

**Министерство образования и науки Российской Федерации
ФГАОУ ВО «Севастопольский государственный университет»**

**Институт информационных технологий
и управления в технических системах**

Методы и средства проектирования информационных систем и технологий

Учебно-методическое пособие

для студентов всех форм обучения направления подготовки
09.03.02 «Информационные системы и технологии»



Севастополь
2015

УДК 004.07

Учебно-методическое пособие по дисциплине «Методы и средства проектирования информационных систем и технологий» для студентов всех форм обучения направления подготовки 09.03.02 «Информационные системы и технологии» / Разраб. Ю.В. Доронина, И.В. Дымченко, О.А. Сырых – Севастополь: Изд-во СевГУ, 2015. – 203 с.

Учебно-методическое пособие рассмотрено и утверждено на заседании кафедры «Информационные системы», протокол № ____ от «31» августа 2015 г.

Допущено учебно-методическим центром СГУ в качестве учебно-методического пособия.

СОДЕРЖАНИЕ

| | |
|--|----|
| Список сокращений | 6 |
| Введение | 7 |
| Раздел 1. Введение в проектирование информационных систем | 8 |
| 1.1. Основные понятия и определения | 8 |
| 1.2. Принципы создания информационных систем | 10 |
| 1.3. Подходы к созданию информационных систем | 11 |
| 1.4. Модели информационных систем | 12 |
| 1.5. Стандартизация, комплексы стандартов на информационные системы и технологии | 13 |
| 1.6. Жизненный цикл информационной системы | 15 |
| 1.6.1. Структура жизненного цикла информационной системы | 15 |
| 1.6.2. Основные модели жизненного цикла информационной системы | 17 |
| 1.6.2.1. Модель кодирования и устранения ошибок («проб и ошибок») (Build-and-fix, Code and fix) | 17 |
| 1.6.2.2. Каскадная модель («водопад» – waterfall) | 18 |
| 1.6.2.3. V-образная (шарнирная) модель | 19 |
| 1.6.2.4. Инкрементная модель (поступательная, многопроходная) | 20 |
| 1.6.2.5. Спиральная (эволюционная, итерационная) модель | 21 |
| 1.6.3. Модифицированные модели жизненного цикла информационной системы | 22 |
| 1.6.3.1. Поэтапная модель с промежуточным контролем («водооборот») | 22 |
| 1.6.3.2. Структурная эволюционная модель быстрого прототипирования (Rapid Prototype Model) | 24 |
| 1.6.3.3. Модель быстрой разработки приложений RAD (Rapid Application Development) | 25 |
| 1.6.3.4. Спиральная модель «Win-Win» | 28 |
| 1.6.3.5. Модель ROP (Rational Objectory Process) методологии RUP (Rational Unified Process) | 29 |
| 1.6.4. Адаптированные модели жизненного цикла | 33 |
| 1.6.4.1. Быстрое отслеживание | 33 |
| 1.6.4.2. Параллельный инжиниринг (Concurrent Engineering, CE) | 33 |
| 1.6.4.3. Эволюционный/инкрементный принцип | 34 |
| 1.6.4.4. Принцип V-образной инкрементной модели | 34 |
| 1.6.4.5. Модель синхронизации и стабилизации MSF (модель синхростабилизации Microsoft – Microsoft Solution Framework) | 35 |
| 1.6.5. Выбор модели жизненного цикла информационной системы | 37 |
| 1.7. Методы и принципы проектирования информационных систем | 37 |
| 1.8. Технологии проектирования информационных систем | 39 |
| 1.8.1. Технология канонического проектирования | 39 |
| 1.8.2. Технологии индустриального проектирования | 39 |
| 1.9. Методологии проектирования информационных систем | 39 |
| 1.9.1. Методологии структурного подхода (анализа) | 40 |
| 1.9.1.1. Методология моделирования функциональной структуры объектов – SADT (Structured Analysis and Design Technique) | 43 |
| 1.9.1.2. Методологии функционального моделирования потоков данных DFD (Data Flow Diagrams) и потоков работ – WFD (Work Flow Diagram) | 45 |
| 1.9.1.3. Методология функционального моделирования процессов – IDEF0 (Integrated DEfinition Function) | 48 |
| 1.9.1.4. Методология моделирования данных – ERD (Entity-Relationship Diagrams) (case-метод Баркера) | 50 |

