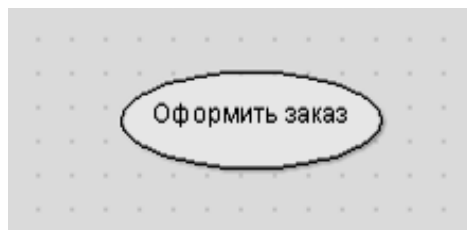


Рис. 2.2. Вариант использования

Идентификацию вариантов использования лучше всего начать со списка актеров, а затем рассмотреть, как каждый актер собирается использовать систему. С помощью такой стратегии можно получить список потенциальных вариантов использования.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемая сущность по запросу актера, т.е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу актера, представляет собой законченную неделимую последовательность действий. Это означает, что после того, как система закончит обработку запроса, она должна возвратиться в исходное состояние, чтобы быть готовой к выполнению следующих запросов.

Варианты использования могут применяться как для спецификации внешних требований к проектируемой системе, так и для спецификации функционального поведения уже существующей системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы. Кроме этого, варианты использования неявно устанавливают требования, определяющие, как актеры должны взаимодействовать с системой, чтобы иметь возможность корректно работать с предоставляемыми сервисами. Для удобства множество вариантов использования может рассматриваться как отдельный пакет.

2.5. Интерфейсы

Интерфейс служит для спецификации параметров модели, которые видимы извне, без указания их внутренней структуры.

В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов для актеров или вариантов использования.

На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя (рис. 2.3). В качестве имени может быть использовано существительное или строка текста.

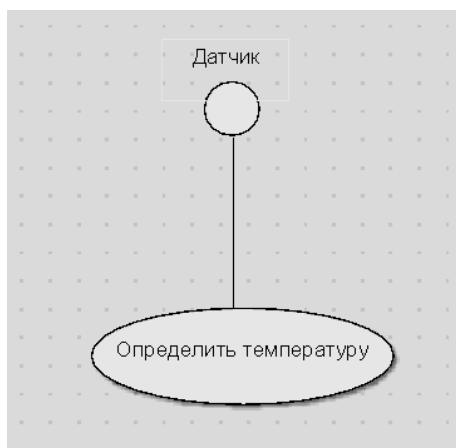


Рис. 2.3. Интерфейс

Графический символ отдельного интерфейса соединяется на диаграмме сплошной или пунктирной линией с тем вариантом использования, который его поддерживает. Сплошная линия указывает, что связанный с интерфейсом вариант использования должен реализовывать все необходимые для него сервисы. Пунктирная линия со стрелкой означает, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Таким образом, интерфейс отделяет спецификацию операций системы от их реализации и определяет общие границы проектируемой системы.

2.6. Примечания

Примечания в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть использованы комментарии

разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Графически примечания обозначаются прямоугольником с загнутым верхним правым углом (рис. 2.4). Внутри прямоугольника содержится текст самого примечания.

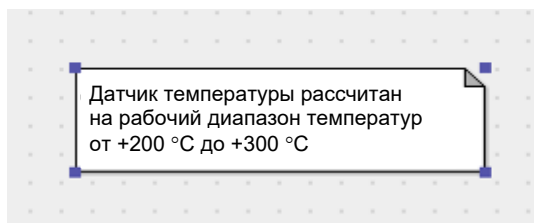


Рис. 2.4. Примечание

Если в примечании указывается ключевое слово «*constraint*», то оно является ограничением, налагаемым на соответствующий элемент модели.

2.7. Отношения

Между элементами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров актеров и вариантов использования, вариантов использования с вариантами использования, а также актеров с актерами. В языке UML существует несколько стандартных видов отношений:

- ассоциации (*association relationship*);
- расширения (*extend relationship*);
- обобщения (*generalization relationship*);
- включения (*include relationship*).

2.8. Отношение ассоциации

Применительно к диаграммам вариантов использования ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы, т.е. это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования отношение ассоциации обозначается сплошной линией между актером и вариантом использования (рис. 2.5). Эта линия может иметь условные обозначения, такие как имя и кратность.

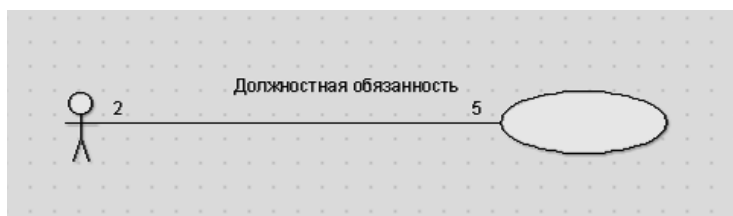


Рис. 2.5. Отношение ассоциации

Кратность (*multiplicity*) ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации, и характеризует количество экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и символа «звездочка». Если кратность отношения ассоциации не указана, то, по умолчанию, принимается ее значение, равное единице.

2.9. Отношение расширения

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров. В метамодели отношение расширения является направленным и указывает, что применительно к отдельным примерам некоторого варианта использования должны быть выполнены конкретные условия, определенные для расширения данного варианта использования. Так, если имеет место отношение расширения от варианта использования *A* к варианту использования *B*, то это означает, что свойства экземпляра варианта использования *B* могут быть дополнены благодаря наличию свойств у расширенного варианта использования *A*.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования (рис. 2.6). Данная линия со стрелкой помечается ключевым словом «*extend*» (расширяет).

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

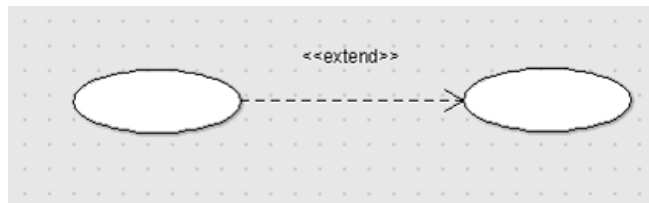


Рис. 2.6. Отношение расширения

Данное отношение включает в себя некоторое условие и ссылки на точки расширения в базовом варианте использования. Чтобы расширение имело место, должно быть выполнено определенное условие данного отношения. Ссылки на точки расширения определяют те места в базовом варианте использования, в которые должно быть помещено соответствующее расширение при выполнении условия.

Один вариант использования может быть расширением для нескольких базовых вариантов, а также иметь в качестве собственных расширений несколько других вариантов. Базовый вариант использования может дополнительно никак не зависеть от своих расширений.

Семантика отношения расширения определяется следующим образом. Если экземпляр варианта использования выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляре другого варианта использования, которая является первой из всех точек расширения исходного варианта, то проверяется условие данного отношения. Если условие выполняется, исходная последовательность действий расширяется посредством включения действий экземпляра другого варианта использования. Следует заметить, что условие отношения расширения проверяется лишь один раз при первой ссылке на точку расширения, и если оно выполняется, то все расширяющие варианты использования вставляются в базовый вариант.

2.10. Отношение включения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования.

Отношение включения, направленное от варианта использования *A* к варианту использования *B*, указывает, что каждый экземпляр варианта *A* включает в себя функциональные свойства, заданные для варианта *B*. Эти свойства специализируют поведение соответствующего варианта *A* на данной диаграмме. Графически

данное отношение обозначается пунктирной линией со стрелкой, которая помечается ключевым словом «*include*» (включает) (рис. 2.7).

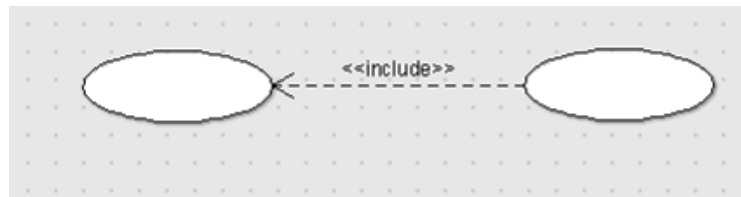


Рис. 2.7. Отношение включения

Семантика этого отношения определяется следующим образом. Когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, экземпляр первого варианта использования выполняет последовательность действий, определяющую поведение экземпляра второго варианта использования, после чего продолжает выполнение действий своего поведения. При этом предполагается, что, даже если экземпляр первого варианта использования может иметь несколько включаемых в себя экземпляров других вариантов, выполняемые ими действия должны закончиться к некоторому моменту, после которого должно быть продолжено выполнение прерванных действий экземпляра первого варианта использования в соответствии с заданным для него поведением.

Один вариант использования может быть включен в несколько других вариантов, а также включать в себя другие варианты. Включаемый вариант использования может быть не зависимым от базового варианта в том смысле, что он предоставляет ему некоторое инкапсулированное поведение, детали реализации которого скрыты и могут быть перераспределены между несколькими включаемыми вариантами использования. Более того, базовый вариант может зависеть только от результатов выполнения включаемого в него поведения, но не от структуры включаемых в него вариантов.

2.11. Отношение обобщения

Отношение обобщения служит для указания того факта, что некоторый вариант использования *A* может быть обобщен до варианта использования *B*. В этом случае вариант *A* будет являться специализацией варианта *B*. При этом *B* называется предком или родителем по отношению к *A*, а вариант *A* – потомком по отношению к варианту использования *B*. Потомок наследует все свойства

и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования (рис. 2.8).



Рис. 2.8. Отношение обобщения

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наследовать новые свойства поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Применительно к данному отношению один вариант использования может иметь несколько родительских вариантов. В этом случае реализуется множественное наследование свойств и поведения отношения предков. С другой стороны, один вариант использования может быть предком для нескольких дочерних вариантов, что соответствует таксономическому характеру отношения обобщения.

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Например, отношение обобщения от актера *A* к актеру *B* отмечает тот факт, что каждый экземпляр актера *A* является одновременно экземпляром актера *B* и обладает всеми его свойствами. В этом случае актер *B* является родителем по отношению к актеру *A*, а актер *A* – потомком актера *B*. При этом актер *A* обладает способностью играть такое же множество ролей, что и актер *B*. Графически данное отношение также обозначается стрелкой обобщения.

2.12. Программное обеспечение для анализа функциональных требований

StarUML

StarUML – коммерческая программная платформа моделирования, которая поддерживает UML (рис. 2.9). Она основана на версии UML 1.4 и поддерживает нотацию UML версии 2.0 и одиннадцать различных типов диаграмм. StarUML активно поддерживает подход MDA (Архитектура, управляемая моделью) и концепцию профилей UML. Данная программная платформа позволяет очень гибко настраивать окружения пользователя и имеет высокую степень расширяемости в том, что касается его функциональных возможностей.

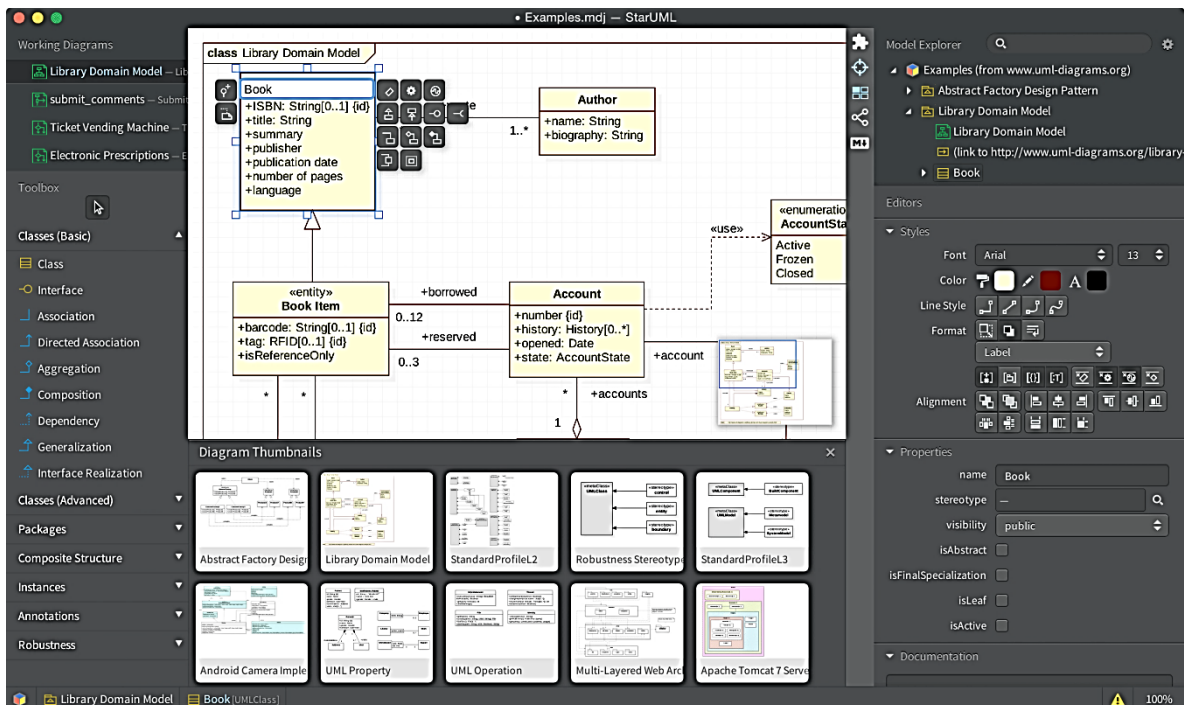


Рис. 2.9. Интерфейс StarUML

StarUML предоставляет максимальные возможности адаптации к требованиям пользователя, предлагая настройку параметров, которые определяют методологию создания программного обеспечения, проектную платформу и язык разработки.

В качестве основных преимуществ использования StarUML можно выделить следующие:

- генерация кода в языки: C#, Java, C++;
- поддержка работы с фреймворками;
- удобный графический редактор;
- полное соответствие стандарту UML 2.0;

- возможность расширения функционала с помощью дополнений;
- экспорт документации в форматы: DOC, PPT, TXT, XLS и др.;
- поддержка паттернов;
- импорт проектов Rational Rose.

ArgoUML

ArgoUML – средство UML моделирования (рис. 2.10). ArgoUML является открытым программным обеспечением и распространяется под лицензией BSD. ArgoUML полностью написан на языке программирования Java, и для работы ему подходит любая операционная система с установленной Java 2 JRE или JDK версии 1.4 или выше.

ArgoUML предоставляет ряд возможностей по редактированию и генерации кода с интуитивно-понятным интерфейсом. Функциональность ArgoUML включает в себя:

- поддержку спецификаций UML 1.3, 1.4, XMI 1.0, 1.1, 1.2;
- 9 видов диаграмм UML (диаграммы классов, состояний, кооперации, последовательности, деятельности, прецедентов, объектов, компонентов, развёртывания);
- поддержку OCL для классов;
- генерацию исходного кода Java, C++, C# и PHP;
- обратный инжиниринг из исходного кода и байткода Java;
- автоматическую верификацию модели UML (*design critics*).

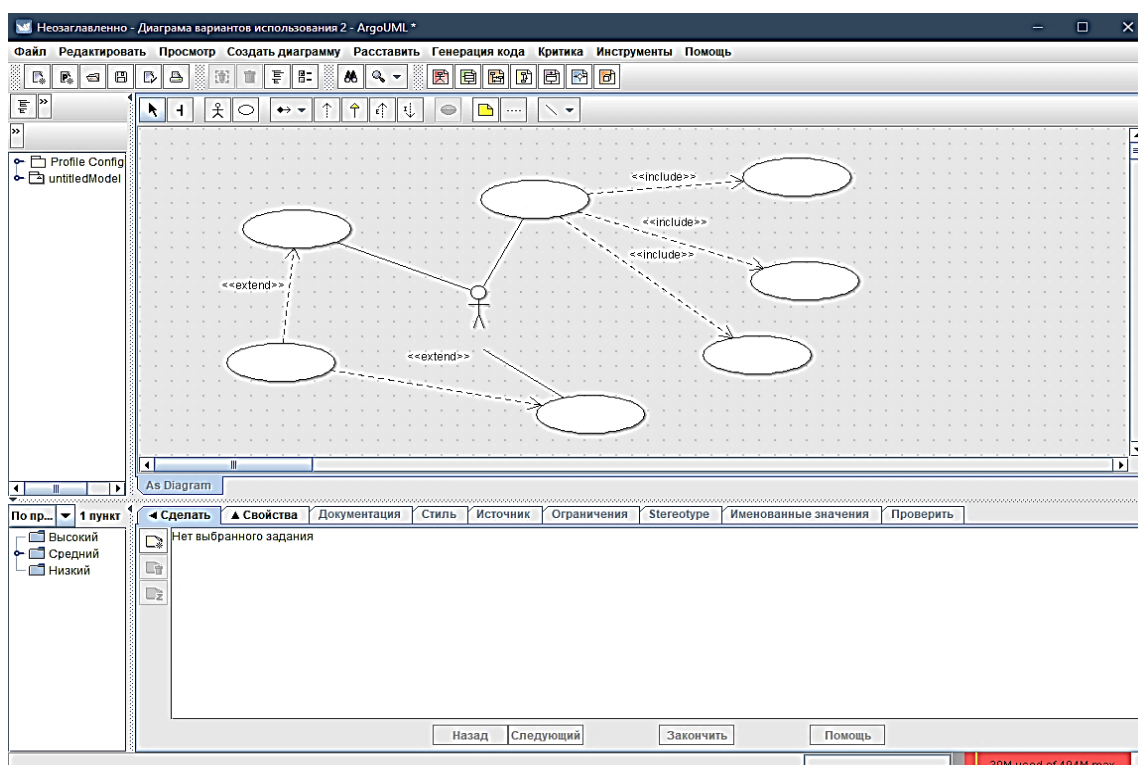


Рис. 2.10. Интерфейс ArgoUML

В качестве главных недостатков использования данного программного продукта можно выделить:

- отсутствие функции обратного проектирования (нет возможности создавать модели из имеющегося кода на Java);
- отсутствие поддержки некоторых методик, таких как Booch или Fusion;
- нет импорта файлов, созданных в других пакетах для работы с UML.

Контрольные вопросы

1. Область применения диаграммы вариантов использования.
2. С помощью чего отделяется система от внешнего мира и определяются части системы?
3. Какие способы изображения актеров вы знаете?
4. В какие отношения могут вступать актеры между собой?
5. Что описывает вариант использования?
6. Каково назначение интерфейса на диаграмме вариантов использования?
7. Что заключается в отношении ассоциации?
8. В чем состоит смысл отношений включения и расширения?
9. В чем заключается смысл отношения обобщения?
10. Перечислите известные вам причины использования вариантов использования.

Задания для самостоятельного выполнения

Требуется разработать диаграмму вариантов использования для некоторой предметной области. Требования к содержанию диаграммы:

- не более двух актеров;
- к каждому актеру, минимум, три варианта использования;
- к каждому из основных вариантов использования, минимум, три отношения включения;
- к каждому из основных вариантов использования, минимум, одно отношение расширения;
- минимум, одно отношение обобщения;
- минимум, один интерфейс;

- минимум, два примечания;
- диаграмма обязательно должна отражать специфику предметной области.

Пример диаграммы вариантов использования приведен в прил. А. Примеры предметной области приведены в прил. Б.

Глава 3

РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПРОГРАММНЫХ МОДУЛЕЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Назначение диаграммы деятельности

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций, когда необходимо представить алгоритмы их выполнения.

В контексте языка UML деятельность представляет собой совокупность отдельных вычислений, выполняемых объектом, приводящих к некоторому результату или действию. На диаграмме деятельности отображаются логика и последовательность переходов от одной деятельности к другой, а внимание аналитика фокусируется на результатах. Результат деятельности может привести к изменению состояния системы или возвращению некоторого значения.

3.2. Начальное состояние

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояние). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить, минимум, одна стрелка, соответствующая переходу (рис. 3.1).



Рис. 3.1. Начальное состояние

На самом верхнем уровне представления объекта переход из начального состояния может быть помечен событием создания

(инициализации) данного объекта. В противном случае переход никак не помечается. Если этот переход не помечен, то он является первым переходом в следующее за ним состояние.

3.3. Конечное состояние

Конечное состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояние). В этом состоянии объект будет находиться непосредственно перед уничтожением после завершения потока управления.

Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность, в которую может входить, минимум, одна стрелка, соответствующая переходу (рис. 3.2).

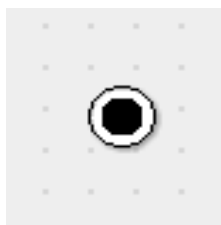


Рис. 3.2. Конечное состояние

Конечное состояние служит для указания на диаграмме графической области, в которой завершается процесс изменения состояний или жизненный цикл данного объекта.

3.4. Состояние действия

Состояние действия является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения алгоритма (процедуры) или потока управления.

Графически состояние действия изображается прямоугольником с закругленными углами (рис. 3.3). Любое состояние действия должно иметь, минимум, одну стрелку входящего перехода и, минимум, одну стрелку, соответствующую исходящему переходу.

Внутри этого изображения записывается выражение действия, которое должно быть уникальным в пределах одной диаграммы деятельности.

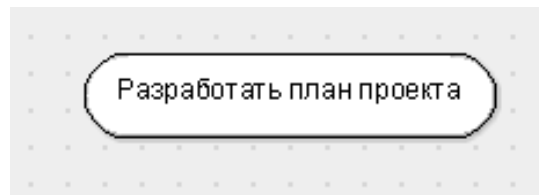


Рис. 3.3. Состояние действия

Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования (например, оплатить счёт,

$A = B + C$ или `int LD = (int)Math.Round(D * M / 20.0);`).

Никаких дополнительных или неявных ограничений при записи действий не накладывается. Рекомендуются в качестве имени простого действия использовать глагол с пояснительными словами. Если же действие может быть представлено в некотором формальном виде, то целесообразно записать его на том языке программирования, на котором предполагается реализовывать конкретный проект.

3.5. Простой переход

Простой переход представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния объекта другим. На диаграмме деятельности переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние (рис. 3.4).

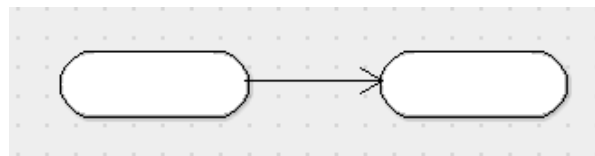


Рис. 3.4. Простой переход

Если пребывание моделируемого объекта в первом состоянии сопровождается выполнением некоторых действий, то переход во второе состояние будет возможен только после завершения этих действий и, возможно, после выполнения некоторых дополнительных условий, называемых сторожевыми условиями.

3.6. Событие

Событие является самостоятельным элементом языка UML. Формально событие представляет собой спецификацию некоторого факта, имеющего место в пространстве и во времени. Про события говорят, что они «происходят», при этом отдельные события должны быть упорядочены во времени. После наступления некоторого события уже нельзя вернуться к предыдущим событиям, если такая возможность не предусмотрена явно в модели.

Семантика понятия события фиксирует внимание на внешних проявлениях качественных изменений, происходящих при переходе моделируемого объекта из состояния в состояние. В языке UML события играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий.

Имя события идентифицирует каждый отдельный переход на диаграмме деятельности и может содержать строку текста, начинающуюся со строчной буквы (рис. 3.5).

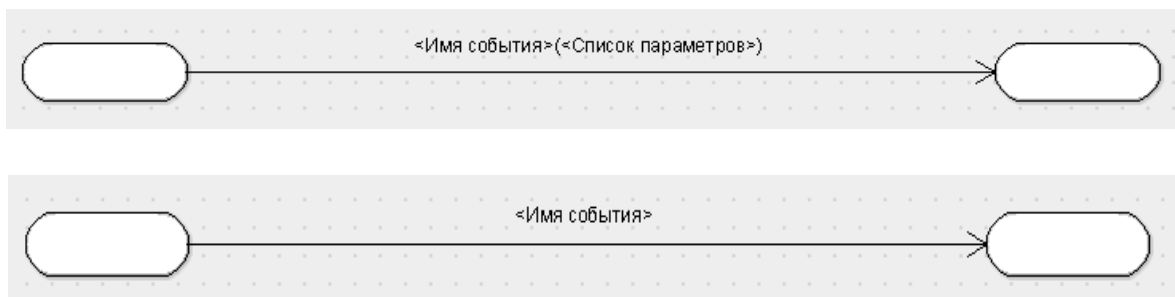


Рис. 3.5. Обозначение события

В этом случае принято считать переход триггерным, т.е. таким, который специфицирует событие-триггер. Если рядом со стрелкой перехода не указана никакая строка текста, то соответствующий переход является нетриггерным, и в этом случае из контекста диаграммы деятельности должно следовать, после окончания какой деятельности он выполняется. После имени события могут следовать круглые скобки для явного задания параметров соответствующего события-триггера. В общем случае список параметров содержит целый ряд отдельных параметров, разделенных символом «,». Если таких параметров нет, то список параметров со скобками может отсутствовать.

3.7. Сторожевое условие

Сторожевое условие, если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское (логическое) выражение (рис. 3.6).

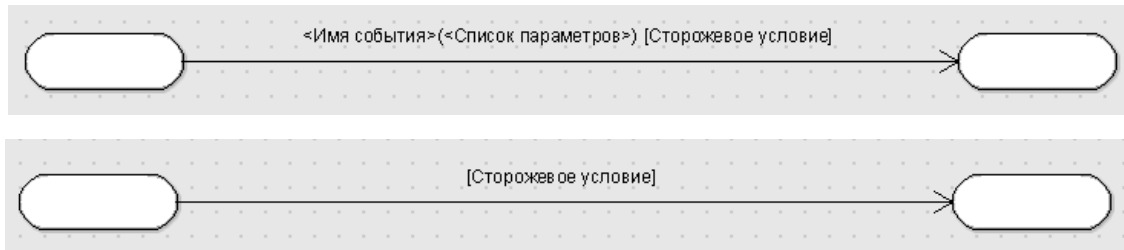


Рис. 3.6. Сторожевое условие

Из контекста диаграммы деятельности должна явно следовать семантика этого выражения, а для записи выражения может использоваться синтаксис языка объектных ограничений.

Введение для перехода сторожевого условия позволяет явно специфицировать семантику его срабатывания. Однако вычисление истинности сторожевого условия происходит только после возникновения ассоциированного с ним события, инициирующего соответствующий переход.

В общем случае из одного состояния может быть несколько переходов с одним и тем же событием-триггером. При этом никакие два сторожевых условия не должны одновременно принимать значение «истина». Каждое из сторожевых условий необходимо вычислять всякий раз при наступлении соответствующего события-триггера.

Примером события-триггера может служить разрыв телефонного соединения с провайдером интернет-услуг после окончания загрузки электронной почты клиентской почтовой программой (при удаленном доступе к интернету). В этом случае сторожевое условие есть не что иное, как ответ на вопрос: «Пуст ли почтовый ящик клиента на сервере провайдера?». В случае положительного ответа – «истина» – следует отключить соединение с провайдером, что и делает автоматически почтовая программа-клиент. В случае отрицательного ответа – «ложь» – следует оставаться в состоянии загрузки почты и не разрывать телефонное соединение.

3.8. Выражение действия

Выражение действия выполняется только при срабатывании перехода. Оно представляет собой атомарную операцию, выполняемую

сразу после срабатывания соответствующего перехода до начала каких бы то ни было действий в целевом состоянии. Атомарность действия означает, что оно не может быть прервано никаким другим действием до тех пор, пока не закончится его выполнение. Данное действие может влиять как на сам объект, так и на его окружение, если это с очевидностью следует из контекста модели (рис. 3.7).

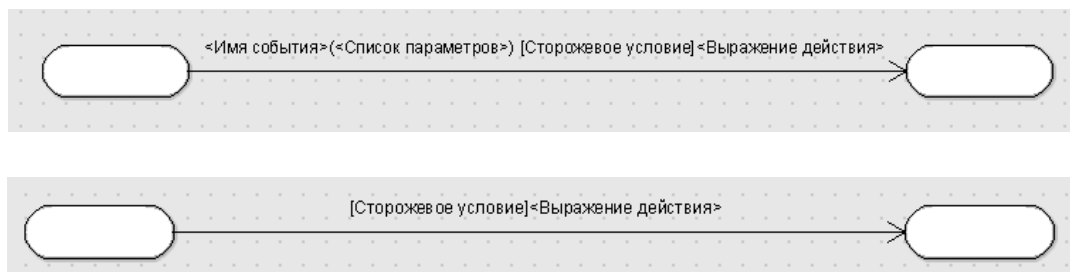


Рис. 3.7. Выражение действия

В общем случае выражение действия может содержать целый список отдельных действий, разделенных символом «;». Все действия списка должны различаться между собой и следовать в порядке записи. На синтаксис записи выражений действия не накладывается никаких ограничений. Основное условие, чтобы запись была понятна разработчикам модели и программистам. Как правило, выражение действия записывают на одном из языков программирования, который предполагается использовать для реализации модели.

3.9. Ветвление

Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста. В этот ромб может входить только одна стрелка от того состояния действия, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей (рис. 3.8).

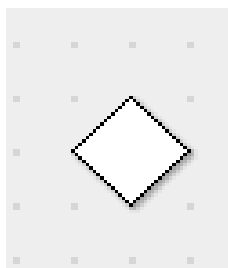


Рис. 3.8. Ветвление

Принято входящую стрелку присоединять к верхней или левой вершине символа ветвления. Выходящих переходов может быть два или более, но для каждого из них явно указывается соответствующее сторожевое условие в форме логического выражения.

3.10. Параллельные процессы

Один из недостатков обычных блок-схем алгоритмов связан с проблемой изображения параллельных ветвей отдельных вычислений. Поскольку распараллеливание вычислений существенно повышает общее быстродействие программных систем, необходимы графические примитивы для представления параллельных процессов. В языке UML для этой цели используется специальный символ разделения и слияния параллельных вычислений или потоков управления. Таким символом является прямая черточка (рис. 3.9).

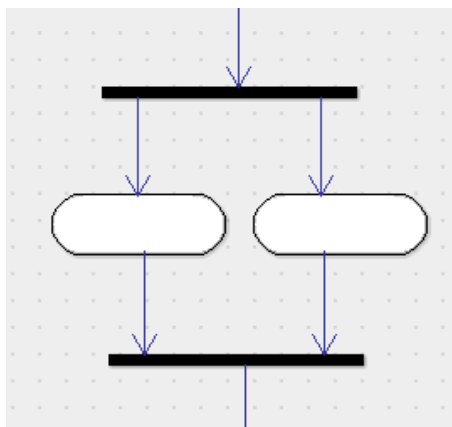


Рис. 3.9. Параллельные процессы

Как правило, такая черточка изображается отрезком горизонтальной линии, толщина которой несколько шире основных сплошных линий диаграммы деятельности. При этом разделение имеет один входящий переход и несколько выходящих, а слияние имеет несколько входящих переходов и один выходящий.

3.11. Дорожки

Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделированием бизнес-процессов. Действительно, деятельность любой организации также представляет собой совокупность

отдельных действий, направленных на достижение требуемого результата. Однако применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несет ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому.

Для моделирования этих особенностей в языке UML используется специальная конструкция, получившая название дорожки (рис. 3.10).

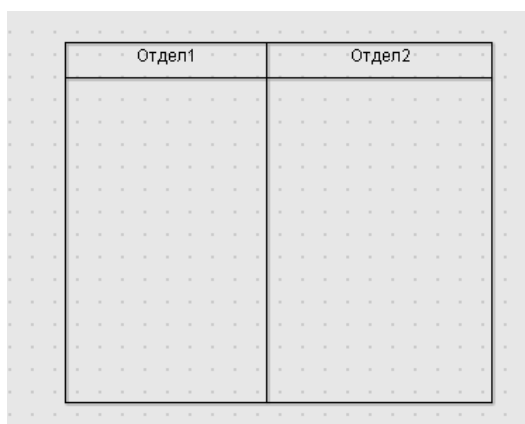


Рис. 3.10. Дорожки

Все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением (отделом, группой, отделением, филиалом) организации.

Названия подразделений явно указываются в верхней части дорожки. Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение. Порядок следования дорожек не несет какой-либо семантической информации и определяется соображениями удобства.

3.12. Объекты

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо иницируют выполнение действий, либо определяют некоторый их результат. Действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Поскольку в таком

ракурсе объекты играют определенную роль в понимании процесса деятельности, иногда возникает необходимость явно указать их на диаграмме.

Для графического представления объектов используется прямоугольник, внутри которого пишется имя соответствующего объекта, которое обязательно подчеркивается (рис. 3.11). Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой. Соответствующая зависимость определяет состояние конкретного объекта после выполнения предшествующего действия. Если стрелка перехода направлена от состояния действия к объекту, то это указывает на то, что объект порождается в результате выполнения действия. Если же стрелка перехода направлена от объекта в сторону состояния действия, то это указывает на тот факт, что данный объект будет являться инициализатором выражения действия в указанном состоянии.



Рис. 3.11. Объекты

На диаграмме деятельности с дорожками расположение объекта может иметь некоторый дополнительный смысл, а именно: если объект расположен на границе двух дорожек, то это может означать, что переход к следующему состоянию действия в соседней дорожке ассоциирован с готовностью некоторого документа (объект в некотором состоянии). Если же объект целиком расположен внутри дорожки, то и состояние этого объекта целиком определяется действиями данной дорожки.

Для синхронизации отдельных действий на диаграмме деятельности никаких дополнительных обозначений не используется, поскольку синхронизация параллельных процессов может быть реализована с помощью переходов «разделение-слияние».

3.13. Программное обеспечение для проектирования алгоритмов работы информационной системы

Microsoft Visio

Microsoft Visio – коммерческое программное обеспечение, которое предоставляет возможности для быстрого создания деловой графики различной степени сложности: схем бизнес-процесса, технических, инженерных рисунков, презентаций, разнообразных вариантов организационных, маркетинговых и технических диаграмм электрических и электронных схем, систем транспортных коммуникаций и т.д. (рис. 3.12).

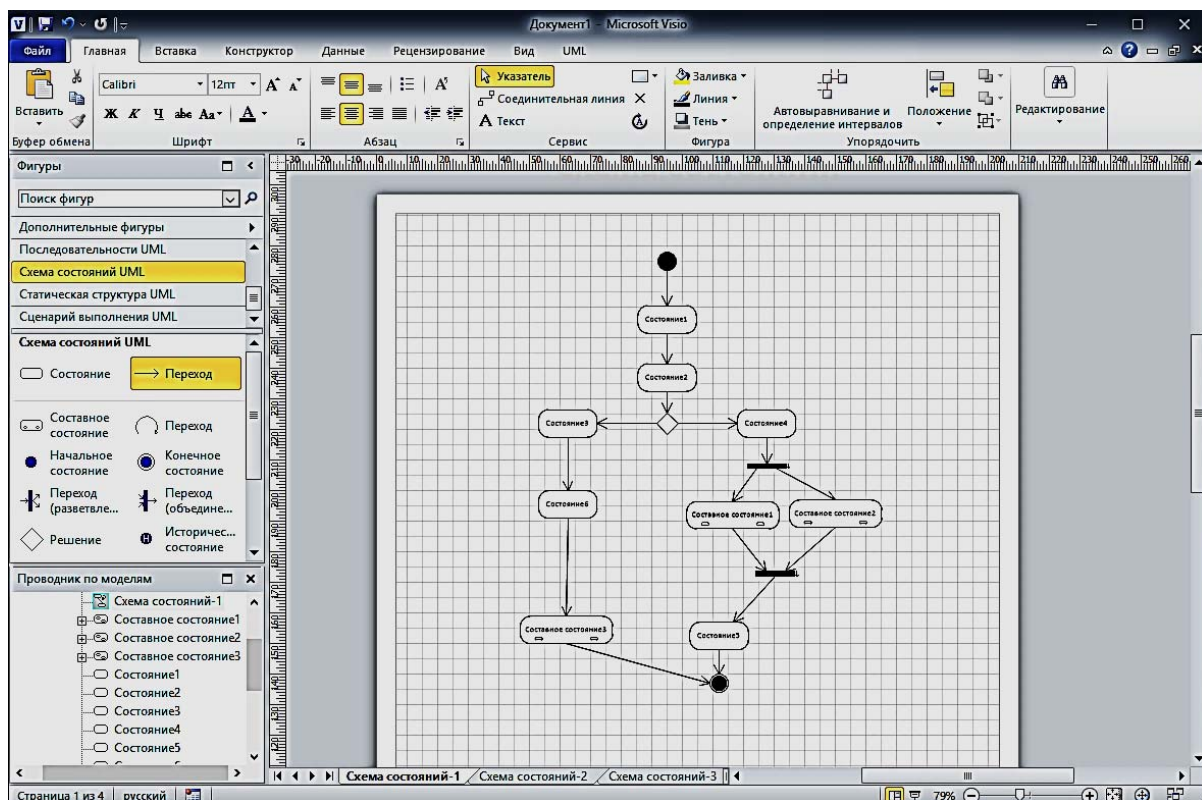


Рис. 3.12. Интерфейс Visio

Основная идея, заложенная в Microsoft Visio, – создать возможность эффективного использования в индивидуальных проектах готовых профессиональных наработок, представленных в виде богатой встроенной коллекции библиотек Visio, в которой весь арсенал элементов разбит по тематическим категориям и скомпонован в трафареты. Таким образом, задача создания необходимой графики сводится к выбору необходимого трафарета и перетаскивания нужной фигуры на страницу документа. На созданные объекты затем легко добавляются цветовые темы, фон и заголовки,

и получают профессионально оформленные документы. Особенно привлекательным Visio делает возможность разрабатывать свои собственные библиотеки (Stensils) с графическими фигурами (Master).

Основные преимущества Visio:

- создание профессиональных схем: с Visio легко приступить к работе благодаря коллекции готовых схем начального уровня, а также контекстным советам и рекомендациям. В распоряжении пользователя шаблоны и тысячи фигур. Можно использовать смарт-фигуры для повышения эффективности, а также темы и эффекты, чтобы быстрее придать схемам профессиональный вид;

- связывание схем с источниками данных: фигуры Visio можно связать с информацией из различных популярных источников данных, в том числе Microsoft Excel, служб Microsoft Excel, Active Directory, MS SQL Server, MS SQL Azure, а также из списков SharePoint и служб Business Connectivity;

- наглядное представление для всех. Созданную блок-схему, временную шкалу, схемы процессов, организационную структуру, ИТ-архитектуру или план этажа можно представить сотрудникам компании для просмотра или совместной работы с помощью служб Visio в Office 365 или SharePoint. По мере изменения данных схемы в браузере обновляются. Безопасность схем можно обеспечить с помощью управления правами на доступ к данным (IRM), защиты от потери данных (DLP) и многофакторной проверки подлинности;

- моделирование бизнес-процессов, соответствующее актуальным отраслевым стандартам. Фигуры соответствуют отраслевым стандартам, включая UML 2.4 (универсальный язык моделирования), BPMN 2.0 (нотация моделирования бизнес-процессов) и IEEE.

Microsoft Visual Studio

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств (рис. 3.13). Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

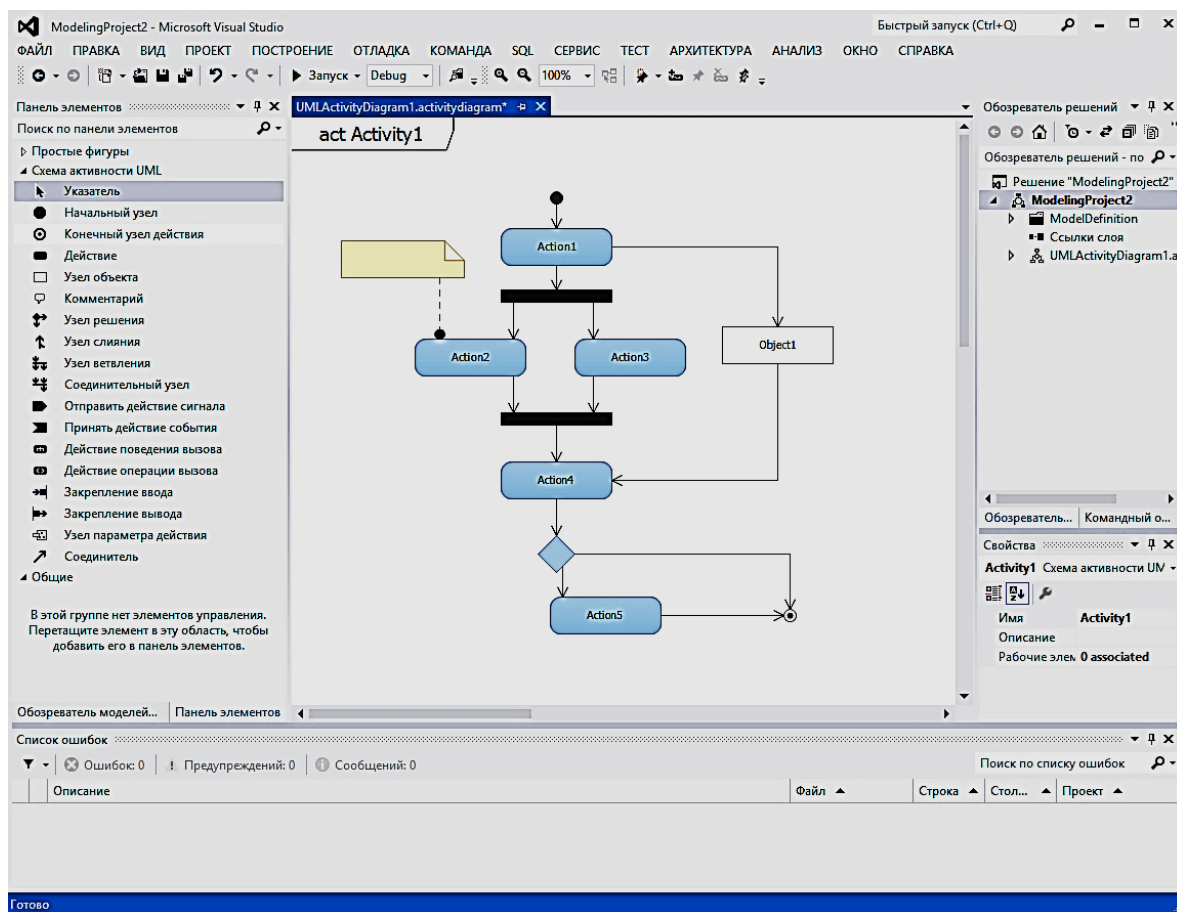


Рис. 3.13. Интерфейс Microsoft Visual Studio

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Контрольные вопросы

1. В чем состоит назначение диаграммы деятельности?
2. Чем диаграммы деятельности отличаются от блок-схем? Какие преимущества это обещает разработчикам?
3. Что представляют собой начальное и конечное состояния? Для чего они служат?
4. Как на диаграмме деятельности обозначается состояние действия?
5. Какие виды переходов можно реализовать на диаграмме деятельности?
6. Что представляет собой событие?
7. Какие требования предъявляются к сторожевым условиям?
8. Основное назначение перехода с указанием выражения действия.
9. Что такое дорожка?
10. В чем состоит смысл объектов?

Задания для самостоятельного выполнения

Требуется разработать диаграмму деятельности для некоторой предметной области. Требования к содержанию диаграммы:

- минимум, три дорожки;
- использование всех типов переходов;
- минимум, один параллельный процесс, реализуемый в рамках одной дорожки;
- минимум, три объекта (должны быть использованы как внутренние, так и внешние объекты);
- минимум, один локальный цикл, реализуемый в рамках одной дорожки;
- минимум, один глобальный цикл, реализуемый в рамках нескольких дорожек;
- диаграмма, обязательно должна отражать специфику предметной области.

Пример диаграммы деятельности приведен в прил. А. Примеры предметной области приведены в прил. Б.

Глава 4

СИСТЕМНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ИНФОРМАЦИОННЫХ СИСТЕМ

4.1. Концепция IDEF0

Методология IDEF0 основана на следующих концептуальных положениях.

1. *Создание функциональной модели системы.* IDEF0 используется для построения функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции. Данную модель разрабатывают для понимания, анализа и принятия решений о реконструкции (реинжиниринге), или замене существующей, либо проектировании новой системы. Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.

2. *Блочное моделирование и его графическое представление.* Основной концептуальный принцип методологии IDEF – представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе.

3. *Лаконичность и точность.* Документация, описывающая систему, должна быть точной и лаконичной. Графический язык позволяет лаконично, однозначно и точно показать все элементы (блоки) системы и все отношения и связи между ними, выявить ошибочные, лишние или дублирующие связи и т.д.

4. *Передача информации.* Средства IDEF0 облегчают передачу информации от одного участника разработки модели (отдельного разработчика или рабочей группы) к другому. К числу таких средств относятся:

- диаграммы, основанные на простой графике блоков и стрелок, легко читаемые и понимаемые;
- метки на естественном языке для описания блоков и стрелок, а также глоссарий и сопроводительный текст для уточнения смысла элементов диаграммы;
- последовательная декомпозиция диаграмм, строящаяся по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение;

– древовидные схемы иерархии диаграмм и блоков, обеспечивающие обозримость модели в целом и входящих в нее деталей.

5. Строгость и формализм. Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей. Все стадии и этапы разработки и корректировки модели должны строго, формально документироваться с тем, чтобы при ее эксплуатации не возникало вопросов, связанных с неполнотой или некорректностью документации.

6. Итеративное моделирование. Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется. Такая организация работы способствует оптимальному использованию знаний системного аналитика, владеющего методологией и техникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.

4.2. Принципы построения модели IDEF0

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. Для описания работы предприятия необходимо построить модель. Такая модель должна быть адекватна предметной области; следовательно, она должна содержать в себе знания всех участников бизнес-процессов организации.

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы. Моделируемая система рассматривается как произвольное подмножество окружающей среды. Произвольное потому, что, во-первых, разработчик сам умозрительно определяет, будет ли некий объект компонентом системы, или он будет его рассматривать как внешнее воздействие, и, во-вторых, оно зависит от точки зрения на систему. Система имеет границу, которая

отделяет ее от внешней среды. Взаимодействие системы с окружающим миром описывается как вход (нечто, что перерабатывается системой), выход (результат деятельности системы), управление (стратегии и процедуры, под управлением которых производится работа) и механизм (ресурсы, необходимые для проведения работы). Находясь под управлением, система преобразует входы в выходы, используя механизмы.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т.е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель. Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами; другими словами, разработчик должен определить, что будет в дальнейшем рассматриваться как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будут существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ; другими словами, первоначально необходимо определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента – широту и глубину. Широта подразумевает определение границ модели, глубина определяет, на каком уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени – трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции.

4.3. Синтаксис IDEF0

Набор структурных компонентов языка, их характеристики и правила, определяющие связи между компонентами, представляют собой синтаксис языка. Компоненты синтаксиса IDEF0 – функциональные блоки, интерфейсные дуги, диаграммы и правила. Функциональные блоки представляют функции, определяемые как деятельность, процесс, операция, действие или преобразование. Интерфейсные дуги представляют данные или материальные объекты, связанные с функциями. Правила определяют, как следует

применять компоненты; диаграммы обеспечивают формат графического и словесного описания моделей. Формат образует основу для управления конфигурацией модели.

4.4. Функциональный блок

Функциональный блок описывает деятельность. Внутри каждого блока помещается его имя и номер (рис. 4.1). Имя должно быть активным глаголом или глагольным оборотом, описывающим функцию. Номер блока размещается в правом нижнем углу. Номера блоков используются для их идентификации на диаграмме и в соответствующем тексте.

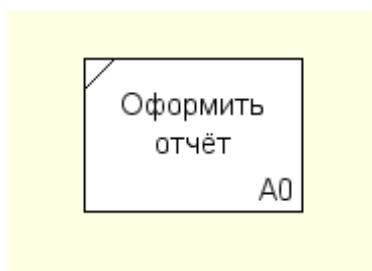


Рис. 4.1. Блок

Синтаксические правила использования блоков:

- размеры функциональных блоков должны быть достаточными для того, чтобы включить имя блока;
- функциональные блоки должны быть прямоугольными, с прямыми углами;
- функциональные блоки должны быть нарисованы сплошными линиями.

4.5. Интерфейсная дуга

Интерфейсная дуга формируется из одного или более отрезков прямых и наконечника на одном конце. Сегменты интерфейсных дуг могут быть прямыми или ломаными; в последнем случае горизонтальные и вертикальные отрезки сопрягаются дугами, имеющими угол 90° (рис. 4.2).

Интерфейсные дуги не представляют поток или последовательность событий, как в традиционных блок-схемах потоков или процессов. Они лишь показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться.