| | |
|--|------------|
| 1.9.1.5. Методология моделирования работы в реальном времени – STD (State Transition Diagrams) | 51 |
| 1.9.1.6. Методология анализа взаимосвязей между информационными потоками – IDEF1 (IDEF1X) (Integrated DEfinition Function) | 54 |
| 1.9.1.7. Методология описания (документирования) и моделирования процессов – IDEF3 (Integrated DEfinition Function) | 57 |
| 1.9.1.8. Методология моделирования, анализа и реорганизации бизнес-процессов – BPMN (Business Process Model and Notation) | 60 |
| 1.9.2. Методологии объектно-ориентированного и модельно-ориентированного подхода (анализа) | 64 |
| 1.9.2.1. Методология прототипного создания приложений – RAD (Rapid Application Development) | 65 |
| 1.9.2.2. Методология синхронизации и стабилизации MSF (модель синхростабилизации Microsoft – Microsoft Solution Framework) | 66 |
| 1.9.2.3. Методология рационального унифицированного процесса RUP (Rational Unified Process) | 68 |
| 1.9.2.4. Методология разработки информационных систем DATARUN | 77 |
| 1.9.2.5. Методология описания бизнес-процессов ORACLE (Oracle Method) | 79 |
| 1.9.2.6. Методология проектирования интегрированных информационных систем ARIS (ARchitecture of Integrated Information Systems) | 80 |
| 1.9.2.7. Методологии модельно-ориентированного проектирования интегрированных информационных систем (MathWorks) | 81 |
| 1.9.3. Гибкие методологии проектирования (agile-методы) | 86 |
| 1.9.3.1. Методология экстремального программирования XP (eXtreme Programming) | 88 |
| 1.9.3.2. Методология управления проектами – Scrum | 90 |
| 1.9.3.3. Методология разработки динамических систем – DSDM (Dynamic Systems Development Method) | 92 |
| 1.9.3.4. Методология разработки, управляемой функциональностью FDD (Feature driven development) | 94 |
| 1.9.3.5. Методология оптимизации потока работ Kanban (Kanban Development) | 94 |
| 1.10. Средства проектирования информационных систем | 97 |
| 1.10.1. Традиционные средства проектирования | 97 |
| 1.10.2. Средства автоматизации проектирования (CASE-средства проектирования) | 97 |
| Раздел 2. Инструктивно-методические указания по подготовке к лекционным занятиям | 100 |
| Раздел 3. Инструктивно-методические указания к практическим занятиям | 103 |
| Практическое занятие №1 Модели жизненного цикла и методы планирования и управления проектами | 103 |
| Практическое занятие №2 Моделирование процесса движения информации (структурный анализ на основе DFD-диаграмм) | 112 |
| Практическое занятие №3 Функциональное моделирование процессов (методология IDEF0) | 119 |
| Практическое занятие №4 Моделирование данных (методология ERD), информационное моделирование процессов, построение реляционных информационных структур (методология IDEF1, IDEF1X) | 132 |
| Практическое занятие №5 Описания логики взаимодействия информационных потоков (методология IDEF3) | 142 |
| Практическое занятие №6 Моделирование, анализ и реорганизация бизнес-процессов (методология BPMN) | 151 |
| Практическое занятие №7 Модельно-ориентированное проектирование систем | 163 |

| | |
|--|------------|
| Раздел 4. Инструктивно-методические указания по выполнению лабораторных работ... | 166 |
| 4.1. Лабораторный практикум | 166 |
| Лабораторная работа №1 Инструментальные средства (CASE-средства) планирования и управления проектами..... | 166 |
| Лабораторная работа №2 Инструментальные средства (CASE-средства) поддержки методологий функционального моделирования потоков данных (методология DFD)..... | 170 |
| Лабораторная работа №3 Инструментальные средства (CASE-средства) поддержки методологий функционального моделирования процессов (методология IDEF0) | 174 |
| Лабораторная работа №4 Инструментальные средства (CASE-средства) поддержки методологий моделирования данных и информационного моделирования процессов (методологии ERD, IDEF1, IDEF1X) | 177 |
| Лабораторная работа №5 Инструментальные средства (CASE-средства) поддержки методологий описания логики взаимодействия информационных потоков (методология IDEF3) | 179 |
| Лабораторная работа №6 Инструментальные средства (CASE-средства) поддержки методологии BPMN | 181 |
| Лабораторная работа №7 Инструментальные средства поддержки проектирования модельно-ориентированных систем (AnyLogic) | 186 |
| 4.2. Требования к содержанию и оформлению отчетов | 192 |
| 4.3. Организация защиты и критерии оценивания выполнения практических заданий и лабораторных работ..... | 193 |
| Раздел 5. Перечень вопросов для подготовки к экзамену по дисциплине «Методы и средства проектирования информационных систем и технологий» | 196 |
| Список литературы и информационных ресурсов | 197 |
| Приложение 1. Варианты заданий к практическим занятиям и лабораторным работам..... | 201 |
| Приложение 2. Образец оформления и содержания отчета по лабораторной работе и практическому занятию..... | 202 |
| Приложение 3. Образец единого титульного листа к отчетам по лабораторным работам и практическим занятиям | 203 |