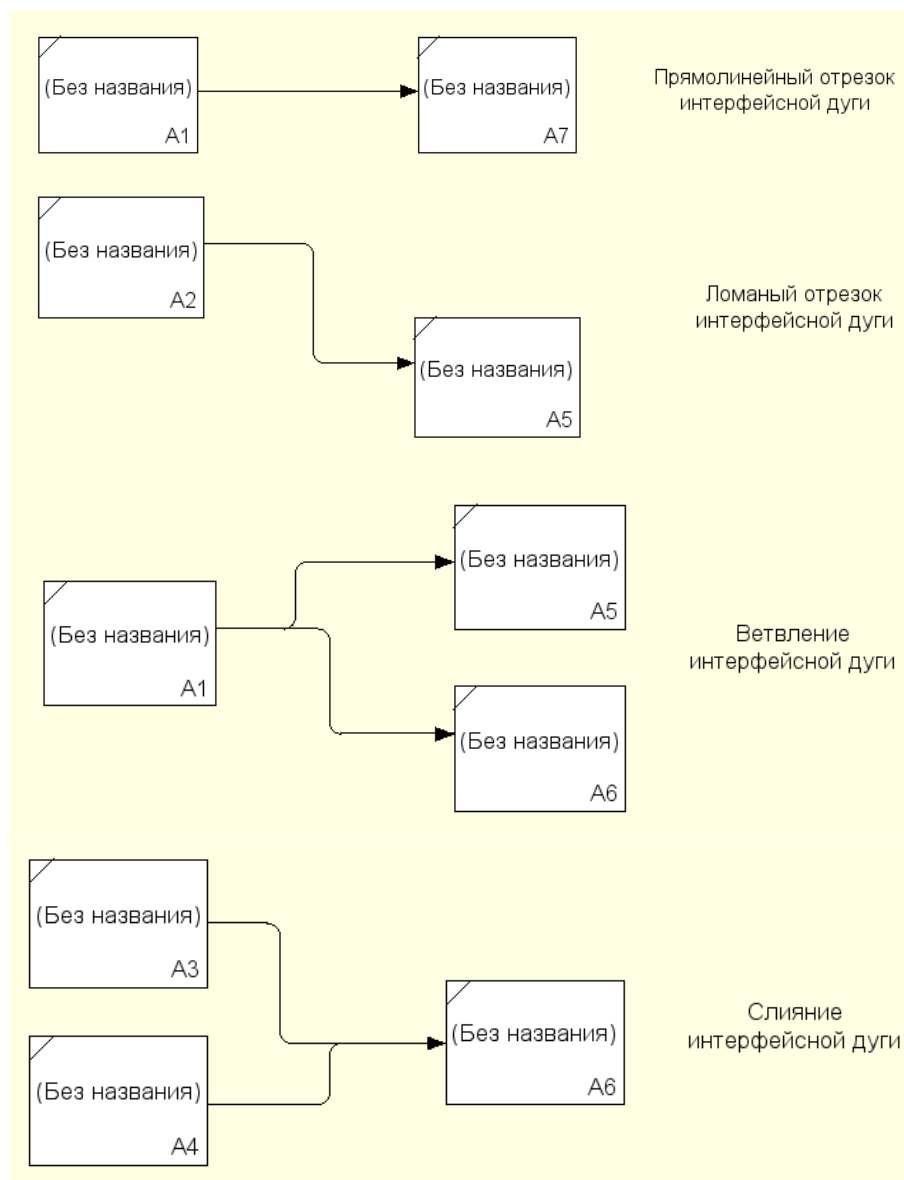


Рис. 4.2. Интерфейсная дуга

Синтаксические правила использования интерфейсных дуг:

- ломаные интерфейсные дуги изменяют направление только под углом 90° ;
- интерфейсные дуги должны быть нарисованы сплошными линиями различной толщины;
- интерфейсные дуги могут состоять только из вертикальных или горизонтальных отрезков; отрезки, направленные по диагонали, не допускаются;
- концы интерфейсных дуг должны касаться внешней границы функционального блока, но не должны пересекать ее;
- интерфейсные дуги должны присоединяться к функциональному блоку на его сторонах (присоединение в углах не допускается).

4.6. Семантика функциональных блоков и интерфейсных дуг

Семантика определяет содержание (значение) синтаксических компонентов языка и способствует правильности их интерпретации. Интерпретация устанавливает соответствие между блоками и интерфейсными дугами, с одной стороны, и функциями и их интерфейсами – с другой.

Поскольку IDEF0 есть методология функционального моделирования, имя блока, описывающее функцию, должно быть глаголом или глагольным оборотом; например, имя блока «Выполнить проверку» означает, что блок с таким именем превращает непроверенные детали в проверенные. После присваивания блоку имени к соответствующим его сторонам присоединяются входные, выходные и управляющие интерфейсные дуги, а также интерфейсные дуги механизма, что и определяет наглядность и выразительность изображения блока IDEF0.

Чтобы гарантировать точность модели, следует использовать стандартную терминологию. Функциональные блоки именуются глаголами или глагольными оборотами, и эти имена сохраняются при декомпозиции. Интерфейсные дуги и их сегменты, как отдельные, так и связанные в «пучок», помечаются существительными или оборотами существительного. Метки сегментов позволяют конкретизировать данные или материальные объекты, передаваемые этими сегментами, с соблюдением синтаксиса ветвлений и слияний.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи «функциональный блок – интерфейсные дуги». В свою очередь, сторона блока, к которой присоединена интерфейсная дуга, однозначно определяет ее роль (рис. 4.3). Интерфейсные дуги, входящие в левую сторону блока, – входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Интерфейсные дуги, входящие в блок сверху, – управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход. Интерфейсные дуги, покидающие блок справа, – выходы, т.е. данные или материальные объекты, произведенные функцией.

Интерфейсные дуги, подключенные к нижней стороне блока, представляют механизмы. Интерфейсные дуги, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Другие средства могут наследоваться из родительского блока. Интерфейсные дуги механизма, направленные вниз, являются интерфейсными дугами вызова.

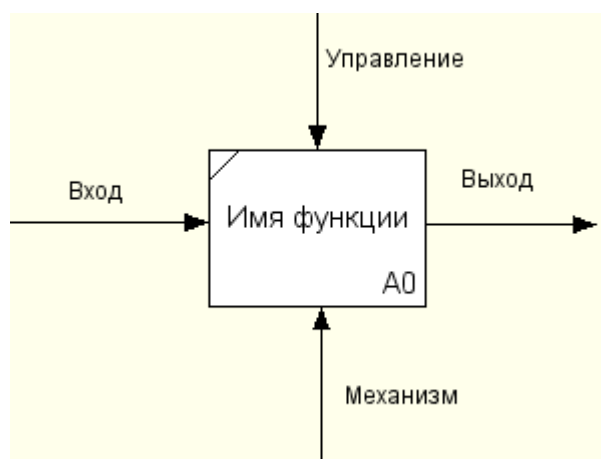


Рис. 4.3. Роли интерфейсных дуг

Интерфейсные дуги вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (функциональный блок).

4.7. Имена и метки

Как указывалось выше, имена функций представляют собой глаголы или глагольные обороты. Примеры таких имен приведены на рис. 4.4.

производить детали	планировать ресурсы	наблюдать
наблюдать за выполнением	проектировать систему	эксплуатировать
разработать детальные чертежи	изготовить компонент	проверять деталь

Рис. 4.4. Имена функций

Интерфейсные дуги идентифицируют данные или материальные объекты, необходимые для выполнения функции или производимые ею. Каждая интерфейсная дуга должна быть помечена существительным или оборотом существительного, как, например, на рис. 4.5.

Спецификации	отчет об испытаниях	бюджет
Конструкторские требования	конструкция детали	директива
Инженер-конструктор	плата в сборе	требования

Рис. 4.5. Метки интерфейсных дуг

Семантические правила блоков и интерфейсных дуг:

- имя блока должно быть активным глаголом или глагольным оборотом;
- обязательными интерфейсными дугами любого функционального блока являются: выход, механизм и управление;
- входные интерфейсные дуги должны связываться с левой стороной блока;
- управляющие интерфейсные дуги должны связываться с верхней стороной блока;
- выходные интерфейсные дуги могут быть связаны со входом, механизмом или управлением функционального блока;
- интерфейсные дуги механизма (кроме интерфейсных дуг вызова) должны указывать вверх и подключаться к нижней стороне блока;
- интерфейсные дуги вызова механизма должны указывать вниз, подключаться к нижней стороне блока и помечаться ссылкой на вызываемый блок;
- сегменты интерфейсных дуг, за исключением интерфейсных дуг вызова, должны помечаться существительным или оборотом существительного, если только единственная метка интерфейсной дуги не относится к ней в целом;
- чтобы связать интерфейсную дугу с меткой, следует использовать «тильду» ();
- в метках интерфейсных дуг не должны использоваться следующие термины: функция, вход, управление, выход, механизм, вызов.

Пример фрагмента модели IDEF0 представлен на рис. 4.6.

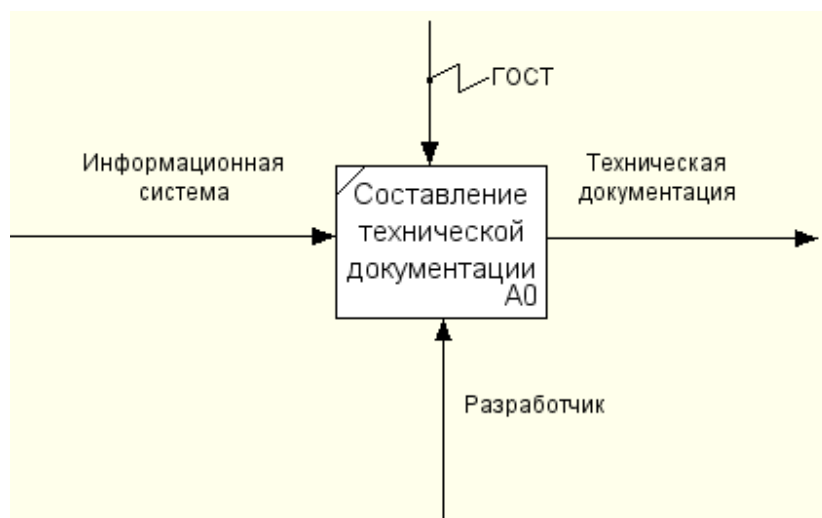


Рис. 4.6. Модель IDEF0

4.8. Диаграммы IDEF0

Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. IDEF0-модели состоят из трех типов документов: графических диаграмм, текста и глоссария. Эти документы имеют перекрестные ссылки друг на друга. Графическая диаграмма – главный компонент IDEF0-модели, содержащий функциональные блоки, интерфейсные дуги, соединения функциональных блоков и интерфейсных дуг и ассоциированные с ними отношения. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Блоки представляют основные функции моделируемого объекта. Эти функции могут быть разбиты (декомпозированы) на составные части и представлены в виде более подробных диаграмм; процесс декомпозиции продолжается до тех пор, пока объект не будет описан на уровне детализации, необходимом для достижения целей конкретного проекта. Диаграмма верхнего уровня обеспечивает наиболее общее или абстрактное описание объекта моделирования. За этой диаграммой следует серия дочерних диаграмм, дающих более детальное представление об объекте.

4.9. Контекстная диаграмма верхнего уровня

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным функциональным блоком с граничными интерфейсными дугами (рис. 4.7). Эта диаграмма называется А-0. Интерфейсные дуги на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя общее для всего проекта. Это же справедливо и для всех интерфейсных дуг диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта. Диаграмма А-0 устанавливает область моделирования и ее границу.

Контекстная диаграмма А-0 также должна содержать краткие утверждения, определяющие точку зрения должностного лица или подразделения, с позиций которого создается модель, и цель, для достижения которой ее разрабатывают. Эти утверждения помогают руководить разработкой модели и ввести этот процесс в определенные рамки. Точка зрения определяет, что и в каком разрезе можно увидеть в пределах контекста модели. Изменение точки зрения приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект.

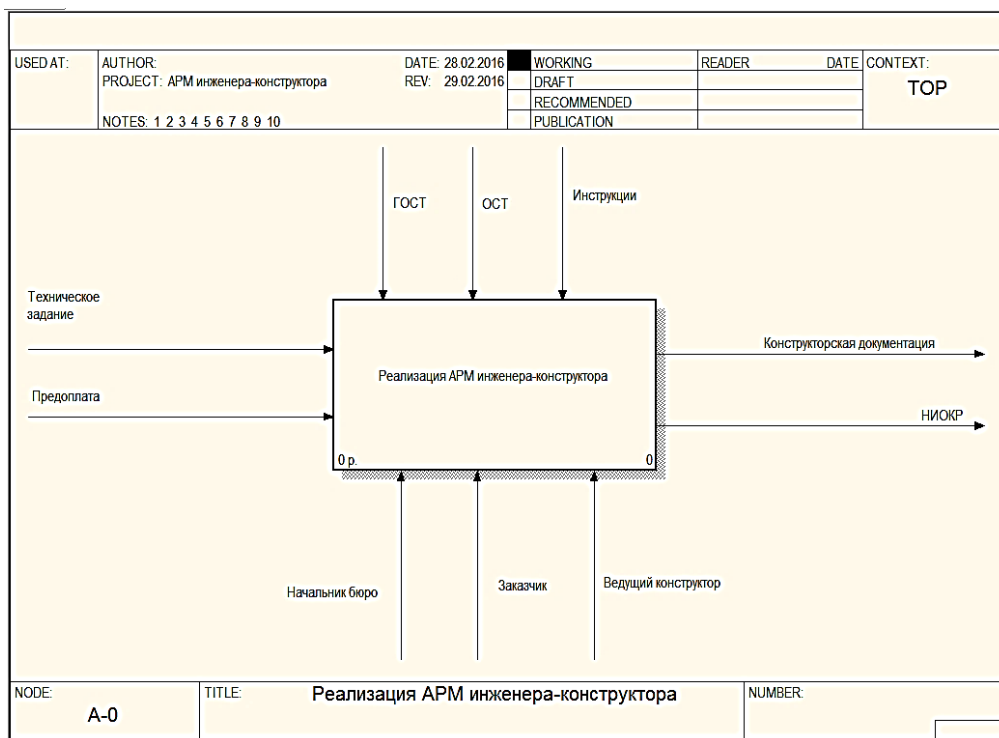


Рис. 4.7. Пример диаграммы А-0

Формулировка цели выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру. Наиболее важные свойства объекта обычно выявляются на верхних уровнях иерархии; по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция, в свою очередь, декомпозируется на элементы следующего уровня, и так происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Каждая подфункция моделируется отдельным функциональным блоком. Каждый родительский блок подробно описывается дочерней диаграммой на более низком уровне. Все дочерние диаграммы должны быть в пределах области контекстной диаграммы верхнего уровня.

4.10. Дочерняя диаграмма

Единственная функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции посредством создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части посредством создания дочерней диаграммы следующего,

более низкого уровня, на которой некоторые или все функции также могут быть разложены на составные части. Каждая дочерняя диаграмма содержит дочерние блоки и интерфейсные дуги, обеспечивающие дополнительную детализацию родительского блока.

Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Таким образом, дочерняя диаграмма как бы вложена в свой родительский блок.

4.11. Родительская диаграмма

Родительская диаграмма – та, которая содержит один или более родительских блоков. Каждая обычная (неконтекстная) диаграмма является также дочерней диаграммой, поскольку, по определению, она подробно описывает некоторый родительский блок. Таким образом, любая диаграмма может быть как родительской диаграммой (содержать родительские блоки), так и дочерней (подробно описывать собственный родительский блок). Аналогично функциональный блок может быть как родительским (подробно описываться дочерней диаграммой), так и дочерним (появляющимся на дочерней диаграмме). Основное иерархическое отношение существует между родительским блоком и дочерней диаграммой, которая его подробно описывает.

То, что блок является дочерним и раскрывает содержание родительского блока на диаграмме предшествующего уровня, указывается специальным ссылочным кодом, написанным ниже правого нижнего угла блока. Этот ссылочный код может формироваться несколькими способами, из которых самый простой заключается в том, что код, начинающийся с буквы А (по имени диаграммы А-0), содержит цифры, определяемые номерами родительских блоков. Например, показанные на рис. 4.8 коды означают, что диаграмма (рис. 4.9) является декомпозицией 1-го блока диаграммы, которая, в свою очередь, является декомпозицией 6-го блока диаграммы А0, а сами коды образуются присоединением номера блока.

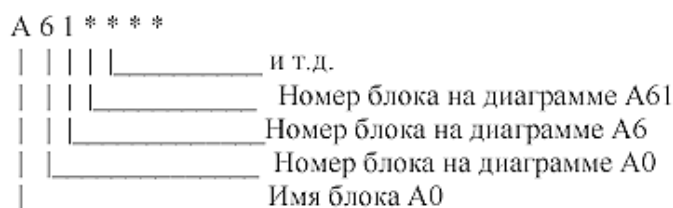


Рис. 4.8. Ссылочный код

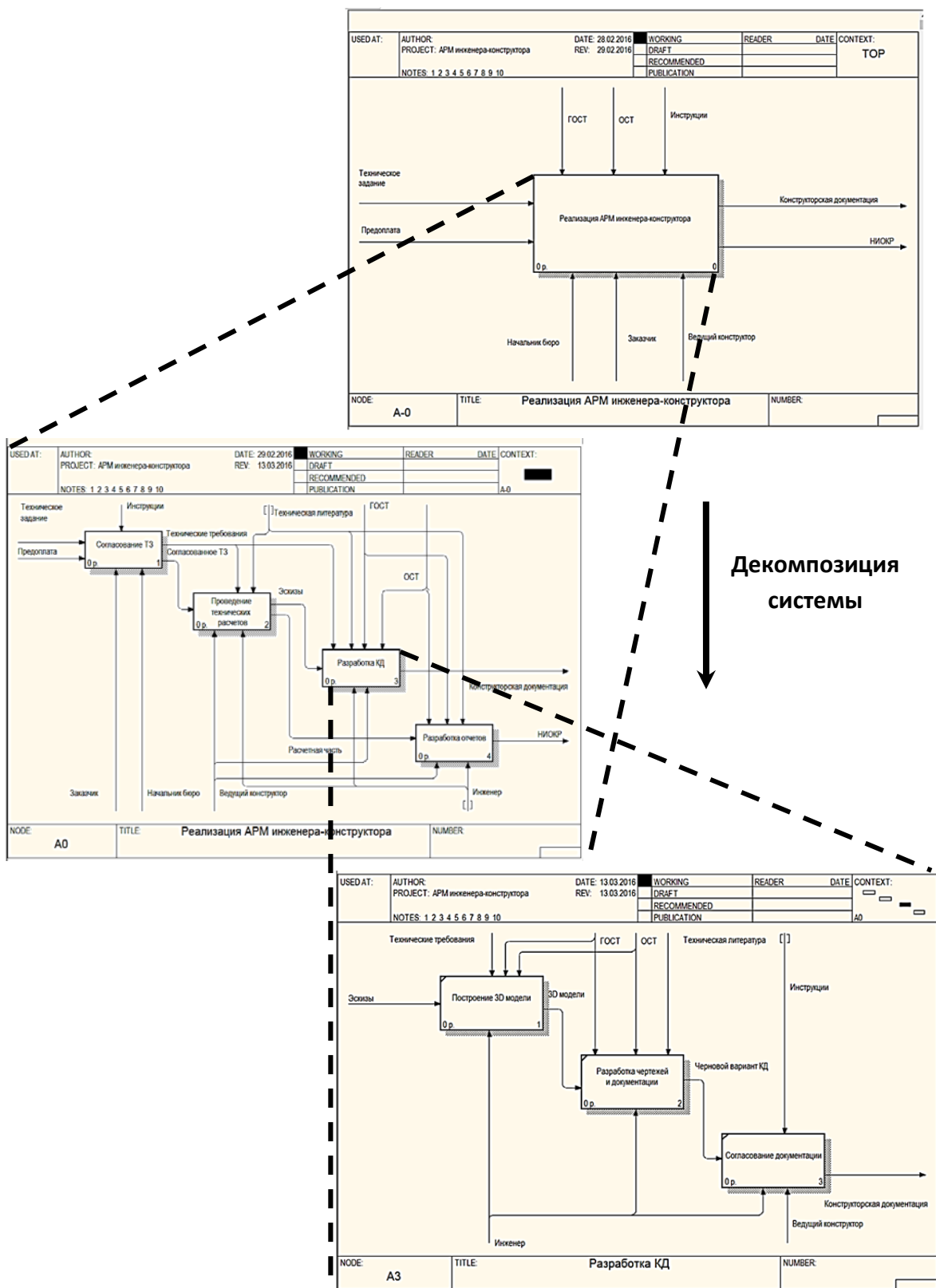


Рис. 4.9. Схема декомпозиции диаграммы IDEF0

4.12. Рекомендации по построению модели IDEF0

В IDEF0 существуют соглашения по построению диаграмм, которые призваны облегчить чтение и экспертизу модели. Рассмотрим общие правила построения диаграмм.

1. В составе модели должна присутствовать контекстная диаграмма А-0, которая содержит только один блок. Номер единственного блока на контекстной диаграмме А-0 должен быть 0.

2. Блоки на диаграмме должны располагаться по диагонали – от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров. Блоки на диаграмме, расположенные вверху слева, «доминируют» над блоками, расположенными внизу справа. «Доминирование» понимается как влияние, которое блок оказывает на другие блоки диаграммы. Расположение блоков на листе диаграммы отражает авторское понимание доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

3. Неконтекстные диаграммы должны содержать не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм на уровне, доступном для чтения, понимания и использования.

Диаграммы с количеством блоков менее трех вызывают серьезные сомнения в необходимости декомпозиции родительской функции. Диаграммы с количеством блоков более шести сложны для восприятия читателями и вызывают у автора трудности при внесении в нее всех необходимых графических объектов и меток.

4. Каждый блок неконтекстной диаграммы получает номер, помещаемый в правом нижнем углу; порядок нумерации – от верхнего левого к нижнему правому блоку (номера от 1 до 6).

5. Каждый блок, подвергнутый декомпозиции, должен иметь ссылку на дочернюю диаграмму; ссылка (например, узловой номер, С-номер или номер страницы) помещается под правым нижним углом блока.

6. Имена блоков (выполняемых функций) и метки интерфейсных дуг должны быть уникальными. Если метки интерфейсных дуг совпадают, это значит, что интерфейсные дуги отображают тождественные данные.

7. При наличии интерфейсных дуг со сложной топологией целесообразно повторить метку для удобства ее идентификации.

8. Следует обеспечить максимальное расстояние между блоками и поворотами интерфейсных дуг, а также между блоками и пересечениями интерфейсных дуг для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные интерфейсные дуги.

9. Блоки всегда должны иметь хотя бы одну управляющую, одну выходную интерфейсную дугу и одну интерфейсную дугу механизма, но могут не иметь входных интерфейсных дуг.

10. Если одни и те же данные служат и для управления, и для входа, вычерчивается только интерфейсная дуга управления. Этим подчеркивается управляющий характер данных и уменьшается сложность диаграммы.

11. Максимально увеличенное расстояние между параллельными интерфейсными дугами облегчает размещения меток, их чтение и позволяет проследить пути интерфейсных дуг (рис. 4.10).

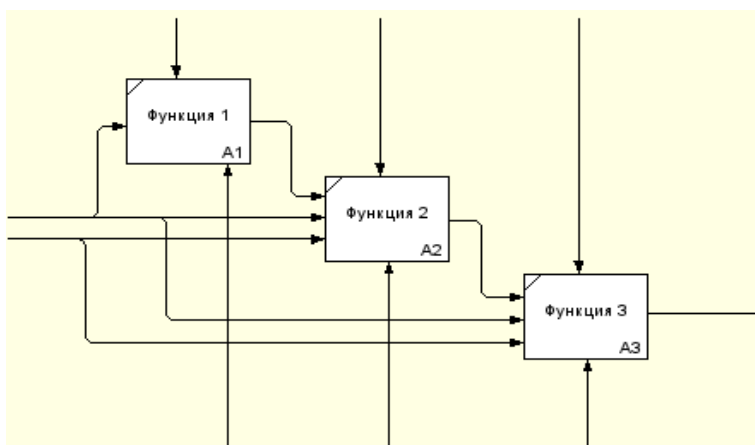


Рис. 4.10. Расположение интерфейсных дуг

12. Интерфейсные дуги связываются (сливаются), если они представляют сходные данные и их источник указан на диаграмме (рис. 4.11).

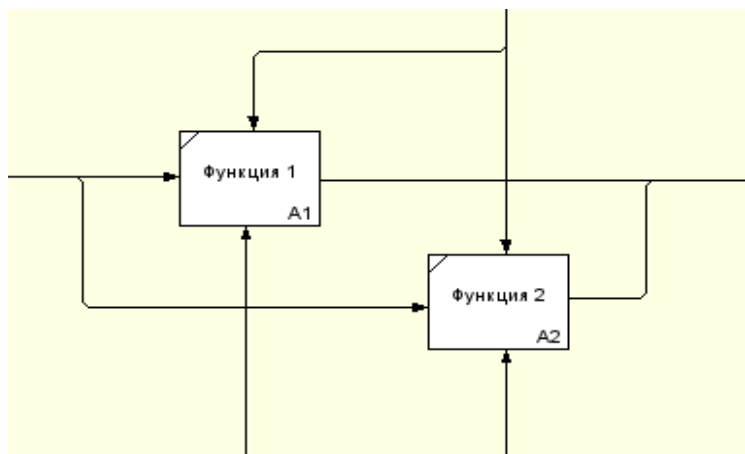


Рис. 4.11. Изображение интерфейсных дуг

13. Обратные связи по управлению должны быть показаны как «вверх и над» (рис. 4.12,а).