Список сокращений

| | | |
|------|---|---|
| АМ | – | Agile Modeling |
| АС | – | автоматизированная система |
| ЖЦ | – | жизненный цикл |
| ИСР | – | иерархия структуры работ |
| ИС | – | информационная система |
| МКП | – | метод критического пути |
| МОП | – | модельно-ориентированное проектирование |
| ПО | – | программное обеспечение |
| СД | – | системная динамика |
| СДР | – | структурная декомпозиция работ |
| ФУБП | – | функциональная устойчивость бизнес – процесса |

Введение

Целью данного учебно-методического пособия является помощь студентам направления подготовки 09.03.02 «Информационные системы и технологии» в организации эффективного изучения дисциплины «Методы и средства проектирования информационных систем и технологий», задача, которой, состоит в формировании системного представления о современных тенденциях в области проектирования информационных систем и технологий.

В первой части пособия систематизированы и приведены теоретические положения, касающиеся подходов, применяемых при проектировании информационных систем в настоящее время. Рассмотрены методы, методологии и технологии проектирования и дано краткое описание инструментальных средств, использующихся для автоматизации процессов разработки.

Во второй части представлены тематические блоки вопросов, соответствующие темам лекций, позволяющие студенту осуществлять, во-первых, подготовку к предстоящему лекционному занятию, а во-вторых, углубленное изучение темы, по рекомендуемым вопросам, выносимым на самостоятельную работу.

В третьей части приведены инструктивно-методические указания к практическим занятиям, целью которых является закрепление теоретических знаний о современных стандартах и технологиях проектирования, полученных в ходе теоретической подготовки, а так же формирование практических навыков по применению современных методологий проектирования информационных систем.

В четвертой части пособия представлены инструктивно-методические указания по выполнению комплекса лабораторных работ, задачей которых является исследование различных подходов, методологий и инструментальных средств, применяемых в процессе проектирования информационных систем. Здесь же приведены: требования к содержанию и оформлению отчетов по лабораторным работам, система организации их защиты и критерии оценивания выполненной работы.

Пятая часть содержит перечень вопросов для итогового контроля полученных знаний.

Пособие составлено, в соответствии с требованиями, предъявляемыми к уровню знаний и формируемым компетенциям Федеральным государственным образовательным стандартом высшего образования по направлению подготовки 09.03.02 «Информационные системы и технологии» (уровень – бакалавриат) и профессиональным стандартом 5.141118 «Специалист по информационным системам».

Раздел 1

Введение в проектирование информационных систем

1.1. Основные понятия и определения

Согласно ФЗ № 149-ФЗ [1] **информационная система (ИС)** – совокупность содержащейся в базах данных информации и обеспечивающих её обработку информационных технологий и технических средств.

В соответствии с ГОСТ 34.321-96 [2] **информационная система** – это система, которая организует хранение и манипулирование информацией о предметной области.

По ISO/IEC 2382-1:1993 [3] **информационная система** – это система обработки информации и соответствующие организационные ресурсы (человеческие, технические, финансовые и т.д.), которые обеспечивают и распространяют информацию.

Определением, учитывающим множество аспектов понятия, можно считать следующее: **информационная система** – это взаимосвязанная совокупность информационных, технических, программных, математических, организационных, правовых, эргономических, лингвистических, технологических и других средств, а также персонала, предназначенная для сбора, обработки, хранения и выдачи информации и принятия решений.