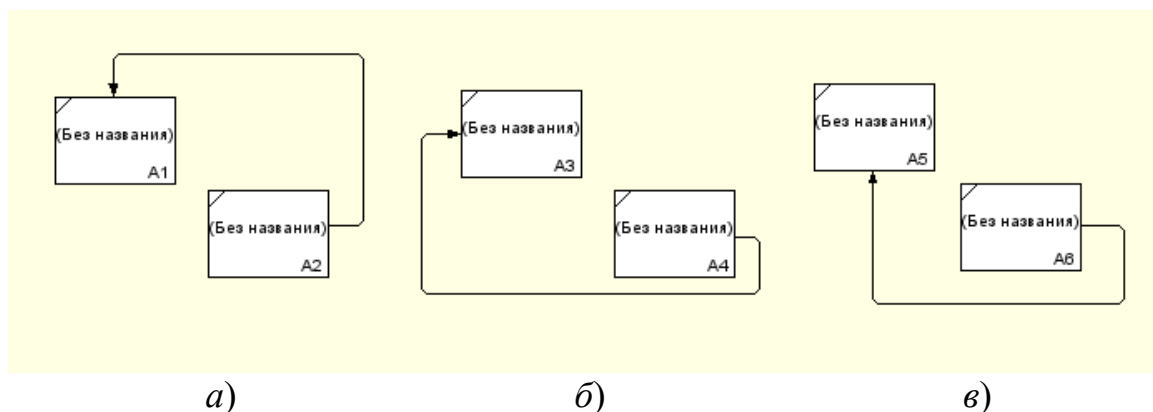


Рис. 4.12. Изображение обратных связей

Обратные связи по входу должны быть показаны как «вниз и под» (рис. 4.12,б). Так же показываются обратные связи посредством механизма (рис. 4.12,в). Таким образом, обеспечивается показ обратной связи при минимальном числе линий и пересечений.

14. Циклические обратные связи для одного и того же блока изображаются только для того, чтобы их выделить (рис. 4.13). Обычно обратную связь изображают на диаграмме, декомпозирующей блок. Однако иногда требуется выделить повторно используемые объекты.

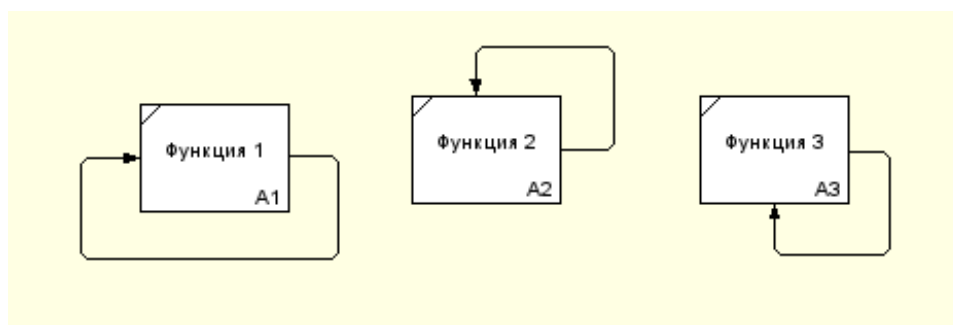


Рис. 4.13. Изображение циклических обратных связей

15. Одни и те же данные или объекты, порожденные одним блоком, могут использоваться сразу в нескольких других блоках. С другой стороны, интерфейсные дуги, порожденные в разных блоках, могут представлять собой одинаковые или однородные данные или объекты, которые в дальнейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций в IDEF0 используются разветвляющиеся и сливающиеся интерфейсные дуги (рис. 4.14).

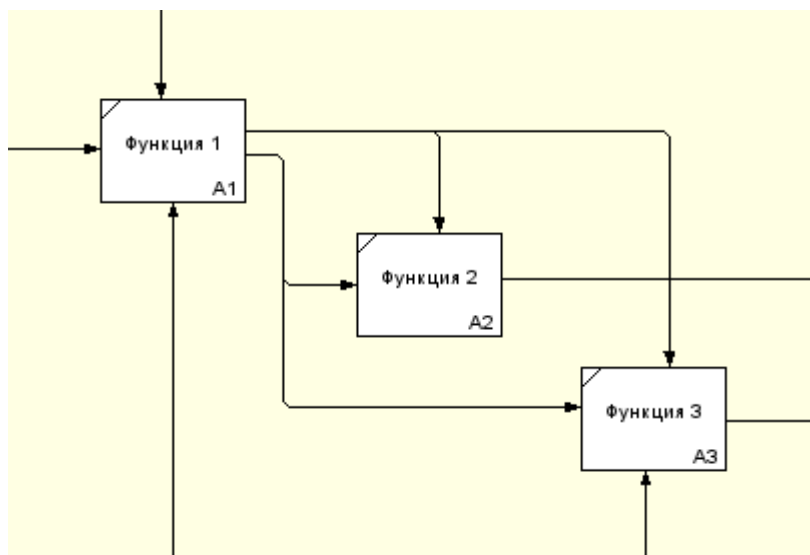


Рис. 4.14. Разветвление интерфейсной дуги

Интерфейсные дуги объединяются, если они имеют общий источник или приемник или они представляют связанные данные. Общее название лучше описывает суть данных. Следует минимизировать число интерфейсных дуг, касающихся каждой стороны блока, если, конечно, природа данных не слишком разнородна.

16. Если возможно, интерфейсные дуги присоединяются к блокам в одной и той же позиции (рис. 4.15). Тогда соединение интерфейсных дуг конкретного типа с блоками будет согласованным и чтение диаграммы упростится.

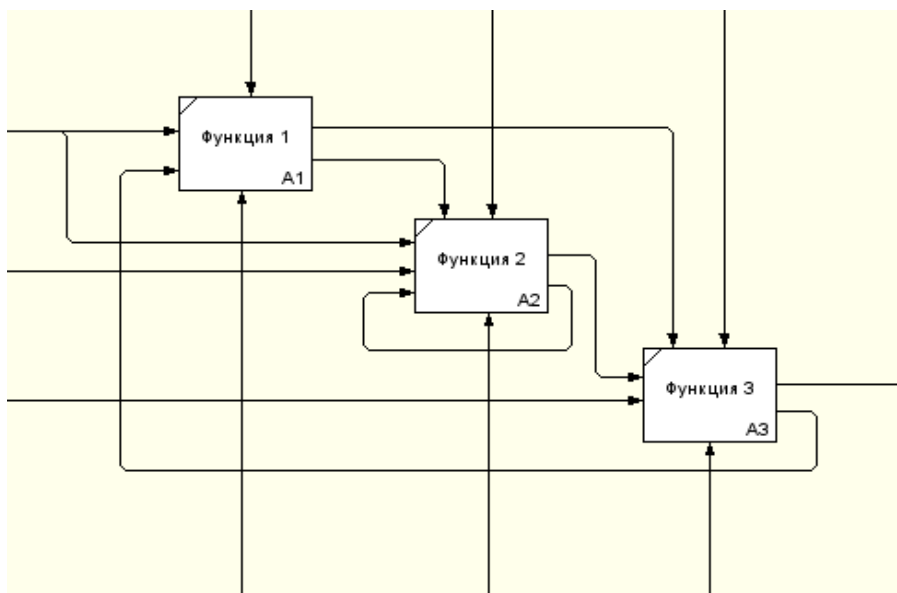


Рис. 4.15. Присоединение интерфейсных дуг к блокам

17. При соединении большого числа блоков необходимо избегать необязательных пересечений интерфейсных дуг. Следует минимизировать число петель и поворотов каждой интерфейсной дуги.

4.13. Программное обеспечение для построения диаграммы IDEF0

AllFusion Process Modeler

AllFusion Process Modeler (ранее BPwin) – коммерческий инструмент для моделирования, анализа, документирования и оптимизации процессов. AllFusion Process Modeler можно использовать для графического представления ИТ-процессов. Графически представленная схема выполнения работ, обмена информацией, документооборота визуализирует модель ИТ-процесса. Графическое изложение этой информации позволяет перевести задачи управления организацией из области сложного ремесла в сферу инженерных технологий.

AllFusion Process Modeler помогает четко документировать важные аспекты любых ИТ-процессов: действия, которые необходимо предпринять, способы их осуществления и контроля, требующиеся для этого ресурсы, а также визуализировать получаемые от этих действий результаты. AllFusion Process Modeler повышает бизнес-эффективность ИТ-решений, позволяя аналитикам и проектировщикам моделей соотносить корпоративные инициативы и задачи с бизнес-требованиями и процессами информационной архитектуры и проектирования приложений. Таким образом, формируется целостная картина деятельности предприятия: от потоков работ в небольших подразделениях до сложных организационных функций (рис. 4.16).

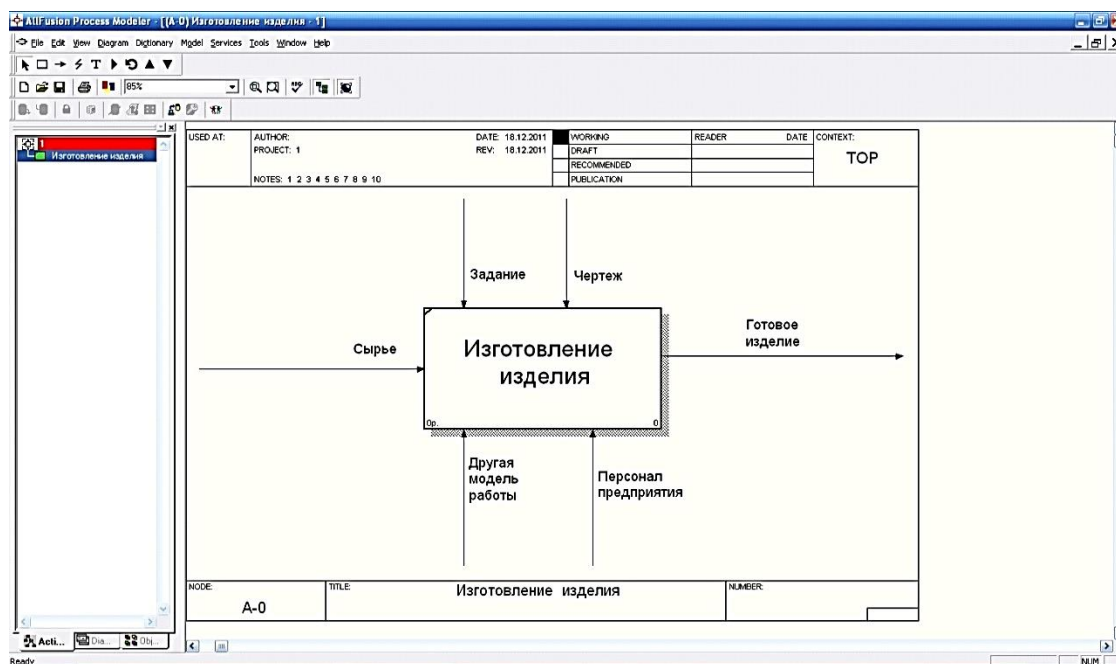


Рис. 4.16. Интерфейс AllFusion Process Modeler

AllFusion Process Modeler эффективен в проектах, связанных с описанием действующих баз предприятий, реорганизацией бизнес-процессов, внедрением корпоративной информационной системы. Продукт позволяет оптимизировать деятельность предприятия и проверить ее на соответствие стандартам ISO 9000, спроектировать оргструктуру, снизить издержки, исключить ненужные операции и повысить эффективность. В основу продукта заложены общепризнанные методологии моделирования, например, методология IDEF0 рекомендована к использованию Госстандартом РФ. Простота и наглядность моделей Process Modeler упрощает взаимопонимание между всеми участниками процессов. Распространенность самого AllFusion Process Modeler позволяет вести согласование функциональных моделей с партнерами в электронном виде. Продукт AllFusion Process Modeler (BPwin) создан компанией Computer Associates. AllFusion Process Modeler наряду с ERwin Data Modeler (ранее: ERwin), Data Model Validator (ранее: ERwin Examiner), Model Manager (ранее: ModelMart) входит в состав пакета программных средств AllFusion Modeling Suite, комплексное использование которого обеспечивает все аспекты моделирования информационных систем.

Ramus Educational

Ramus – это бесплатная программа, при помощи которой можно создавать визуальные диаграммы, используемые для наглядного отображения различных бизнес-процессов. Данное решение будет крайне полезно на «мозговых штурмах» и собраниях сотрудников предприятия. Помимо визуализации разных процессов и задач, создаваемые программой диаграммы также подходят для классификации и систематизации различных данных. Главное преимущество Ramus Educational заключается в том, что она поддерживает сразу две популярные методологии: DFD и IDEF0.

Программа предлагает пользователю удобный встроенный графический редактор для работы над диаграммами (рис. 4.17). В нем имеется библиотека уже готовых элементов: блоков, связей и даже целых структур. К сожалению, шаблоны с самими диаграммами в Ramus не предусмотрены, что является чуть ли не единственным ее недостатком. Из положительных моментов стоит отметить возможность создания собственных кодировок для элементов, которые помогают быстрее получить к ним доступ и упростить поиск нужного блока в готовом проекте. Кроме того, помимо самой диаграммы, Ramus также позволяет создать сопроводительную

документацию к ней. Функции для работы с отчетностью даже вынесены в отдельный модуль.

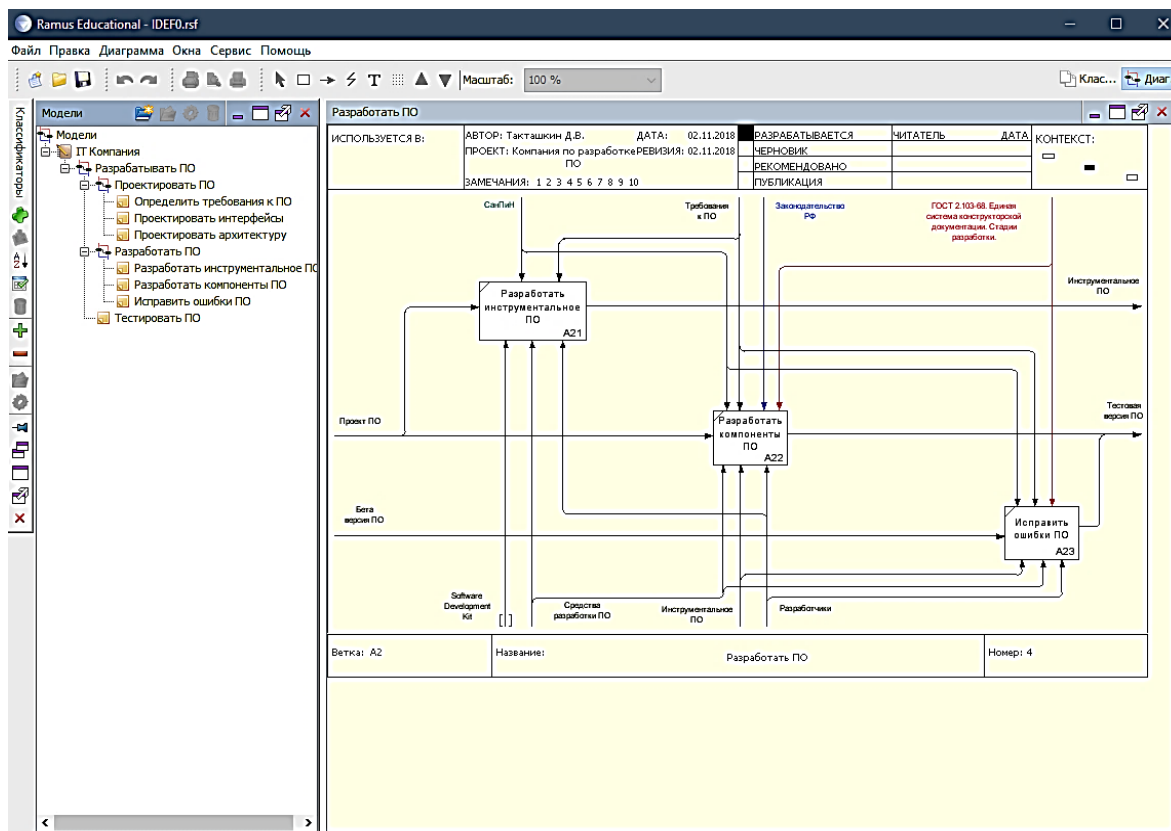


Рис. 4.17. Интерфейс Ramus Educational

Среди функций Ramus Educational стоит отметить возможность работы с формулами, управление логическими операциями и набор инструментов для улучшения внешнего вида диаграмм. Например, программа позволяет добавлять в графические проекты сглаживание и автоматически «выравнивать» элементы по горизонтали или вертикали. Разобраться в использовании всех этих функций несложно, потому как Ramus имеет довольно интуитивный интерфейс, который полностью переведен на русский язык.

Ключевые особенности:

- создание диаграмм по методологиям DFD и IDEF0;
- удобный встроенный графический редактор с большой базой готовых элементов;
- автоматическое составление отчетов и сопроводительной документации;
- масса инструментов для улучшения внешнего вида проектов;
- простой и понятный интерфейс.

Контрольные вопросы

1. Что такое функциональная модель?
2. В чем суть методологии IDEF0?
3. Перечислите основные понятия, лежащие в основе IDEF0 методологии.
4. Что представляет собой функциональный блок?
5. Каковы типы интерфейсных дуг в IDEF0 модели?
6. Приведите правило формулировки названия функционального блока в IDEF0 модели.
7. С чего начинается построение IDEF0 модели?
8. Что такое контекстная диаграмма?
9. Что такое родительский блок?
10. Для чего в IDEF0 методологии предусмотрено туннелирование?

Задания для самостоятельного выполнения

Требуется разработать диаграмму IDEF0 для некоторой предметной области. Требования к содержанию диаграммы:

- минимум, три уровня декомпозиции;
- минимум, по три механизма и управления к каждому функциональному блоку;
- на каждом уровне декомпозиции, минимум, по две туннелированных интерфейсных дуги;
- минимум, два цикла, размещенных на разных уровнях декомпозиции;
- диаграмма обязательно должна отражать специфику предметной области.

Пример диаграмм IDEF0 приведен в прил. А. Примеры предметной области приведены в прил. Б.

Глава 5

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ МЕТОДОЛОГИИ IDEF1X

5.1. Создание модели данных

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный для всех ее будущих пользователей способ организации данных и инструментальные средства управления данными.

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить следующие этапы проектирования.

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Концептуальное (инфологическое) проектирование – построение семантической модели предметной области, т.е. информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную систему управления базой данных (СУБД) и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, «концептуальная модель базы данных» и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности. Конкретный вид и содержание концептуальной модели базы данных определяются выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

- описание информационных объектов, или понятий предметной области и связей между ними;
- описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

3. Даталогическое (логическое) проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

4. Физическое проектирование – создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т.п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т.д.

5.2. Системный анализ предметной области

Первый этап проектирования БД – это системный анализ, т.е. подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами.

1. Функциональный подход к проектированию БД.

Этот метод является наиболее распространённым. Он реализует принцип «от задач» и применяется в том случае, когда известны функции некоторой группы лиц и комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

2. Предметный подход к проектированию БД.

Предметный подход применяется в тех случаях, когда у разработчиков есть чёткое представление о самой предметной области и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию предметной области и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД.

5.3. Инфологическое проектирование

Инфологическая модель (информационно-логическая модель) применяется на втором этапе проектирования БД, т.е. после словесного описания предметной области и постановки задачи. Инфологическая модель должна включать такое формализованное описание предметной области, которое будет «читабельно» не только для специалистов по базам данных, но и сторонних людей. Описание должно быть настолько ёмким, чтобы можно было оценить глубину и корректность проработки проекта БД, и не должно быть привязано к конкретной СУБД.

5.4. Даталогическое проектирование

Следующим шагом является выбор конкретной СУБД и отображение в ее среду спецификаций инфологической модели предметной области. Эту стадию называют даталогическим (логическим) проектированием БД.

Цель создания модели данных состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть интегрированы в любую базу данных.

При создании моделей данных используется метод семантического моделирования. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты диаграмм «сущность–связь» (entity-relationship diagram, ER-диаграмм, ERD).

ER-диаграмма – это графическое представление инфологической модели. Основные элементы ER-моделей:

- сущности (объекты);
- атрибуты сущностей;

- ключ сущности;
- связи между сущностями.

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от остальных экземпляров. Атрибут выражает определенное свойство объекта. На физическом уровне сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы. Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности и в конкретном атрибуте.

Модель «сущность–связь» относится к концептуальным моделям. В современном своем варианте она также является и логической моделью данных. Модель «сущность–связь» чаще всего применяется для проектирования структур баз данных. Важной особенностью модели «сущность–связь» является то, что по ней однозначно может быть построена физическая модель базы данных или ее схема.

Модель «сущность–связь» не является строго формальной. Первый вариант модели «сущность–связь» был предложен в 1976 г. Питером Ченом. В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). Кроме того, различные CASE-системы, реализующие одну и ту же нотацию, могут отличаться своими возможностями. По сути, все варианты диаграмм «сущность – связь» исходят из одной идеи – использовать графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями.

5.5. Сущность

Сущность – это класс однотипных объектов, информация о которых должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примерами сущностей могут быть такие классы объектов, как «Студент», «Адрес», «Человек». Каждая сущность в модели изображается в виде прямоугольника с наименованием (рис. 5.1).

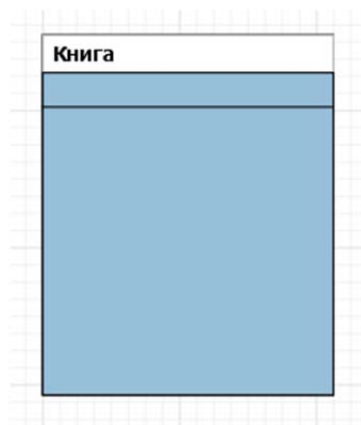


Рис. 5.1. Сущность

Экземпляр сущности – это конкретный представитель данной сущности. Например, представителем сущности «Человек» может быть «Иванов». Экземпляры сущностей должны быть различимы, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

5.6. Атрибут сущности

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством (параметром) сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными). Атрибуты изображаются в пределах прямоугольника, определяющего сущность (рис. 5.2).



Рис. 5.2. Атрибуты

Примерами атрибутов сущности «Человек» могут быть такие атрибуты, как «Фамилия», «Имя», «Отчество», «Номер паспорта» и т.п.

5.7. Ключ сущности

Ключ сущности – это избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Избыточность заключается в том, что удаление любого атрибута из ключа нарушает его уникальность. Ключевые атрибуты изображаются на диаграмме в разделе под названием таблицы, как атрибут «idКнига» (рис. 5.3).

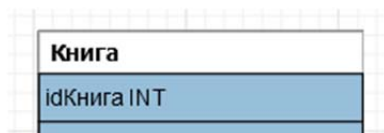


Рис. 5.3. Ключ

Сущность может иметь несколько различных ключей.

5.8. Связь

Связь – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Графически связь изображается линией, соединяющей две сущности.

Каждая связь имеет собственное имя, два конца и два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать» и т.п. Каждое из наименований относится к своему концу связи. Наименования можно не указывать. Будучи объявленными, имя связи и наименования концов обрабатываются CASE-системой и применяются в алгоритмах обработки диаграмм «сущность – связь» для построения таблиц базы данных.

Каждая связь имеет два параметра: тип и модальность.

Связь типа один-к-одному означает, что один экземпляр первой сущности связан только с одним экземпляром второй сущности. Связь один-к-одному может соответствовать ситуации, когда две сущности отражают разные характеристики одного и того же объекта (рис. 5.4).

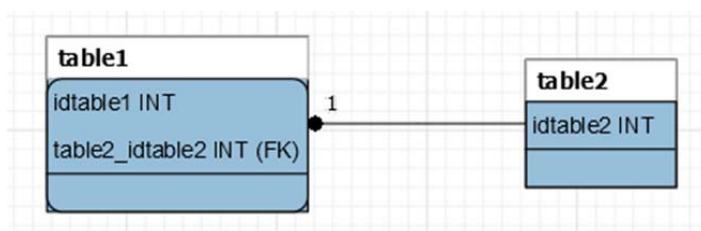


Рис. 5.4. Связь типа один-к-одному

Связь типа один-ко-многим означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется родительской, правая (со стороны «много») – дочерней (рис. 5.5).

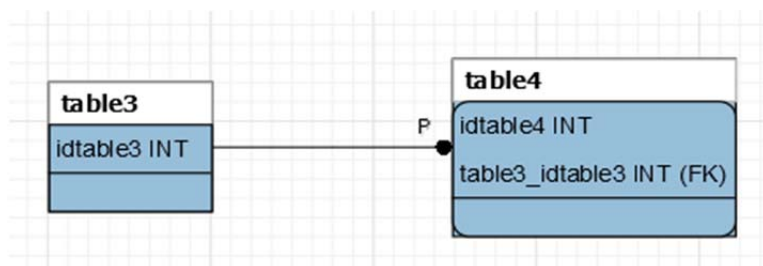


Рис. 5.5. Связь типа один-ко-многим

Связь типа много-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности (рис. 5.6).

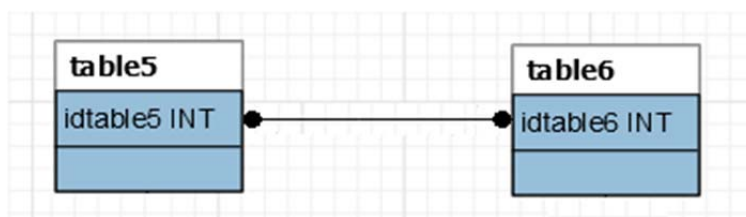


Рис. 5.6. Связь типа много-ко-многим

Связь типа много-ко-многим может быть создана только на уровне логической модели. Нотация требует, чтобы на физическом уровне связь много-ко-многим была преобразована в две связи типа один-ко-многим через новую вспомогательную сущность (рис. 5.7).

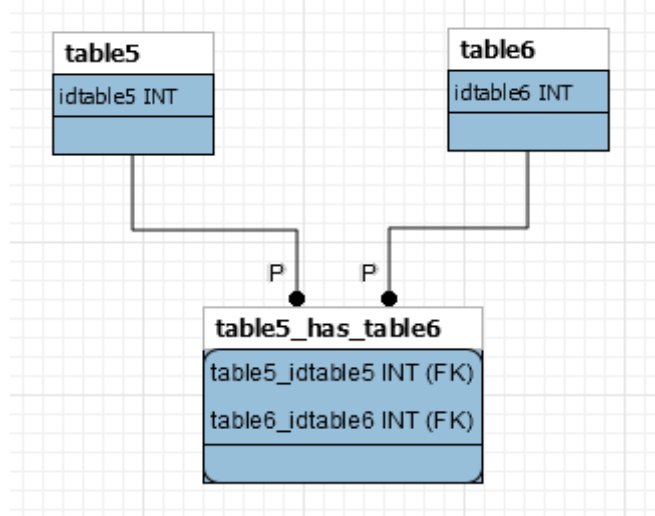


Рис. 5.7. Пример преобразования связи типа много-ко-многим

Каждая связь может быть реализована в одном из двух видов: идентифицируемая связь (модальность «должен») и неидентифицируемая связь (модальность «может») (рис. 5.8).

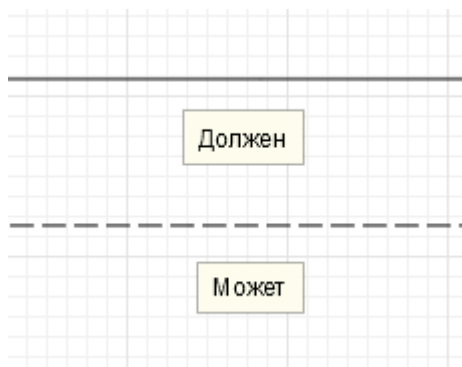


Рис. 5.8. Идентифицируемая и неидентифицируемая связи

Модальность «может» означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может и не быть связан ни с одним экземпляром. Модальность «должен» означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности.

5.9. Физическое проектирование

Физическое проектирование является последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например, реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации

логического проекта базы данных. Физическое проектирование заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных.

Проектирование базы данных – это итерационный процесс, который имеет свое начало, но не имеет конца и состоит из бесконечного ряда уточнений. Его следует рассматривать, прежде всего, как процесс познания. Как только проектировщик приходит к пониманию работы предприятия и смысла обрабатываемых данных, а также выражает это понимание средствами выбранной модели данных, приобретенные знания могут показать, что требуется уточнение и в других частях проекта. Особо важную роль в общем процессе успешного создания системы играет концептуальное и логическое проектирование базы данных. Если на этих этапах не удастся получить полное представление о деятельности предприятия, то задача определения всех необходимых пользовательских представлений или обеспечения защиты базы данных становится чрезмерно сложной или даже неосуществимой. К тому же, может оказаться затруднительным определение способов физической реализации или достижения приемлемой производительности системы. С другой стороны, способность адаптироваться к изменениям является одним из признаков удачного проекта базы данных.

5.10. Программное обеспечение для проектирования базы данных по методологии IDEF1X

AllFusion ERwin Data Modeler

AllFusion ERwin Data Modeler (ранее: ERwin) – CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

AllFusion ERwin Data Modeler (ERwin) предназначен для всех компаний, разрабатывающих и использующих базы данных, для администраторов баз данных, системных аналитиков, проектировщиков баз данных, разработчиков, руководителей проектов. AllFusion ERwin Data Modeler позволяет управлять данными в процессе корпоративных изменений, а также в условиях стремительно изменяющихся технологий.

AllFusion ERwin Data Modeler (ERwin) позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда AllFusion ERwin Data Modeler упрощает разработку базы данных и автоматизирует множество трудоемких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных (рис. 5.9).

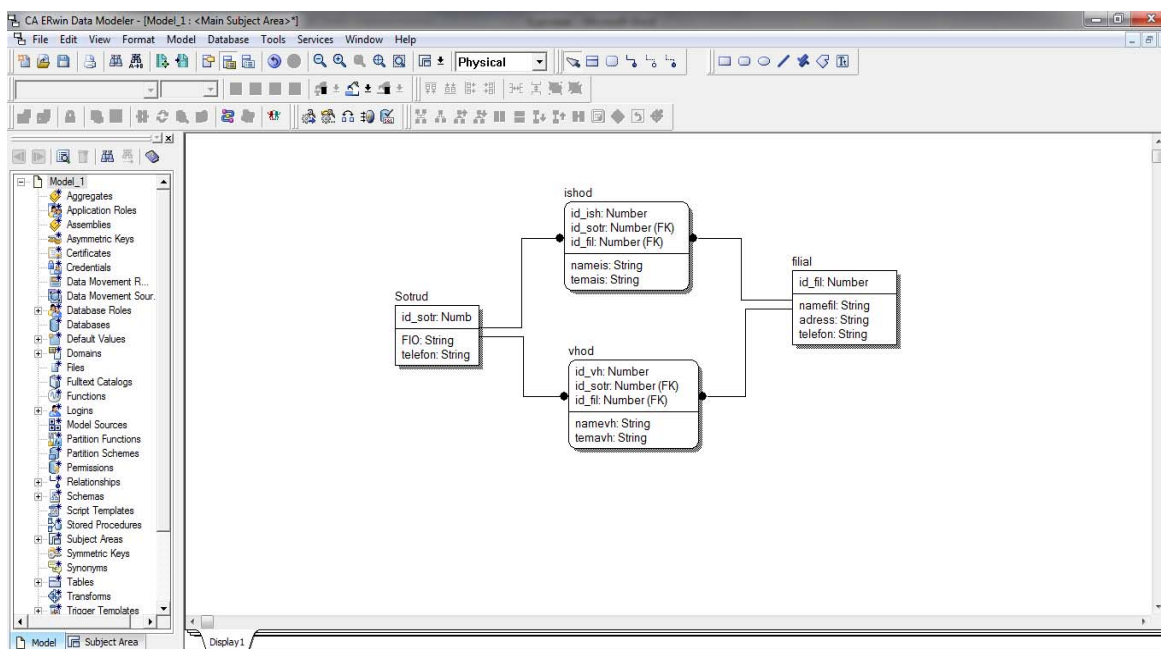


Рис. 5.9. Интерфейс AllFusion ERwin Data Modeler

Данное программное решение улучшает коммуникацию в организации, обеспечивая совместную работу администраторов и разработчиков баз данных, многократное использование модели, а также наглядное представление комплексных активов данных в удобном для понимания и обслуживания формате.

MySQL Workbench

MySQL Workbench – бесплатно распространяемый инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое окружение для системы баз данных MySQL. MySQL Workbench предоставляет комплекс инструментов для настройки сервера, администрирования пользователей и многое другое. Программа доступна для работы в операционных системах: Windows, Linux и Mac OS.

MySQL Workbench позволяет администратору или проектировщику баз данных визуально моделировать, создавать и управлять базами данных (рис. 5.10). Программное обеспечение обладает

всем необходимым для создания комплексной ER-моделей, прямой и обратной разработки, а также позволяет легко произвести сложные изменения в базе данных или исправления в документации, которые обычно требуют много времени и усилий.

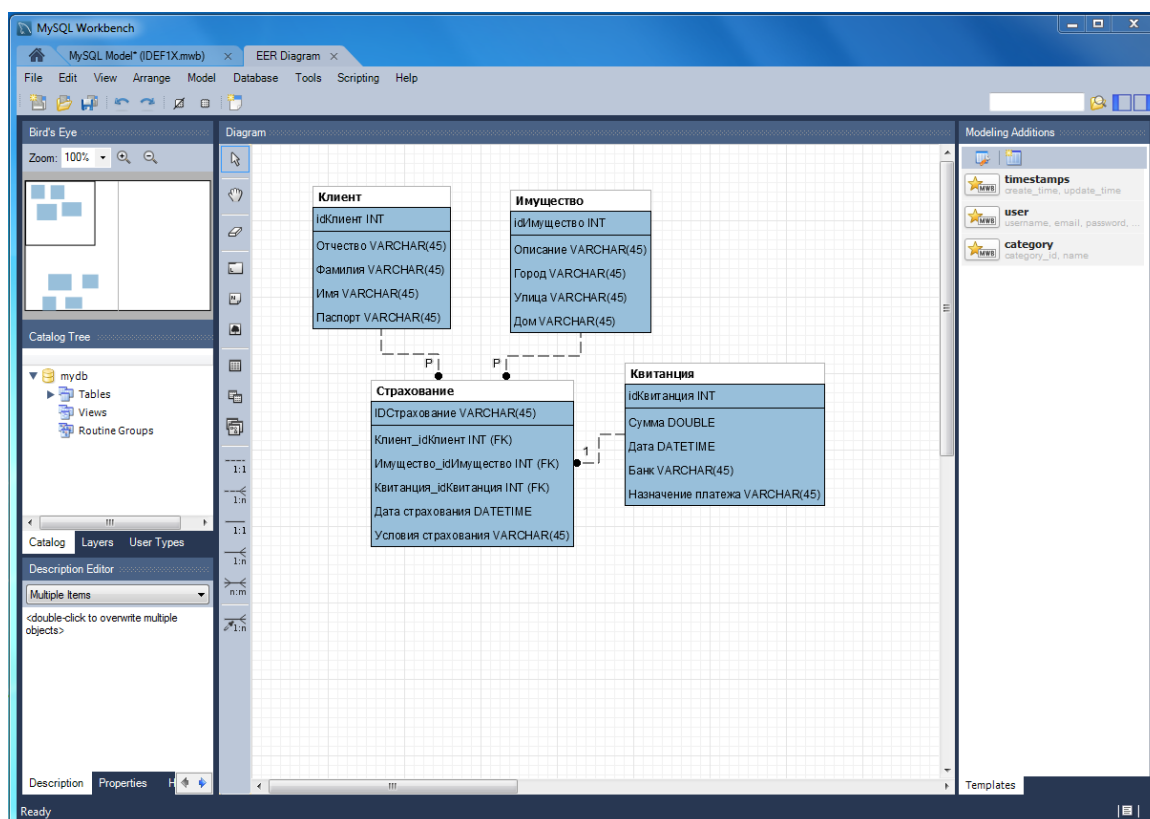


Рис. 5.10. Интерфейс MySQL Workbench

Программа позволяет визуально создавать, выполнять и оптимизировать SQL-запросы. SQL-редактор поддерживает синтаксическую подсветку и историю выполненных SQL-запросов. Панель подключений к базам данных даёт возможность легко переключаться между разными базами данных, что позволяет работать с несколькими БД одновременно. Просмотрщик объектов обеспечивает быстрый доступ к таблицам баз данных и объектам этих таблиц.

MySQL Workbench упрощает разработку и поддержку баз данных, автоматизирует выполнение наиболее долгих и сложных задач и улучшает взаимодействие между разработчиками и администраторами баз данных. Программа позволяет проектировщикам данных наглядно предоставить требования и, связавшись с коллегами, быстро решить проблему, до того как будет потрачено большое количество рабочих ресурсов и времени. С помощью MySQL Workbench можно легко создавать надёжные, хорошо

структурированные базы данных и в то же время достаточно гибкие для того, чтобы изменяться и улучшаться, отвечая новым требованиям бизнес-задач. Утилиты для проверки моделей данных и структур таблиц обеспечивают высокую надёжность при разработке. Это избавляет разработчика от ошибок во время моделирования новой ER-диаграммы или создания физической MySQL базы данных.

Главные преимущества программы:

- наглядное представление модели базы данных в графическом виде;
- наглядный и функциональный механизм установки связей между таблицами, в том числе многие-ко-многим с созданием таблицы связей;
- Reverse Engineering – восстановление структуры таблиц из уже существующей на сервере БД;
- удобный редактор SQL-запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы;
- возможность редактирования данных в таблице в визуальном режиме.

Контрольные вопросы

1. Назовите уровни методологии IDEF1X.
2. Каковы задачи, решаемые на этапе концептуального проектирования?
3. Дайте характеристику модели типа «сущность – связь».
4. Какое назначение имеет информационная (концептуальная) модель в процессе проектирования автоматизированной информационной системы?
5. В чем состоит отличие понятия типа сущности и элемента сущности?
6. Опишите правила формирования сущностей.
7. Какие типы связей можно выделить при реализации модели IDEF1X?
8. Какие можно выделить правила разложения связи типа много-ко-многим?
9. Чем отличается идентифицирующая связь от неидентифицирующей?
10. Как классифицируются атрибуты?

Задания для самостоятельного выполнения

Требуется разработать диаграмму IDEF1X для некоторой предметной области. Требования к содержанию диаграммы:

- минимум, одна связь многие-ко-многим любого типа;
- минимум, одна связь один-к-одному, идентифицирующая;
- минимум, одна связь один-к-одному, неидентифицирующая;
- минимум, одна связь один-ко-многим любого типа, независимая (не являющаяся результатом разложения связи многие-ко-многим);
- для каждой сущности должны быть определены, минимум, пять атрибутов без учета внешних ключей;
- диаграмма обязательно должна отражать специфику предметной области.

Пример диаграммы IDEF1X приведен в прил. А. Примеры предметной области приведены в прил. Б.

Глава 6

ПЛАНИРОВАНИЕ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

6.1. Общие принципы управления проектом

Создание информационной системы представляет собой сложный комплекс взаимосвязанных процессов, требующих использования различных видов ресурсов. При этом любой вид работ в конечном итоге сводится к организационной деятельности, основной задачей которой – создание не просто списка действий, позволяющего в конечном итоге получить готовый программный продукт, а установление четкой последовательности этих действий с привязкой ко времени. Такая совокупность распределенных во времени действий (процессов, операций, элементарных работ), направленных на достижение поставленной цели, называется проектом.

Проект – временное предприятие, предназначенное для создания уникальных продуктов или услуг. Проект обладает рядом собственных ему характеристик, определив которые, можно точно сказать, относится ли анализируемый вид деятельности к проектам:

- *временность* – любой проект имеет четкие временные рамки (это не относится к его результатам). В случае, если таких рамок не имеется, деятельность называется операцией и может длиться сколько угодно долго;

- *уникальные продукты, услуги, результаты* – проект должен порождать уникальные результаты, достижения, продукты; в противном случае такое предприятие становится серийным производством;

- *последовательная разработка* – любой проект развивается во времени, проходя через определенные ранее этапы или шаги, но при этом составление спецификаций проекта строго ограничивается содержанием, установленным на этапе инициации проекта.

Каждый проект характеризуется жизненным циклом, на основе которого формируется стандартный подход к проектному управлению. Жизненный цикл проекта – это промежуток времени между моментами его начала и завершения. Он делится на четыре фазы.

1. Концептуальная фаза. Включает формулирование целей, анализ инвестиционных возможностей, обоснование осуществимости (технико-экономическое обоснование) и планирование проекта.

2. Фаза разработки проекта. Включает определение структуры работ и исполнителей, построение календарных графиков работ, бюджета проекта, разработку проектно-сметной документации, переговоры и заключение контрактов с подрядчиками и поставщиками.

3. Фаза выполнения проекта. Включает работы по реализации проекта, в том числе строительство, маркетинг, обучение персонала и т.п.

4. Фаза завершения проекта. Включает в общем случае приемочные испытания, опытную эксплуатацию и сдачу проекта в эксплуатацию.

Результат проекта – это некоторая продукция или полезный эффект, создаваемые в ходе реализации проекта. В качестве результата, в зависимости от цели проекта, могут выступать: научная разработка, новый технологический процесс, программное средство, строительный объект, реализованная учебная программа, реструктурированная компания, сертифицированная система качества и т.д. Об успешности проекта судят по тому, насколько его результат соответствует по своим затратным, доходным, инновационным, качественным, временным, социальным, экологическим и другим характеристикам запланированному уровню.

Управляемыми параметрами проекта являются:

- объемы и виды работ;
- стоимость, издержки, расходы по проекту;
- временные параметры, включающие сроки, продолжительности и резервы выполнения работ и этапов проекта, а также взаимосвязи между работами;
- ресурсы, требуемые для осуществления проекта, в том числе человеческие или трудовые, финансовые, материально-технические, а также ограничения по ресурсам;
- качество проектных решений, применяемых ресурсов, компонентов проекта и прочее.

Управление проектом – это процесс планирования, организации и управления работами и ресурсами, направленный на достижение поставленной цели, как правило, в условиях ограничений на время, имеющиеся ресурсы или стоимость работ.

Управление проектом состоит из трех основных этапов:

- 1) формирование плана проекта;
- 2) контроль за реализацией плана и оперативная его коррекция;
- 3) завершение проекта.

На первом этапе осуществляется обоснование проекта, составляется перечень работ и имеющихся ресурсов, производится распределение ресурсов по работам и оптимизация плана по критериям времени завершения проекта, суммарной стоимости проекта, равномерного распределения ресурсов, минимизации рисков. Здесь

же производится заключение всех необходимых договоров со сторонними исполнителями, подрядчиками и поставщиками. Второй этап предполагает контроль выполнения проекта с целью своевременного выявления и устранения наметившихся отклонений от первоначального плана. При значительных отклонениях первоначальный план перерабатывается и составляется новый. Завершение проекта означает выполнение определенных регламентированных действий, необходимых для завершения и прекращения работ по проекту. Например, подписание акта приемки/сдачи выполненных работ.

В настоящее время для автоматизированного управления проектами используется методология сетевого планирования и управления. Сетевое планирование и управление состоит из структурного и календарного планирования и оперативного управления.

Структурное планирование заключается в разбиении проекта на этапы и работы, оценки их длительности, определении последовательности их выполнения. Результатом структурного планирования является сетевой график работ, который используется для оптимизации проекта по длительности.

Календарное планирование заключается в составлении временной диаграммы работ и распределении между работами трудовых ресурсов (исполнителей). Результатом календарного планирования является диаграмма Ганта, графически отображающая периоды выполнения работ на оси времени. На этом этапе может выполняться оптимизация ресурсов и бюджета проекта.

Оперативное управление состоит в регулярном сопоставлении фактического графика работ с плановым. Результатом серьезных отклонений является принятие решений об изменении первоначального структурного или календарного плана.

6.2. Структурное планирование

Структурное планирование включает в себя несколько этапов:

- 1) разбиение проекта на совокупность отдельных работ, выполнение которых необходимо для реализации проекта;
- 2) построение сетевого графика, описывающего последовательность выполнения работ;
- 3) оценка временных характеристик работ и анализ сетевого графика.

Основную роль на этапе структурного планирования играет сетевой график. Сетевой график – это ориентированный граф, в котором вершинами обозначены работы проекта, а дугами – временные взаимосвязи работ.

Сетевой график должен удовлетворять следующим свойствам.

1. Каждой работе соответствует одна и только одна вершина. Ни одна работа не может быть представлена на сетевом графике дважды. Однако любую работу можно разбить на несколько отдельных работ, каждой из которых будет соответствовать отдельная вершина графика.

2. Ни одна работа не может быть начата до того, как закончатся все непосредственно предшествующие ей работы, т.е. если в некоторую вершину входят дуги, то работа может начаться только после окончания всех работ, из которых выходят эти дуги.

3. Ни одна работа, которая непосредственно следует за некоторой работой, не может начаться до момента ее окончания. Другими словами, если из работы выходит несколько дуг, то ни одна из работ, в которые входят эти дуги, не может начаться до окончания этой работы.

4. Начало и конец проекта обозначены работами с нулевой продолжительностью. Такие работы называются вехами и обозначают начало или конец наиболее важных этапов проекта.

Предположим, что проект «Разработка информационной системы» состоит из работ, характеристики которых приведены на рис. 6.1.



Номер работы	Название	Дата начала	Дата окончания	Длительность
1	• Начало реализации проекта	11.02.19	11.02.19	0
2	• Постановка задачи	11.02.19	22.02.19	10
3	• Разработка интерфейса	25.02.19	01.03.19	5
4	• Разработка модулей обработки данных	05.03.19	13.03.19	7
5	• Разработка структуры базы данных	25.02.19	04.03.19	6
6	• Заполнение базы данных	05.03.19	14.03.19	8
7	• Отладка программного комплекса	15.03.19	21.03.19	5
8	• Тестирование и исправление ошибок	22.03.19	04.04.19	10
9	• Составление документации	22.03.19	28.03.19	5
10	• Завершение проекта	05.04.19	05.04.19	0

Рис. 6.1. Характеристики проекта

Сетевой график для данного проекта изображен на рис. 6.2. На нем вершины, соответствующие обычным работам, обведены тонкой линией, а толстой линией обведены вехи проекта.

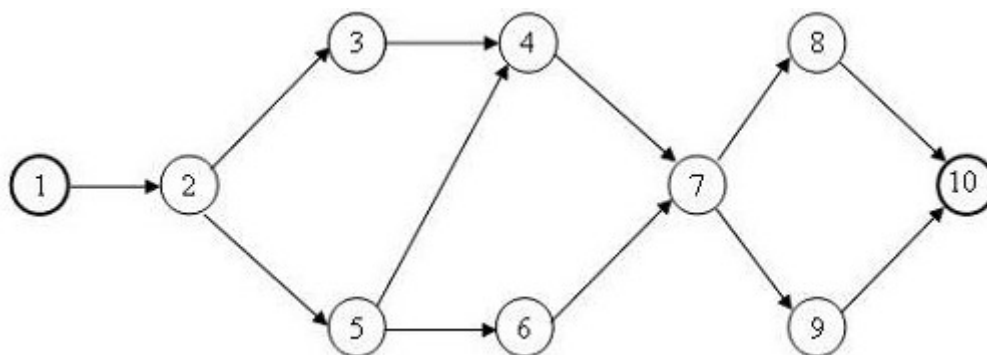


Рис. 6.2. Сетевой график проекта

Сетевой график позволяет определить временные характеристики входящих в проект работ. Наиболее важное значение при этом принимают критические работы, задержка начала которых приводит к задержке срока окончания проекта в целом. Такие работы не имеют запаса времени. Некритические работы имеют некоторый запас времени, и в пределах этого запаса их начало может быть задержано. Непрерывная последовательность критических работ определяет критический путь проекта (на рис. 6.3 отмечен пунктирной линией).

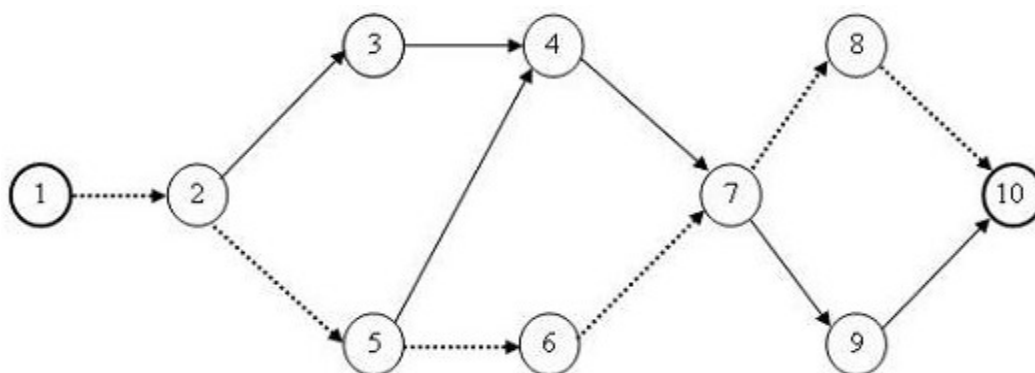


Рис. 6.3. Критический путь

Особенность критического пути состоит в том, что именно его длительность определяет минимальный срок осуществления проекта в целом. Сроки выполнения работ, лежащих вне критического пути, в той или иной степени «плавают», т.е. для таких работ всегда есть возможность либо увеличить длительность, либо начать с опозданием, на общей продолжительности проекта это никак не скажется.