Информационная технология, согласно ГОСТ 34.003-90 [4] – это приемы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных.

Согласно ФЗ № 149-ФЗ [1] **информационная технология** – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

А в соответствии ISO/IEC 38500:2015 [5], **информационная технология** – это ресурсы, необходимые для сбора, обработки, хранения и распространения информации.

Проектирование (лат. «projectus» – брошенный вперед) – процесс создания прототипа, прообраза предполагаемого или возможного объекта, явления. Результатом проектирования является научно-теоретически и практически обоснованное определение версий или вариантов развития или изменения того или иного объекта, явления [6].

Проектирование информационной системы – представляет сложный многоступенчатый вид деятельности – процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования аналогичных объектов в проект ИС. Другими словами – это процесс перехода от первичного описания информационной системы в виде проектного (технического) задания к описанию ее в виде набора стандартных документов (проектной документации), достаточного для создания системы.

Целью проектирования ИС является подбор технического и формирование информационного, математического, программного и

организационно-правового обеспечения. Документ, полученный в результате проектирования, носит название **проект** и представляет собой совокупность проектной документации, в которой представлено **описание проектных решений** по созданию и эксплуатации ИС. В реальных условиях **проектирование** – это поиск способа (выбор технологии) создания ИС, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений [7].

Жизненный цикл (ЖЦ) информационной системы – непрерывный процесс, начинающийся с момента принятия решения о создании информационной системы и заканчивающийся в момент полного изъятия ее из эксплуатации [8].

Модель жизненного цикла – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, которые осуществляются в ходе разработки, функционирования и сопровождения информационной системы в течение всей жизни системы, от определения требований до завершения ее использования.

Модель ЖЦ включает в себя:

- стадии;
- результаты выполнения работ на каждой стадии;
- ключевые события – точки завершения работ и принятия решений.

Стадия – часть процесса создания информационной системы, ограниченная определенными временными рамками и заканчивающаяся выпуском конкретного продукта (моделей, программных компонентов, документации), определяемого заданными для данной стадии требованиями.

Стандартизация проектирования информационных систем – набор правил и соглашений, которые должны соблюдаться всеми участниками проекта, является технологической основой для разработки открытых информационных систем.

Технология проектирования информационных систем – совокупность технологических операций в их последовательности и взаимосвязи, приводящая к разработке проекта системы, при этом определяется совокупность методов (методология) и средств, направленных не только непосредственно на проектирование ИС, но и на организацию, управление, внедрение и модернизацию проекта ИС, то есть обеспечение всего жизненного цикла системы.

Методология создания информационных систем – описание процесса создания и сопровождения информационной системы в виде жизненного цикла, представление его как некоторой последовательности стадий и выполняемых на них процессов, а так же обеспечение управления этим процессом для того, чтобы гарантировать выполнение требований, как к самой системе, так и к характеристикам процесса разработки [6].

Методы проектирования информационных систем – использование определённых программных и аппаратных средств, составляющих средства проектирования.

Метод проектирования представляет собой совокупность трёх компонент:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

Средства проектирования информационных систем – комплекс инструментальных средств, обеспечивающих в рамках выбранной методологии проектирования поддержку полного жизненного цикла ИС.

1.2. Принципы создания информационных систем

Согласно ГОСТ РД 50-680-88 [9] основными принципами создания ИС определены:

- системность;
- развитие (открытость);
- совместимость;
- стандартизация (унификация);
- эффективность.

Принцип системности – при декомпозиции должны устанавливаться такие связи между структурными компонентами системы, которые обеспечивают цельность системы и ее взаимодействие с другими системами.

Принцип развития (открытости) – внесение изменений в систему, обусловленных самыми различными причинами (внедрением новых информационных технологий, изменением законодательства, организационной перестройкой внутри фирмы и т. п.), должно осуществляться только путем дополнения системы без переделки уже созданного, т. е. не нарушать ее функционирования.

Принцип совместимости – при создании системы должны быть реализованы информационные интерфейсы, благодаря которым, она может взаимодействовать с другими системами, согласно установленным правилам (особенно касается сетевых связей локального и глобального уровней).

Принцип стандартизации (унификации) – при создании системы должны быть рационально использованы типовые, унифицированные и стандартизованные элементы, проектные решения, пакеты прикладных программ, комплексы, компоненты.