Создающийся резерв времени можно использовать на самые различные цели: уменьшения риска невыполнения работы, оптимизацию расходования ресурсов, оптимизацию денежных потоков и т.д. Любые же задержки в выполнении работ, лежащих на критическом пути, непременно вызовут отставание от сроков исполнения проекта в целом.

В процессе управления ходом комплекса работ внимание сосредотачивается на главном направлении – на работах критического пути, сумма которых определяет длительность критического пути или срок выполнения проекта. Это позволяет наиболее целесообразно и оперативно контролировать ограниченное число работ, влияющих на срок разработки, а также лучше использовать имеющиеся ресурсы.

Для анализа критического пути проекта строится PERT-диаграмма (рис. 6.4), которая представляет собой сетевой график, т.е. граф, вершины которого отображают состояния некоторой системы, а дуги – работы, ведущиеся над этой системой. Такая диаграмма также предоставляет возможности для анализа времени, которое требуется для выполнения каждой отдельной задачи, а также помогает определить минимальное необходимое время для завершения проекта.

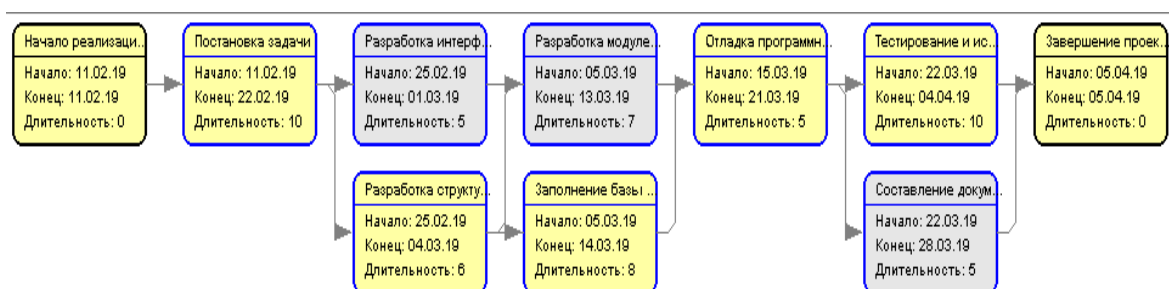


Рис. 6.4. PERT-диаграмма

Резерв времени события – это такой промежуток времени, на который может быть отсрочено наступление этого события без нарушения сроков завершения комплекса работ в целом. Резерв времени события определяется как разность между поздним и ранним сроками наступления события.

Поздний из допустимых сроков – это такой срок наступления события, превышение которого вызовет аналогичную задержку наступления завершающего события, т.е. если событие наступило в момент позднего из допустимых сроков, оно попало в критическую зону и последующие за ним работы должны находиться под таким же контролем, как и работы критического пути.

Ранний из возможных сроков наступления события – это срок, необходимый для выполнения всех работ, предшествующих данному событию. Это время находится путем выбора максимального значения из продолжительности всех путей, ведущих к данному событию.

Полный резерв времени пути – это разница между длиной критического пути и длиной рассматриваемого пути. Полный резерв показывает, насколько в сумме может быть увеличена продолжительность всех работ, принадлежащих пути, т.е. предельно допустимое увеличение продолжительности этого пути. Полный резерв времени пути может быть распределен между отдельными работами, находящимися на этом пути.

Полный резерв времени работы – это максимальный период времени, на который можно увеличить продолжительность данной работы, не изменяя при этом продолжительность критического пути.

Свободный резерв времени работы – максимальный период времени, на который можно увеличить ее продолжительность или отсрочить ее начало, не изменяя при этом ранних сроков последующих работ, при условии, что начальное событие этой работы наступило в свой ранний срок.

Возможности смещения сроков начала и окончания каждой работы определяются с помощью ранних и поздних сроков наступления событий, между которыми выполняется данная работа.

Этапы разработки и управления ходом работ с помощью сетевого графика имеют следующую последовательность основных операций:

- составление перечня всех действий и промежуточных результатов (событий) при выполнении комплекса работ и графическое их отражение;
- оценка времени выполнения каждой работы, а затем расчет сетевого графика для определения срока достижения поставленной цели;
- оптимизация рассчитанных сроков и необходимых затрат;
- оперативное управление ходом работ путем периодического контроля и анализа получаемой информации о выполнении заданий и выработка корректирующих решений.

Как правило, план-график проекта разрабатывается менеджером проекта с привлечением людей, которые являются экспертами в той или иной области. Результатом является полный перечень работ, структурированный по иерархическому признаку, т.е. структурная декомпозиция работ. Основными недостатками сетевого графика являются отсутствие возможности отображения временного масштаба и загроможденность.

6.3. Календарное планирование

На этапе календарного планирования с целью расчета сроков выполнения основных проектных операций, определения полной продолжительности проекта разрабатывается календарный план. В рамках календарного плана производится согласование сроков со всеми заинтересованными сторонами и утверждение соответствующих документов. Расписание в цепочке процессов планирования проекта тесно сопряжено с иерархической структурой работ, его бюджетом и матрицей ответственности.

Глубина проработки календарного плана зависит от того, как быстро находятся ресурсы под выполнение декомпозируемой задачи. Имеются в виду, в первую очередь, ответственные ресурсы за результаты задач в лице сотрудников компании или привлекаемых со стороны участников. Во вторую очередь, во внимание принимаются другие ресурсы: финансовые, материальные и информационные. Для определения степени детализации менеджеру проекта следует понимать, что он способен контролировать ходы выполнения всех задач и вытекающих из них действий. При этом избыточная детализация не нужна и даже вредна.

Создание календарного плана проекта – это определение длительностей работ и их взаимосвязей. Например, какие-то работы могут выполняться строго последовательно, а какие-то – параллельно друг с другом во времени. Для того, чтобы «увязать» сроки работ по проекту, их продолжительность и зависимости, используют простой и вместе с тем полезный инструмент календарного планирования – диаграмму Ганта. Диаграмма Ганта отображает следующие параметры проекта:

- структуру работ, полученную на основе сетевого графика;
- состав используемых ресурсов и их распределение между работами;
- календарные даты, к которым привязываются моменты начала и завершения работ.

Диаграмма Ганта, как одна из форм представления расписания календарного плана, инструментально позволяет достигать наилучшего качества оценки ресурсных составов и взаимосвязей работ. Это уменьшает потребности в изменениях в ходе последующей реализации проектных мероприятий.

Кроме составления перечня работ, календарное планирование включает в себя также создание ресурсной модели проекта.

Можно выделить два или три (в зависимости от степени абстракции) фундаментальных вида ресурсов, которые могут быть задействованы в проекте.

Это материалы (материальные или невозобновляемые ресурсы), а также рабочее время людей и оборудования (рабочие или возобновляемые ресурсы). Каждый из этих ресурсов обладает рядом особенностей, влияющих на их использование. К примеру, материальный ресурс имеет определенную стоимость. Точно так же ни люди, ни даже оборудование (если учесть амортизацию, затраты на ремонт, топливо, энергию) бесплатно не работают. Все эти затраты составляют бюджет проекта. Бюджет, как правило, жестко ограничен, что оказывает серьезное влияние на сроки исполнения проекта и саму его осуществимость. Если недостаток людей и оборудования в большинстве случаев приводит лишь к замедлению осуществления проекта, то недостаток средств на материальные ресурсы означает невозможность осуществления проекта в целом.

За каждой из работ «закрепляются» свои ресурсы. Один и тот же ресурс может одновременно быть связан с несколькими работами. Разумеется, такая ситуация легко может привести к «перегрузке» ресурса. Или наоборот – какой-то ресурс в определенный момент времени будет «недогружен». Использование сетевого планирования легко позволяет предвидеть и предотвращать подобные ситуации. В частности, такой эффект достигается «маневрированием» сроками и интенсивностью работ, лежащих вне критического пути.

Помимо стоимости, ресурсы обладают такими характеристиками, как:

- календарь (например, люди могут работать с 9.00 до 18.00 ч или по сменам: 12 ч каждая);
- затраты на использование (например, командировочные расходы менеджера проекта);
- максимальная доступность ресурса, измеряемая в процентах (например, менеджер проекта ведет одновременно два проекта, и в каждом из них он может быть занят на половину своего рабочего времени, т.е. из максимально доступных 100 % – по 50 % в каждом проекте).

Таким образом, календарное планирование – это итерационный процесс, позволяющий моделировать проект и получать в итоге оптимальный вариант календарного плана-графика проекта с оптимальными сроками.

Любое календарное планирование включает в себя:

- планирование содержания (scope) проекта и построение СДР – структурной декомпозиции работ, или WBS (Work Breakdown Structure);

- определение последовательности работ и построение сетевого графика;

- планирование сроков, длительностей и логических связей работ и построение диаграммы Ганта;

- определение потребности в ресурсах (люди, машины и механизмы, материалы и т.д.) и составление ресурсного плана проекта;

- расчет затрат и трудозатрат по проекту.

После создания плана-графика выполнения работ требуется назначение ресурсов на работы.

Возможно использование ресурсов двух типов: возобновляемые и невозобновляемые.

Возобновляемые, т.е. многократно используемые ресурсы, – это люди или оборудование.

Невозобновляемые ресурсы используются однократно. Это материальные ресурсы, например, бетон, лесоматериалы, гвозди, электроэнергия и т.п.

Ресурсное планирование проекта – это процесс назначения ресурсов работам проекта и связанное с ним редактирование варианта календарного графика.

Объем работ (трудозатраты) – общее количество трудового ресурса, необходимое для выполнения работы. Выражается в человеко-часах, человеко-днях, человеко-месяцах и т.д.

Объем назначений – общее количество единиц ресурса (объем ресурса), назначенных данной работе. Объем назначений может быть выражен в абсолютных единицах или в процентах. Так, например, для выполнения работы объем назначений «топ-менеджер – 50 %» означает, что ежедневно данный ресурс задействован в данной работе только половину своего рабочего дня.

Фиксированным является единовременное назначение работе некоторого материального ресурса, например, в начале работы. При изменяющемся назначении объем израсходованного ресурса является функцией времени.

Календарь ресурса – распределение рабочего и нерабочего времени для конкретного ресурса; календарь может быть задан только для возобновляемого ресурса; формат календаря ресурса идентичен формату стандартного календаря.

Доступность ресурса – это период рабочего времени, в течение которого он может быть запланирован для выполнения работы.

Доступность ресурса определяется:

- рабочим временем, установленным календарем ресурса;
- начальной и конечной датой использования ресурса;
- располагаемым количеством ресурса в данный период времени.

При ресурсном планировании проекта возможны два основных подхода:

– ресурсное планирование «от ресурсов» – распределение имеющихся ресурсов между работами с целью последующего выявления дефицитных и избыточных ресурсов;

– ресурсное планирование «от задач» – назначение работам требуемых ресурсов в необходимом количестве с целью определения общих потребностей в ресурсах различного типа.

Сведения о ресурсах проекта вводятся с помощью «Листа ресурсов», без указания для трудовых ресурсов максимального объема назначения.

Для каждого исполнителя в окне «Сведения о ресурсе» описывается график рабочего времени. Для каждого трудового ресурса задается максимальный объем назначения.

Общая стоимость (затраты) проекта складывается из фиксированной стоимости ресурсов и работ, а также стоимости назначений. Стоимость назначений на работы определяется ценами и тарифами для ресурсов (цены на материальные ресурсы и ставки оплаты труда).

При календарном планировании можно учитывать следующие типы затрат:

- нормированные затраты;
- затраты на использование;
- фиксированные затраты.

Нормированные затраты на ресурс – это затраты на трудовые ресурсы, например, люди или арендуемое оборудование, для которых назначаются стандартная ставка и (при необходимости) ставка сверхурочных. Такие ресурсы обычно рассчитываются на почасовой основе. При назначении ресурса задаче рассчитываются итоговые затраты на ресурс с использованием указанных почасовых ставок затрат на ресурс и времени (или длительности) выполнения задачи.

Нормированные затраты на материалы – это затраты на расходные материальные ресурсы, такие как отделочные материалы или принадлежности, которым назначаются стандартные нормы. Нормы материальных ресурсов назначаются единице материала (например, норма на метр или норма на тонну).

Стоимость ресурса может изменяться в ходе реализации проекта, или ресурс может оплачиваться по различным ценам для различных работ, например, при работе с различными сортами материала.

Затраты на использование – это разовая плата за пользование ресурсом. Затраты на использование никогда не зависят от объема выполняемых работ. Для трудовых ресурсов такие затраты начисляются при каждом использовании ресурса и зависят от числа назначений, а для материальных ресурсов – только один раз.

Если затраты на работу точно известны, например, затраты на оборудование, то можно ввести фиксированную стоимость работы в поле Фиксированные затраты. Фиксированные затраты – это установленные затраты, которые остаются постоянными независимо от длительности работы, величины выполненного объема трудозатрат или количества назначенного материального ресурса.

Планируя стоимость проекта, необходимо для каждого ресурса определить метод начисления затрат. Метод начисления затрат определяет момент времени, когда следует учесть стоимость ресурса.

Построение календарного графика рассмотрим на примере проекта «Разработка информационной системы». Прежде всего, нужно определиться с ресурсами, которые будут использоваться этим проектом. Предположим, что в качестве ресурсов выступают только исполнители и они распределены между работами согласно рис. 6.5.



Номер	Имя	Стандартная роль	Стандартная ставка
1	Руководитель проекта	управляющий проектом	350
	Постановка задачи		
	Составление документации		
2	Проектировщик	аналитик	200
	Постановка задачи		
	Разработка интерфейса		
	Разработка структуры базы данных		
3	Программист	разработчик	250
	Разработка интерфейса		
	Разработка модулей обработки данных		
	Разработка структуры базы данных		
	Заполнение базы данных		
	Отладка программного комплекса		
	Тестирование и исправление ошибок		
4	Тестировщик	разработчик	150
	Отладка программного комплекса		
	Тестирование и исправление ошибок		
	Составление документации		

Рис. 6.5. Распределение ресурсов

Календарный график (диаграмма Ганта) изображен на рис. 6.6, где ромбиками обозначены вехи, прямоугольными блоками – решаемые задачи, сплошными линиями со стрелками – связи между соответствующими задачами, определяющие последовательность выполнения работ.

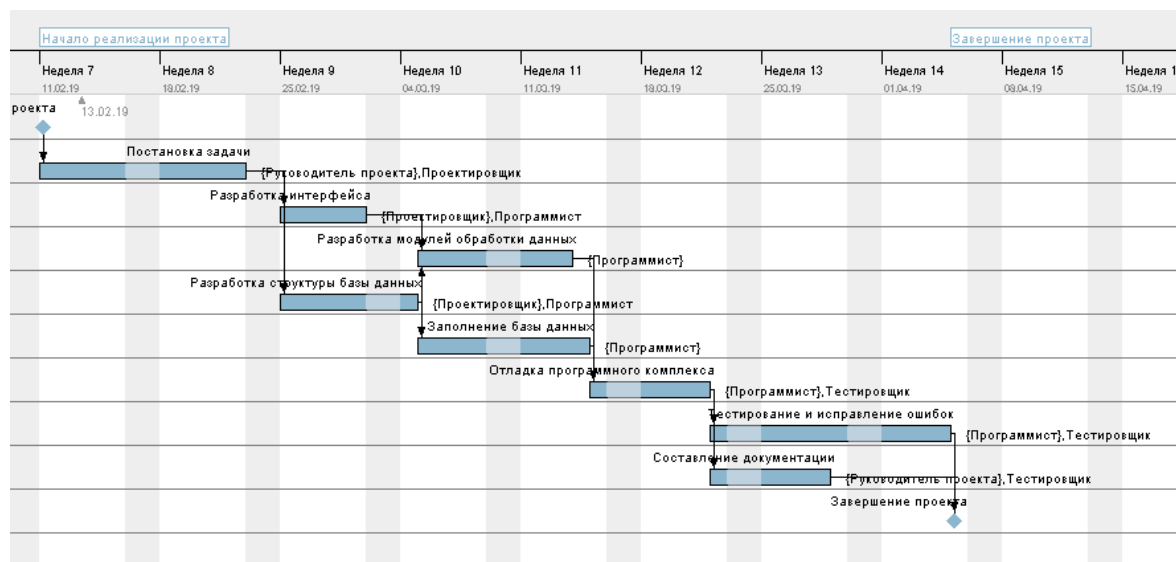


Рис. 6.6. Диаграмма Ганта

На основании диаграммы Ганта может быть построен график загрузки ресурсов. Этот график показывает процент загрузки конкретного трудового ресурса в ходе выполнения проекта. По оси абсцисс откладывается временной интервал проекта, а по оси ординат – суммарный процент загрузки исполнителя по всем задачам проекта, которые он выполняет в текущий момент времени.

Обычно исполнитель целиком занят решением некоторой задачи и по ее завершении переходит к следующей. Это соответствует 100 % загрузки. Однако в некоторых случаях он может быть параллельно задействован в двух или более задачах, выделяя для их решения часть рабочего времени. Например, две задачи – по 50 % каждая, т.е. по половине рабочего дня на задачу. График загрузки ресурса позволяет в этом случае контролировать суммарную занятость исполнителя и выявить возможные периоды перегрузки, когда ему запланировано больше работы, чем он может выполнить в течение рабочего дня. Об этом свидетельствует суммарная загрузка более 100 %.

Пример графиков загрузки ресурсов проекта «Разработка информационной системы» изображен на рис. 6.7. Он построен, исходя из предположения, что каждый работник в состоянии

совмещать выполнение, как минимум, двух параллельных задач в рамках одного проекта.

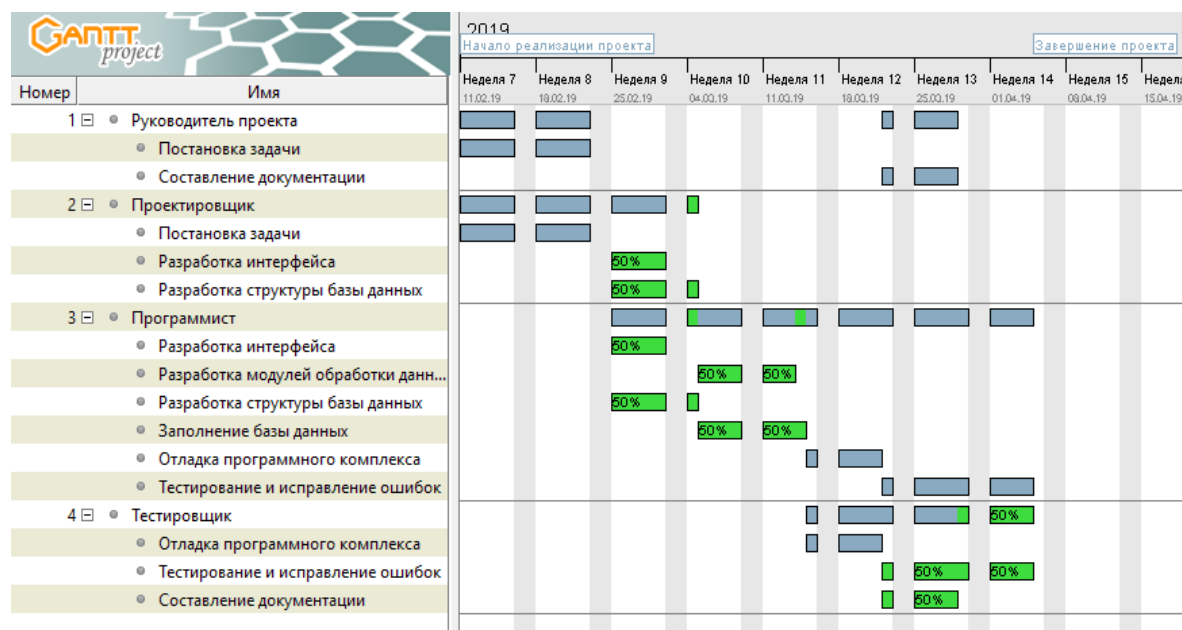


Рис. 6.7. Графики загрузки ресурсов

Из данных графиков видно, что «Проектировщик» вынужден на 9-й неделе выполнения проекта совмещать работу по разработке интерфейса и разработку структуры БД для того, чтобы уложиться в график разработки. При этом затраты по времени у проектировщика будут распределяться равномерно между указанными задачами. По аналогичному графику будут задействованы также «Программист» и «Тестировщик».

6.4. Оперативное управление

На этапе оперативного управления происходят выполнение работ по проекту и непрерывный контроль над ходом его реализации. Каким бы хорошим ни был первоначальный план, жизнь обязательно внесет в него свои коррективы. Поэтому задачами менеджера проекта являются:

- отслеживание фактического графика выполнения работ;
- сравнение фактического графика с плановым;
- принятие решений по ликвидации наметившихся отклонений от плана;
- перепланирование проекта в случае значительных отклонений.

Первые две задачи решаются при помощи диаграммы Ганта. На ней параллельно линиям продолжительности работ наносятся линии, обозначающие процент фактического выполнения этих работ. Это позволяет легко обнаружить возникшие отклонения.

Метод ликвидации отклонения зависит от имеющихся в распоряжении менеджера ресурсов. Для завершения запаздывающей работы можно либо привлечь дополнительных работников (дополнительные ресурсы), либо использовать тот же состав работников в сверхурочном режиме. В обоих случаях за ликвидацию отклонения придется платить увеличением стоимости проекта (не запланированная ранее оплата дополнительных работников, ресурсов и сверхурочных работ).

Если же отклонение таково, что не может быть исправлено привлечением дополнительных и сверхурочных ресурсов, или увеличение стоимости проекта недопустимо, нужно заново перепланировать проект и выполнить следующие действия:

- завершенным работам приписываются нулевые значения длительности;
- для частично выполненных работ устанавливаются значения длительности, соответствующие оставшемуся объему работ;
- в сетевой график вносятся структурные изменения с целью ликвидации оказавшихся ненужными работ и добавления других, ранее не запланированных;
- повторный расчет критического пути и повторное календарное планирование проекта.

После создания скорректированного проекта он утверждается руководством, начинаются его реализация и оперативное управление. Такая корректировка может выполняться несколько раз.

6.5. Программное обеспечение для планирования разработки информационной системы

Microsoft Office Project

Microsoft Office Project – это платное программное обеспечение для управления проектами, которое разработано корпорацией Microsoft и предназначено для менеджеров проектов с целью помочь им в разработке календарных планов, распределении ресурсов и финансов по задачам, отслеживании хода исполнения проекта и анализе объемов работ.

Microsoft Project позволяет эффективно управлять проектом на различных этапах его реализации (рис. 6.8). Он дает возможность

выполнить структуризацию проекта путем разделения его на этапы, задачи и подзадачи, выявить критические задачи (задачи, длительность которых существенно влияет на длительность реализации всего проекта), получить сетевой график и календарный план проекта, осуществить назначение ресурсов задачам проекта, эффективно контролировать загрузку ресурсов. Пакет поддерживает все необходимые типы связей между задачами: FS (Finish-Start), SS (Start-Start), FF (Finish-Finish).

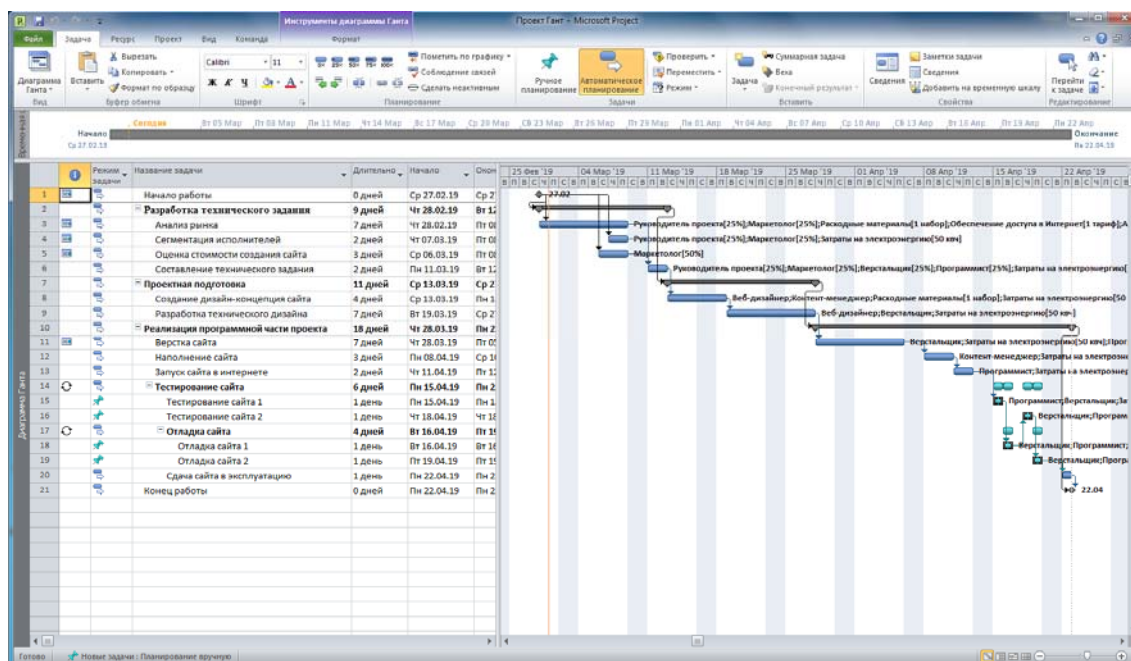


Рис. 6.8. Интерфейс Microsoft Project

Методика использования пакета Microsoft Project для управления IT проектом на этапе подготовки к реализации, целью которой является получение сетевого графика и календарного плана проекта, может быть представлена в виде последовательности следующих шагов:

- создание календаря проекта (т.е. учет нерабочих и праздничных дней);
- составление списка задач, которые надо выполнить для успешной реализации проекта;
- определение связей между задачами;
- выявление задач, длительность реализации которых существенно влияет на длительность реализации всего проекта, и, возможно, изменение порядка выполнения задач проекта;
- формирование списка доступных для реализации проекта ресурсов;

– распределение ресурсов (назначение ресурсов конкретным задачам проекта).

Наиболее очевидным преимуществом продукта является то, что он входит в семейство Microsoft Office. Это обеспечивает следующие плюсы, характерные для всех продуктов MS Office:

– такое же малое время обучения пользователей, как и с остальными программами Microsoft Office;

– богатые возможности по настройке в стиле формул Microsoft Excel (сам продукт выдержан в интерфейсе, максимально приближенном к Microsoft Excel);

– возможность адаптировать продукт под свою специфику путём программирования или покупки готовых решений, созданных на базе Visual Basic или Microsoft .Net.

GanttProject

GanttProject – бесплатное кросс-платформенное программное обеспечение, предназначенное для планирования проектов и управления задачами и ресурсами с применением диаграмм Ганта в качестве основного инструмента планирования (рис. 6.9).

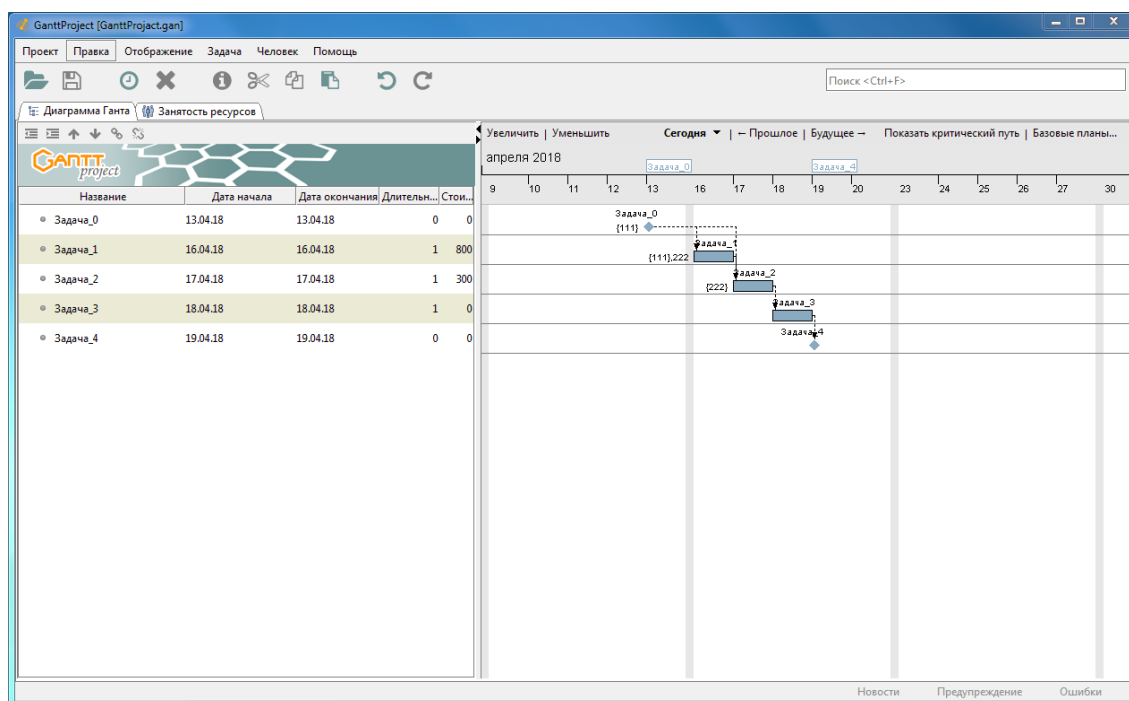


Рис. 6.9. Интерфейс GanttProject

Помимо диаграмм Ганта, программа позволяет выстраивать диаграммы типа PERT. Поскольку практически весь софт этого ряда в той или иной мере является альтернативой платному продукту

от Microsoft (Ms Project), важно, что Gantt Project Planner, созданная на Java, поддерживает импорт-экспорт документов Microsoft Project. При этом альтернативная программа распространяется бесплатно на правах Открытого лицензионного соглашения GPL.

Данный софт для планирования проектов может устанавливаться на операционные системы Linux, Windows, MacOSX. Совместимость с другими (сторонними) проектами обеспечивает использование библиотеки MPXJ. Однако Gantt Project поддерживает только один из пяти доступных в библиотеке форматов – MPX (Microsoft Project Exchange). Импортируются и экспортируются файлы в форматах .txt и .xml. Отчёты сохраняются в форматах .html и .pdf, а изображения – в форматах .png, .jpeg или .csv, по выбору, что даёт возможность впоследствии применять программы для работ с электронными таблицами. Поскольку программа рассчитана, в первую очередь, на однопользовательское применение, бонусом становится возможность загрузки-сохранения проектного файла на FTP, позволяющая открывать один и тот же документ нескольким пользователям. Однако одновременное редактирование может создать содержательные проблемы, которые предполагается разрешать вручную.

Планируемый в программе проект представляется в виде дерева задач, для выполнения каждой из которых пользователь назначает исполнителя. На каждую задачу отводится определённое время. Чтобы запланированные процессы могли сформировать единый логически цельный проект, между задачами устанавливаются зависимости и связи, например, в формате: «Не начинать задание Y до окончания выполнения задания X». Для вывода информации с привязкой к календарю используют два типа диаграмм – Ганта и PERT.

Контрольные вопросы

1. Что такое проект, и какими свойствами он обладает?
2. Что такое жизненный цикл проекта, и каковы его фазы?
3. Какие этапы входят в методологию структурного планирования?
4. Что такое сетевой график, и какими свойствами он обладает?
5. Что такое критическая работа и критический путь?
6. Что такое диаграмма Ганта?
7. Для чего предназначен график загруженности ресурсов?
8. Каким образом по графику загруженности ресурсов можно найти перегрузку ресурса?

9. В чем сущность процесса оперативного управления?

10. Какие действия следует выполнить при перепланировании проекта в процессе оперативного управления?

Задания для самостоятельного выполнения

Требуется разработать диаграмму Ганта для некоторой предметной области. Требования к содержанию диаграммы:

- период планирования проекта – 1 месяц;
- минимум, по пять ресурсов каждого типа (трудовые, материальные, затраты);
- минимум, три комплексные задачи (этапа);
- обязательное использование последовательных, параллельных и циклических задач;
- не допускается использование ресурсов с превышением доступности;
- диаграмма обязательно должна отражать специфику предметной области.

Пример диаграммы Ганта приведен в прил. А. Примеры предметной области приведены в прил. Б.

ЗАКЛЮЧЕНИЕ

Информационные технологии (ИТ) в жизни современного общества используются повсеместно. Трудно представить работу большинства современных организаций без использования прогрессивных достижений в различных областях науки и техники. Именно внедрение ИТ меняет облик и содержание всех сфер жизнедеятельности человека, ускоряя динамику развития общества.

Информационные технологии являются одной из наиболее динамично развивающихся областей знаний. Тенденции развития информационных технологий определены постоянным возрастанием сложности программного обеспечения. Технология создания программных систем и комплексов систем предъявляет особые требования к методикам реализации и программным инструментальным средствам.

Специфика технологического знания определяет его высокую востребованность в процессе формирования профессиональных умений и навыков. Для ИТ-специалистов уровень владения современными информационными технологиями определяет их профессиональную состоятельность. Данное учебное пособие направлено на формирование единой концепции современных информационных технологий в области проектирования программных систем и комплексов. Теоретический материал дополнен практическими примерами используемых методологий в различных программных средах.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Буч, Грейди. Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. – Москва : ДМК, 2015. – 432 с.
2. Вендров, А. М. CASE-технологии. Современные методы и средства проектирования информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 2005. – 98 с.
3. Вичугова, А. А. Методы и средства концептуального проектирования информационных систем: сравнительный анализ структурного и объектно-ориентированного подходов / А. А. Вичугова. – Москва : Синергия, 2014. – 631 с.
4. Гаврилов, М. В. Информатика и информационные технологии : учеб. для бакалавров / М. В. Гаврилов, В. А. Климов. – Москва : Юрайт, 2013. – 378 с.
5. Голицына, О. Л. Информационные технологии : учебник / О. Л. Голицына, Н. В. Максимов, Т. Л. Партыка, И. И. Попов. – Москва : ФОРУМ : ИНФРА-М, 2013. – 608 с.
6. Гришин, В. Н. Информационные технологии в профессиональной деятельности : учебник / В. Н. Гришин, Е. Е. Панфилова. – Москва : ФОРУМ : ИНФРА-М, 2013. – 416 с.
7. Информационные технологии : учебник / Ю. Ю. Громов, И. В. Дидрих, О. Г. Иванова, М. А. Ивановский, В. Г. Однолько. – Тамбов : Изд-во ТГТУ, 2015. – 260 с.
8. Исаев, Г. Н. Информационные технологии : учеб. пособие / Г. Н. Исаев. – Москва : Омега-Л, 2013. – 464 с.
9. Коцюба, И. Ю. Основы проектирования информационных систем : учеб. пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.
10. Максимов, Н. В. Информационные технологии в профессиональной деятельности : учеб. пособие / Н. В. Максимов, Т. Л. Партыка, И. И. Попов. – Москва : ФОРУМ, 2010. – 496 с.
11. Максимов, Н. В. Современные информационные технологии : учеб. пособие / Н. В. Максимов, Т. Л. Партыка, И. И. Попов. – Москва : ФОРУМ, 2013. – 512 с.
12. Мельников, В. П. Информационные технологии : учеб. для студентов высших учебных заведений / В. П. Мельников. – Москва : ИЦ Академия, 2009. – 432 с.
13. Осетрова, И. С. Управление проектами в Microsoft Project : учеб. пособие / И. С. Осетрова. – Санкт-Петербург : НИУ ИТМО, 2013. – 69 с.

14. Румянцева, Е. Л. Информационные технологии : учеб. пособие / Е. Л. Румянцева, В. В. Слюсарь ; под ред. Л. Г. Гагарина. – Москва : ФОРУМ : ИНФРА-М, 2013. – 256 с.

15. Светлов, Н. М. Информационные технологии управления проектами : учеб. пособие / Н. М. Светлов, Г. Н. Светлова. – Москва : ИНФРА-М, 2012. – 232 с.

16. Советов, Б. Я. Информационные технологии : учеб. для бакалавров / Б. Я. Советов, В. В. Цехановский. – Москва : Юрайт, 2013. – 263 с.

17. Федотова, Е. Л. Информационные технологии в профессиональной деятельности : учеб. пособие / Е. Л. Федотова. – Москва : ФОРУМ : ИНФРА-М, 2012. – 368 с.

18. Федотова, Е. Л. Информационные технологии и системы : учеб. пособие / Е. Л. Федотова. – Москва : ФОРУМ : ИНФРА-М, 2013. – 352 с.

19. Хлебников, А. А. Информационные технологии : учебник / А. А. Хлебников. – Москва : КноРус, 2014. – 472 с.

ПРИМЕРЫ ВЫПОЛНЕНИЯ ДИАГРАММ

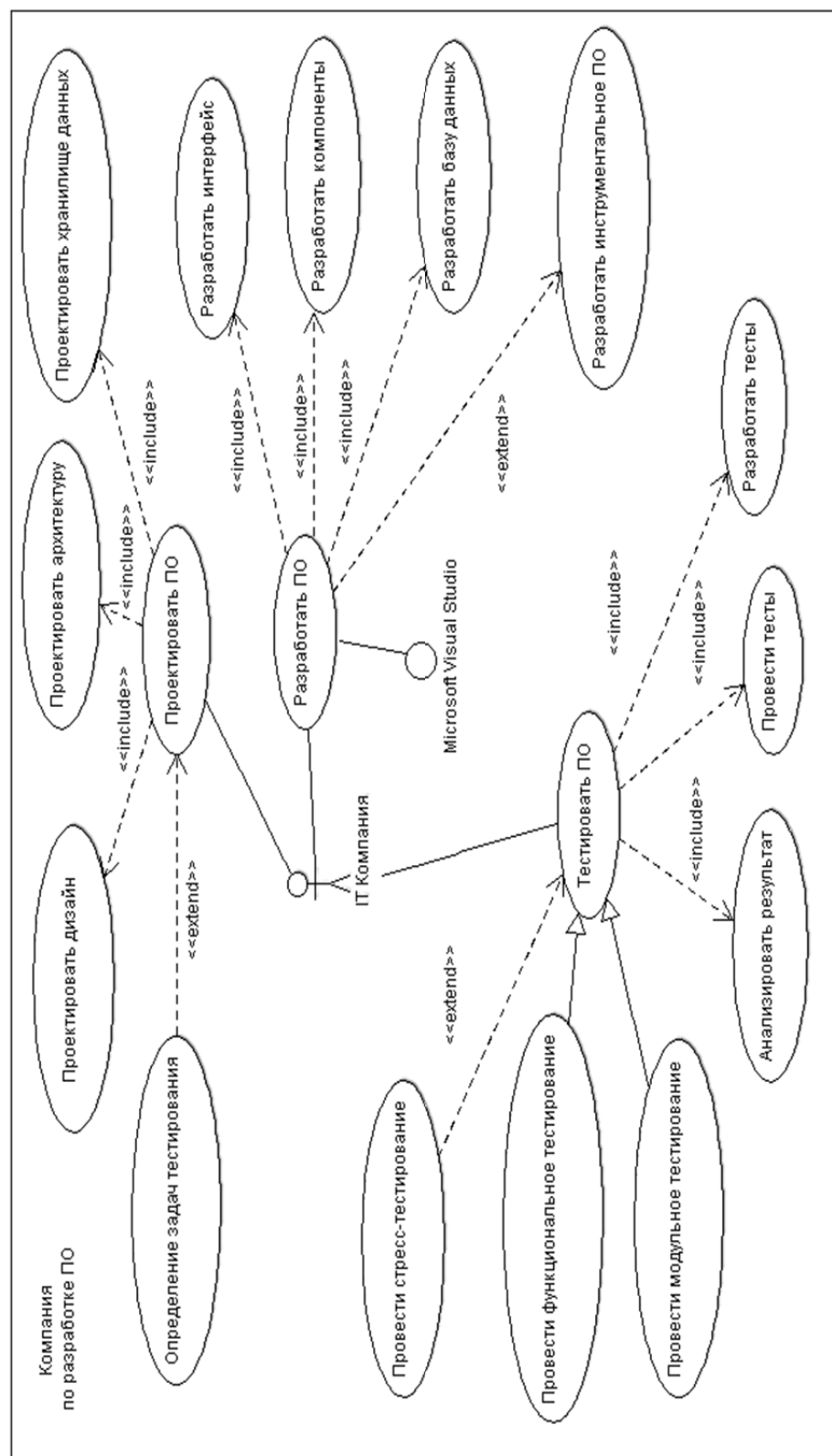


Рис. П.А.1. Диаграмма вариантов использования для предметной области «Компания по разработке ПО»

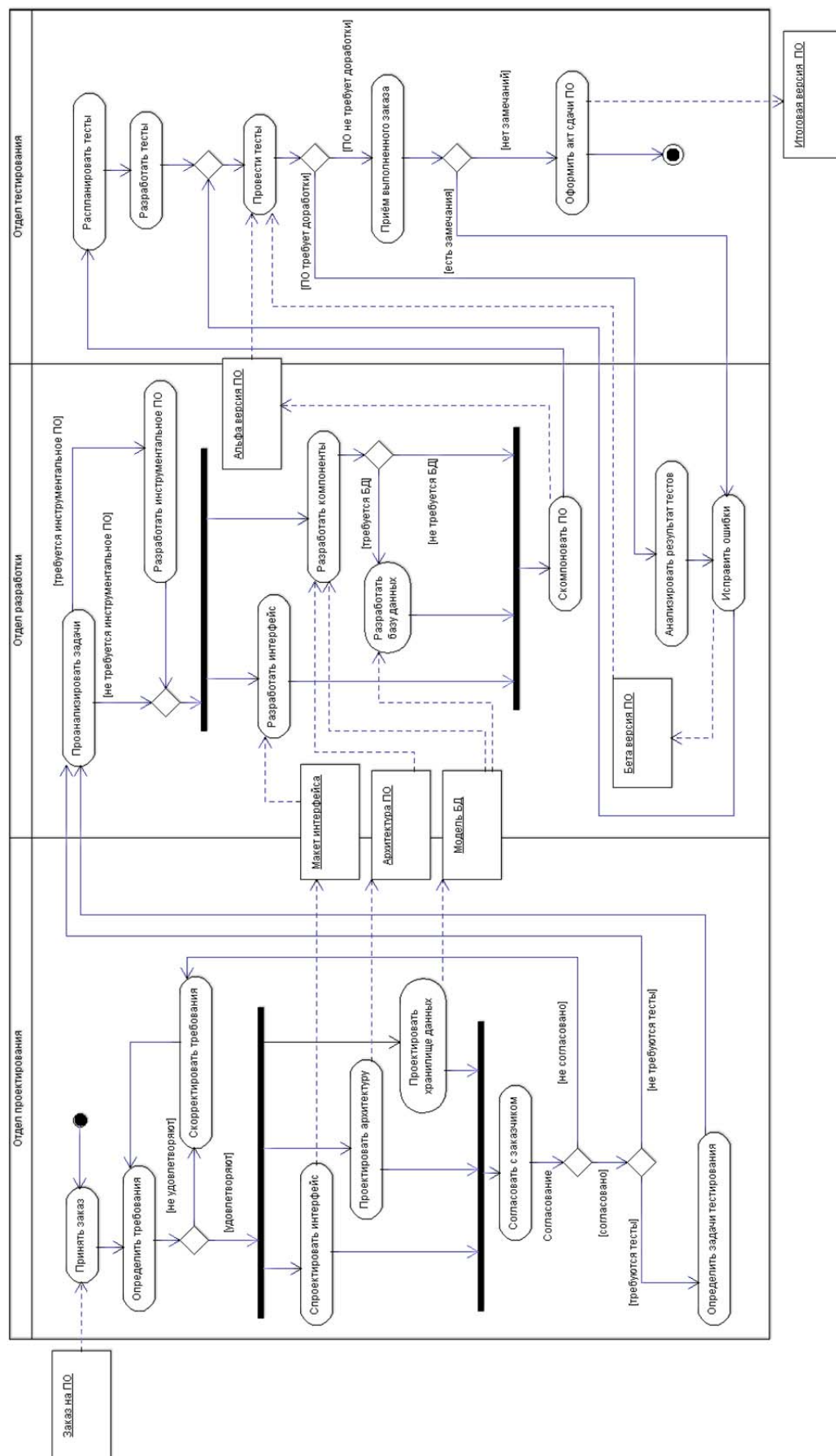


Рис. П.А.2. Диаграмма деятельности для предметной области «Компания по разработке ПО»

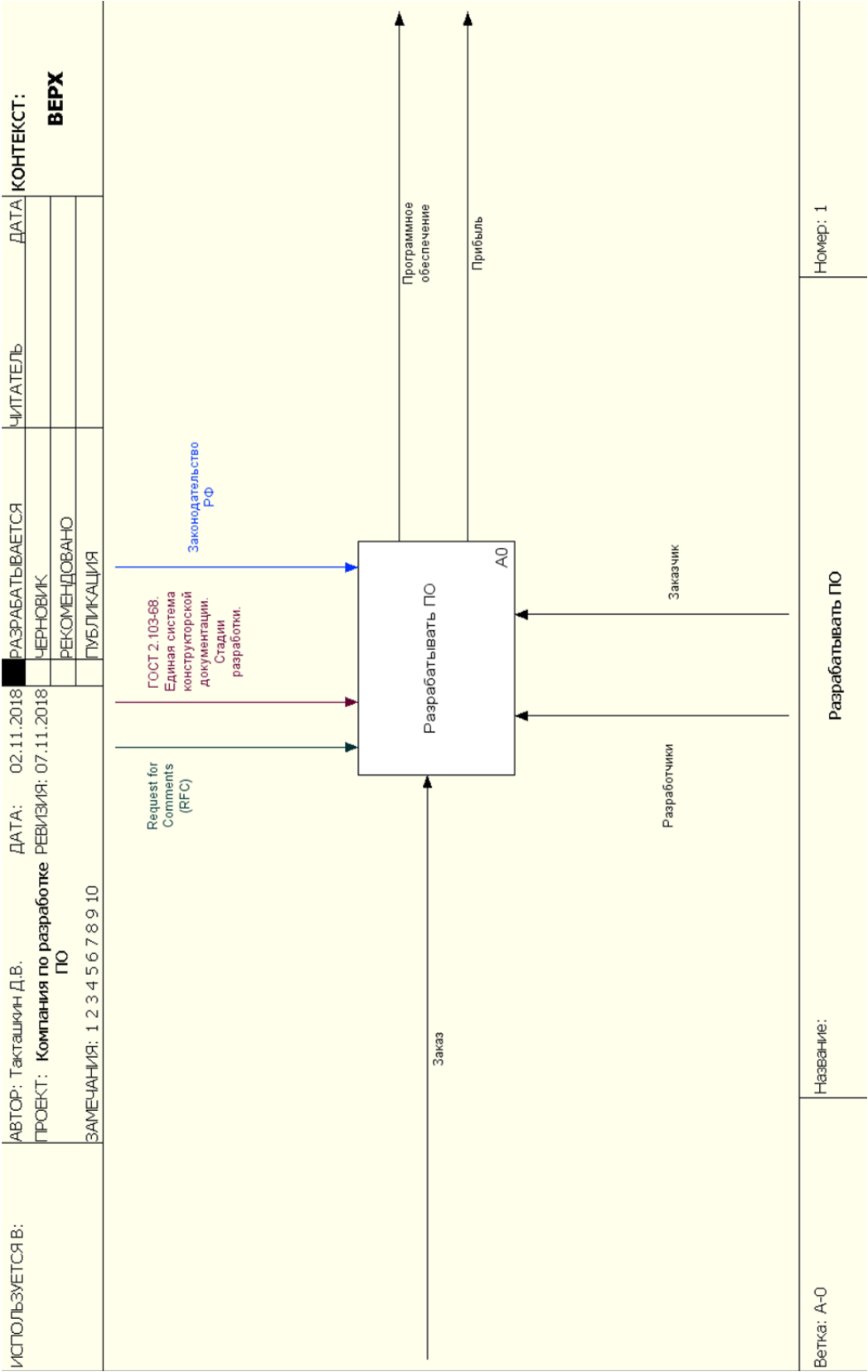


Рис. П.А.3. Диаграмма IDEF0 (контекстная) для предметной области «Компания по разработке ПО»

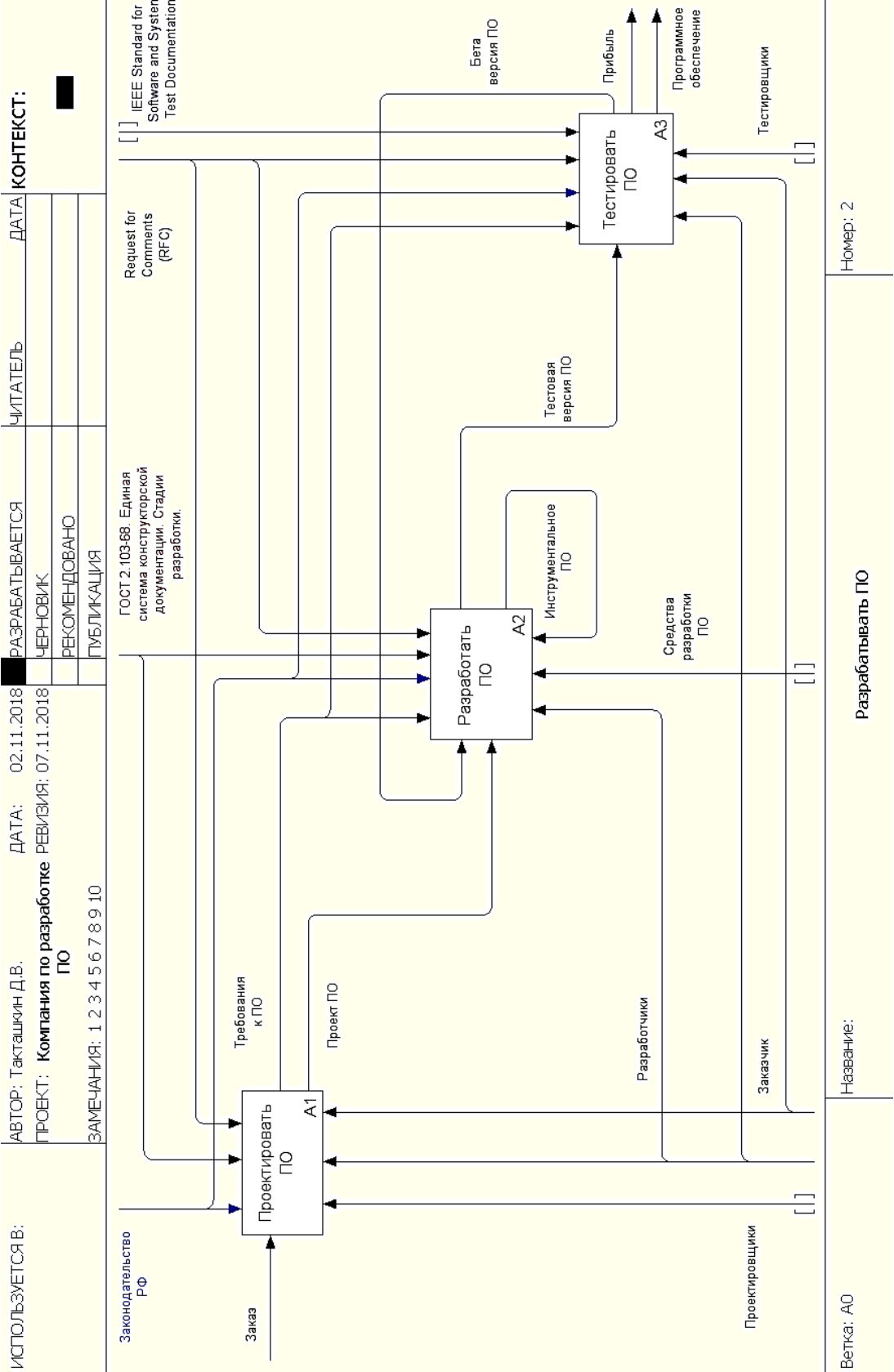


Рис. П.А.4. Диаграмма декомпозиции функционального блока «Разрабатывать ПО»

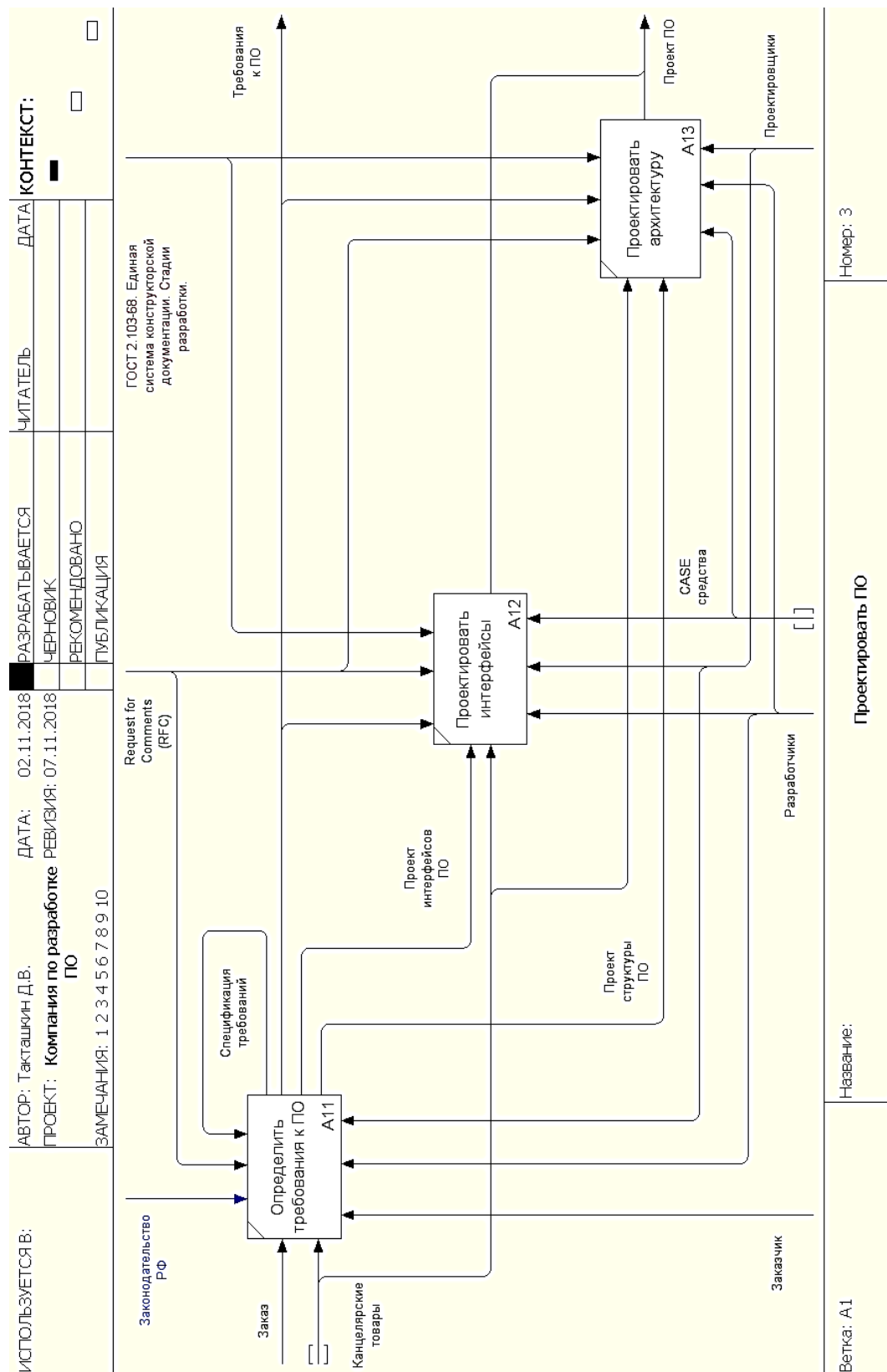


Рис. П.А.5. Диаграмма декомпозиции функционального блока «Проектировать ПО»

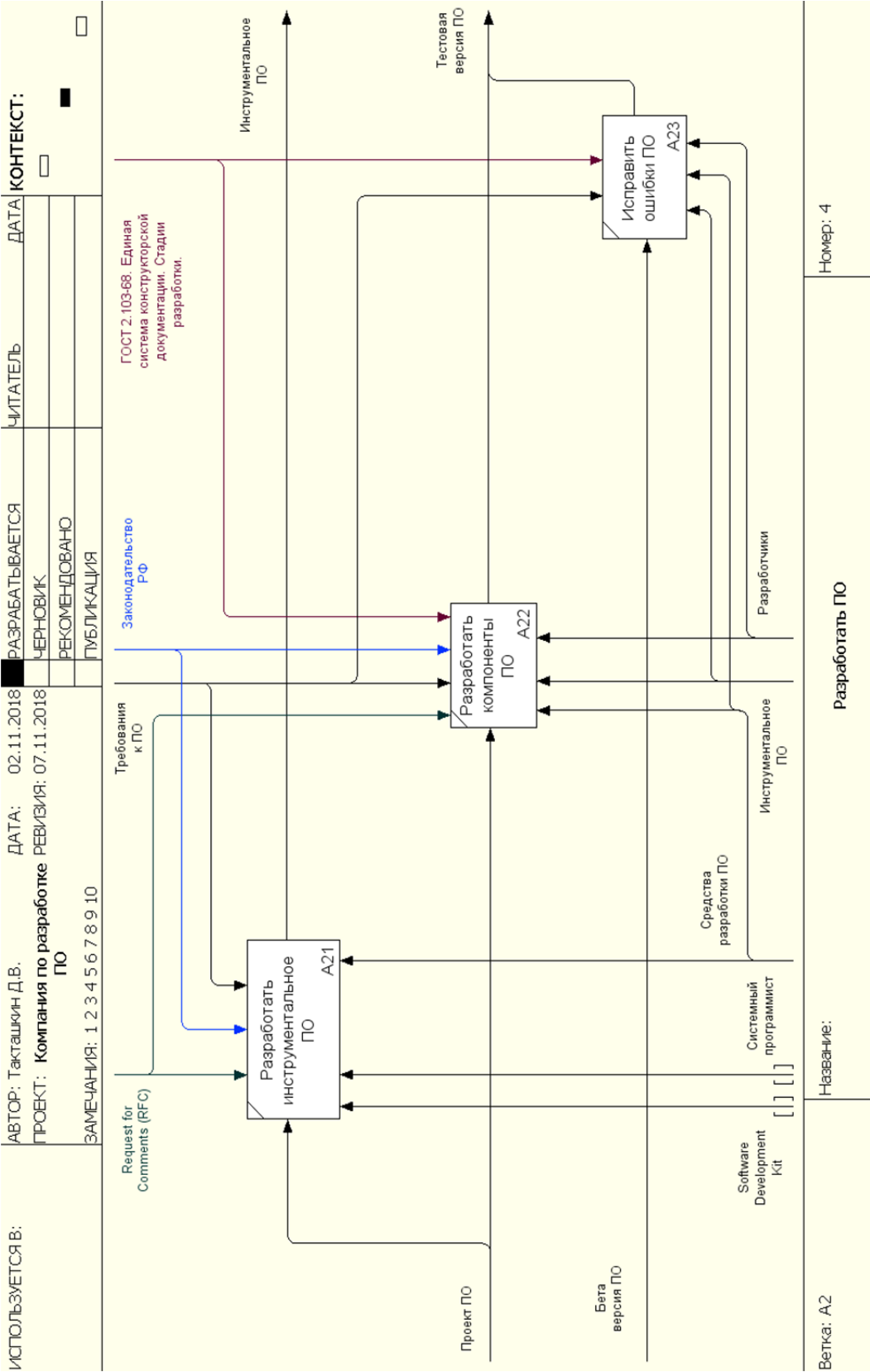


Рис. П.А.6. Диаграмма декомпозиции функционального блока «Разработать ПО»

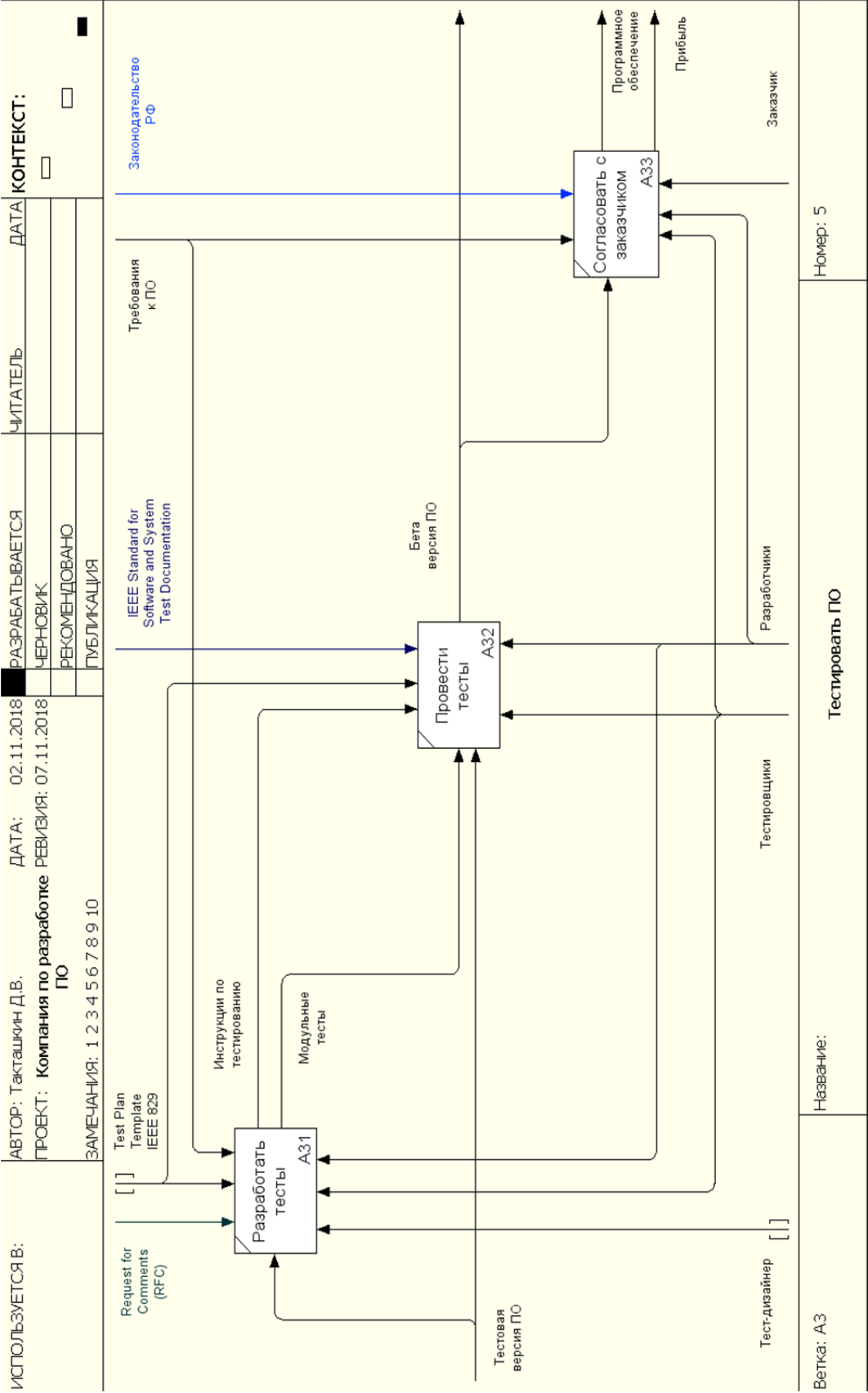


Рис. П.А.7. Диаграмма декомпозиции функционального блока «Тестировать ПО»

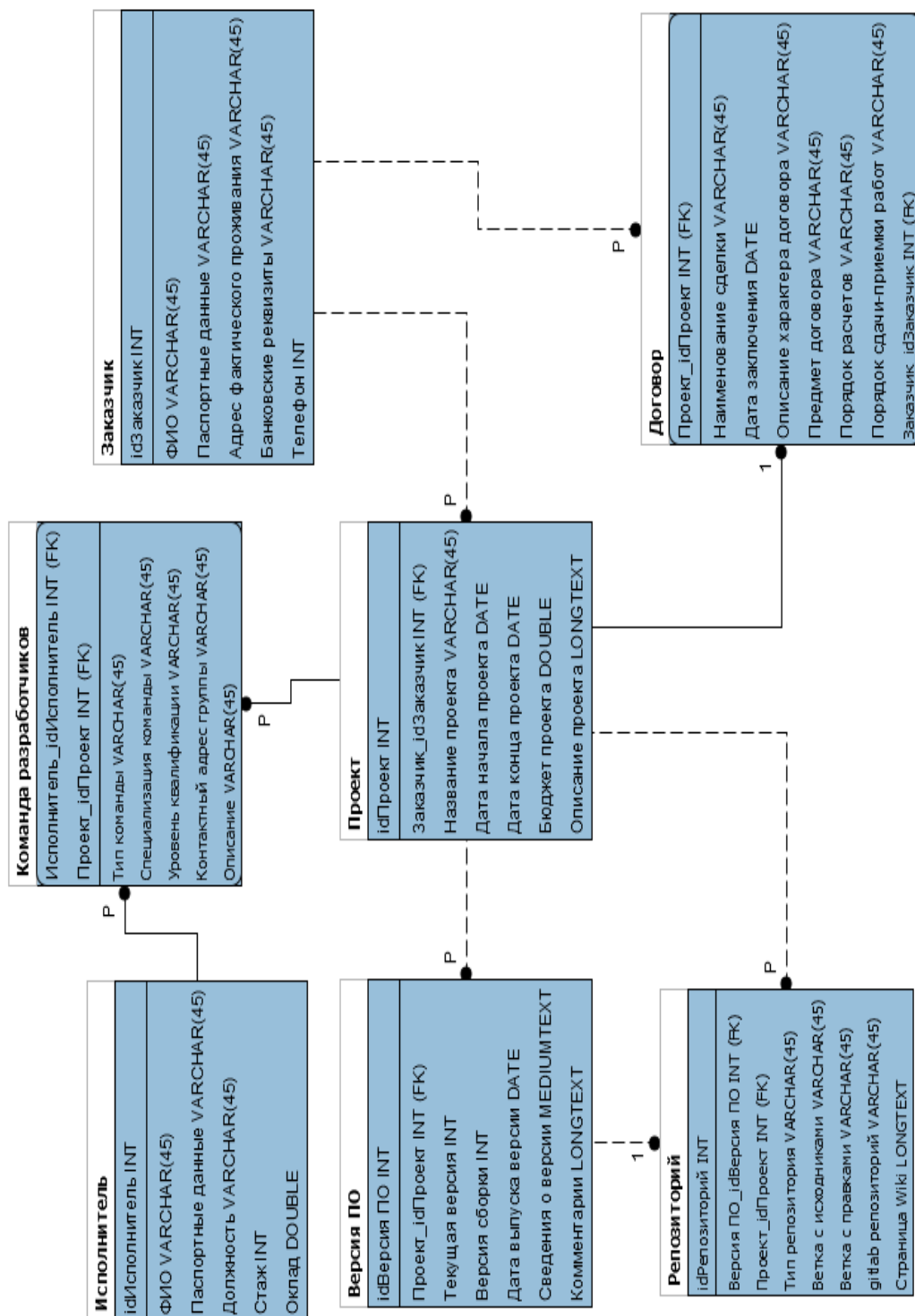


Рис. П.А.8. Диаграмма IDEF1X для предметной области «Компания по разработке ПО»

Название задачи	Длительно	Начало	Окончание	Предш	Названия ресурсов
Начало работы	0 дней	Ср 27.02.19	Ср 27.02.19		
Разработка технического задания	9 дней	Чт 28.02.19	Вт 12.03.19	1	
Анализ рынка	7 дней	Чт 28.02.19	Пт 08.03.19	1	Руководитель проект
Сегментация исполнителей	2 дня	Чт 07.03.19	Пт 08.03.19	1	Руководитель проект
Оценка стоимости создания сайта	3 дня	Ср 06.03.19	Пт 08.03.19	1	Маркетолог[50%]
Составление технического задания	2 дня	Пн 11.03.19	Вт 12.03.19	3;4;5	Руководитель проект
Проектная подготовка	11 дней	Ср 13.03.19	Ср 27.03.19	2	
Создание дизайн-концепции сайта	4 дня	Ср 13.03.19	Пн 18.03.19	6	Веб-дизайнер;Контент
Разработка технического дизайна	7 дней	Вт 19.03.19	Ср 27.03.19	8	Веб-дизайнер;Верстка
Реализация программной части проекта	18 дней	Чт 28.03.19	Пн 22.04.19	7	
Верстка сайта	7 дней	Чт 28.03.19	Пт 05.04.19	9	Верстальщик;Затраты
Наполнение сайта	3 дня	Пн 08.04.19	Ср 10.04.19	11	Контент-менеджер;Затраты
Запуск сайта в интернете	2 дня	Чт 11.04.19	Пт 12.04.19	12	Программист;Затраты
Тестирование сайта	6 дней	Пн 15.04.19	Пн 22.04.19		
Тестирование сайта 1	1 день	Пн 15.04.19	Пн 15.04.19	13	Программист;Верстка
Тестирование сайта 2	1 день	Чт 18.04.19	Чт 18.04.19	18	Верстальщик;Программ
Отладка сайта	4 дня	Вт 16.04.19	Пт 19.04.19		
Отладка сайта 1	1 день	Вт 16.04.19	Вт 16.04.19	15	Верстальщик;Программ
Отладка сайта 2	1 день	Пт 19.04.19	Пт 19.04.19	16	Верстальщик;Программ
Сдача сайта в эксплуатацию	1 день	Пн 22.04.19	Пн 22.04.19	19	Маркетолог;Затраты
Конец работы	0 дней	Пн 22.04.19	Пн 22.04.19	20;10	

Рис. П.А.10. Календарный план компании по разработке ПО

Название ресурса	Тип	Единицы измерения	Краткое название	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Начисление	Базовый календарь
☐ Тип: Затраты	Затраты							
Расходы на регистрацию	Затраты		P				В начале	
Рекламные расходы	Затраты		P				В начале	
Расходы на использование программного продукта	Затраты		P				В начале	
Амортизационные отчисления	Затраты		A				По окончании	
☐ Тип: Материальный	Материальный							
Затраты на электроэнергию	Материальный	квч	З		4,00 P		По окончании	
Расходные материал	Материальный	набор	P		2 500,00 P		В начале	
Обеспечение доступа в Интернет	Материальный	тариф	O		2 100,00 P		В начале	
Аренда дискового пространства на сервере	Материальный	тариф	A		1 500,00 P		В начале	
☐ Тип: Трудовой	Трудовой			600%				
Руководитель проекта	Трудовой		P	100%	1 100,00 P/ч	2 200,00 P/ч	Пропорциональнс	Стандартный
Маркетолог	Трудовой		M	100%	500,00 P/ч	1 000,00 P/ч	Пропорциональнс	Стандартный
Веб-дизайнер	Трудовой		B	100%	580,00 P/ч	1 160,00 P/ч	Пропорциональнс	Стандартный
Верстальщик	Трудовой		B	100%	600,00 P/ч	1 200,00 P/ч	Пропорциональнс	Стандартный
Программист	Трудовой		P	100%	900,00 P/ч	1 800,00 P/ч	Пропорциональнс	Стандартный
Контент-менеджер	Трудовой		K	100%	300,00 P/ч	600,00 P/ч	Пропорциональнс	Стандартный

Рис. П.А.1.1. Таблица ресурсов для компании по разработке ПО

ПРИЛОЖЕНИЕ Б

ПРИМЕРЫ ПРЕДМЕТНОЙ ОБЛАСТИ ДЛЯ ВЫПОЛНЕНИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Предметная область «Гостиница».
2. Предметная область «Клуб обучения танцам».
3. Предметная область «Магазин косметики».
4. Предметная область «Парикмахерская».
5. Предметная область «Рекламное агентство».
6. Предметная область «Санаторий».
7. Предметная область «Такси».
8. Предметная область «Туристическое агентство».
9. Предметная область «Животноводческая ферма».
10. Предметная область «Кинокомпания».
11. Предметная область «Филармония».
12. Предметная область «Ветеринарная лечебница».
13. Предметная область «Строительная компания».
14. Предметная область «Сельскохозяйственная организация».
15. Предметная область «Книжное издательство».
16. Предметная область «Фотоателье».
17. Предметная область «Кинотеатр».
18. Предметная область «Зоопарк».
19. Предметная область «Кондитерский цех».
20. Предметная область «Ателье по пошиву одежды».
21. Предметная область «Производство мебели (прием индивидуальных и типовых заказов и изготовление)».
22. Предметная область «Компания по дизайну и ремонту помещений».
23. Предметная область «Аптека».
24. Предметная область «Предприятие по лесозаготовке».
25. Предметная область «Ресторан быстрого питания».
26. Предметная область «Картинная галерея».
27. Предметная область «Агентство по организации детских праздников».
28. Предметная область «Организация по ландшафтному дизайну и благоустройству территории».
29. Предметная область «Больница».
30. Предметная область «Автовокзал».

Учебное издание

Балашова Ирина Юрьевна,
Такташкин Денис Витальевич

**Современные информационные
технологии в проектировании
программных систем и комплексов**

Под редакцией **Макарычева** Петра Петровича

Редактор *Т. Н. Судовчихина*
Технический редактор *Н. В. Иванова*
Компьютерная верстка *Н. В. Ивановой*
Дизайн обложки *А. А. Стаценко*

Подписано в печать 29.11.2019.
Формат 60×84¹/₁₆. Усл. печ. л. 6,16.
Тираж 68. Заказ № 542.

Издательство ПГУ.
440026, Пенза, Красная, 40.
Тел./факс: (8412) 56-47-33; e-mail: iic@pnzgu.ru