Принцип эффективности – достижение рационального соотношения между затратами на создание системы и целевыми эффектами, включая конечные результаты, отражающиеся на прибыльности и получаемые по окончании внедрения автоматизации в управленческие процессы.

1.3. Подходы к созданию информационных систем

Модели, применяемые на стадии конструирования, образуют метафору проектирования или подход к проектированию.

При проектировании информационных систем используют локальный или системный подходы.

Сущность *локального подхода* к проектированию состоит в последовательном наращивании задач, решаемых в системе. В этом случае проектирование информационной системы состоит из решения задач, ориентированных на удовлетворение потребностей конкретных подразделений или требований, связанных с реализацией конкретных условий. При этом данные организовывают в отдельные логически структурированные файлы. Этот подход имеет серьезные недостатки:

- избыточность информации;
- противоречивость;
- недостаточная скорость обработки;
- отсутствие гибкости;
- низкая стандартизация программного обеспечения.

Системный подход [7], будучи общей методологической базой проектирования информационных систем, основан на концепции интеграции данных, которые описывают все сферы деятельности объекта информатизации. Этот подход предусматривает рассмотрение все элементов и составляющих процесса проектирования в их взаимосвязи, взаимозависимости и взаимном влиянии в интересах оптимального достижения как отдельных, так и общих целей создания информационной системы. Данный подход исходит из обязательной необходимости анализа элементов и составляющих процесса проектирования в их взаимосвязи на основе широкого использования современных методов исследования.

В зависимости от принципа рассмотрения, анализа и моделирования предметной области возможна конкретизация системного подхода. В этом случае на современном этапе можно выделить четыре основных подхода к проектированию:

- *структурный подход* (*функционально-ориентированное проектирование*), который использует структурные методы для построения функциональной, информационной и других моделей информационной системы;
- *блочно-иерархический подход* к проектированию использует идеи декомпозиции сложных описаний объектов и соответственно средств их создания на иерархические уровни и аспекты, вводит понятие стиля проектирования (восходящее, нисходящее, смешанное), устанавливает связь между параметрами соседних иерархических уровней;
- *объектный подход* (*объектно-ориентированное проектирование*) предлагает набор объектных моделей для описания предметной области;

– *модельный подход (модельно-ориентированное проектирование)* основан на настройке и доработке типовой конфигурации информационной системы в среде специализированных инструментальных систем.

1.4. Модели информационных систем

Для проектирования ИС используют информационные модели, представляющие объекты и процессы в форме рисунков, схем, чертежей, таблиц, формул, текстов и т.п.

Информационная модель – это модель объекта, процесса или явления, в которой представлены информационные аспекты моделируемого объекта, процесса или явления. Она является основой разработки моделей ИС.

Процесс построения информационных моделей с помощью формальных языков называют **формализацией**.

Модели, построенные с использованием математических понятий и формул, называют **математическими моделями**.

Модель должна учитывать как можно большее число факторов. Однако реализовать такое положение затруднительно особенно в слабоструктурируемых системах. Поэтому зачастую стремятся создавать модели достаточно простых элементов, с учётом их микро- и макросвязей. Это позволяет получать обозримые результаты.

Обычно различают реальное (материальное, предметное) и мысленное (идеализированное, концептуально-методологическое) моделирование.

Концептуально-методологическое моделирование представляет собой процесс установления соответствия реальному объекту некоторой абстрактной конструкции, позволяющий получить характеристики объекта. Данная модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности.

Концептуальное моделирование представляет собой структурированный процесс создания систем, состоящий из следующих этапов:

- анализ;
- проектирование;
- программирование;
- тестирование;
- внедрение.

Важнейшей формой исследования сложных систем, основанного на системном анализе, является **имитационное моделирование** на ЭВМ, описывающее процессы функционирования систем в виде алгоритмов. Его применяют в случаях, когда необходимо учесть большое разнообразие исходных данных, изучить протекание процессов в различных условиях. Процесс имитации на любом этапе может быть приостановлен для проведения научного эксперимента на вербальном (описательном) уровне, результаты которого после оценки и обработки могут быть использованы на последующих этапах имитации.

1.5. Стандартизация, комплексы стандартов на информационные системы и технологии

Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте происходит при соблюдении всеми участниками проекта правил и соглашений, касающихся, рис.1.1:

- проектирования;
- оформления проектной документации;
- пользовательского интерфейса.

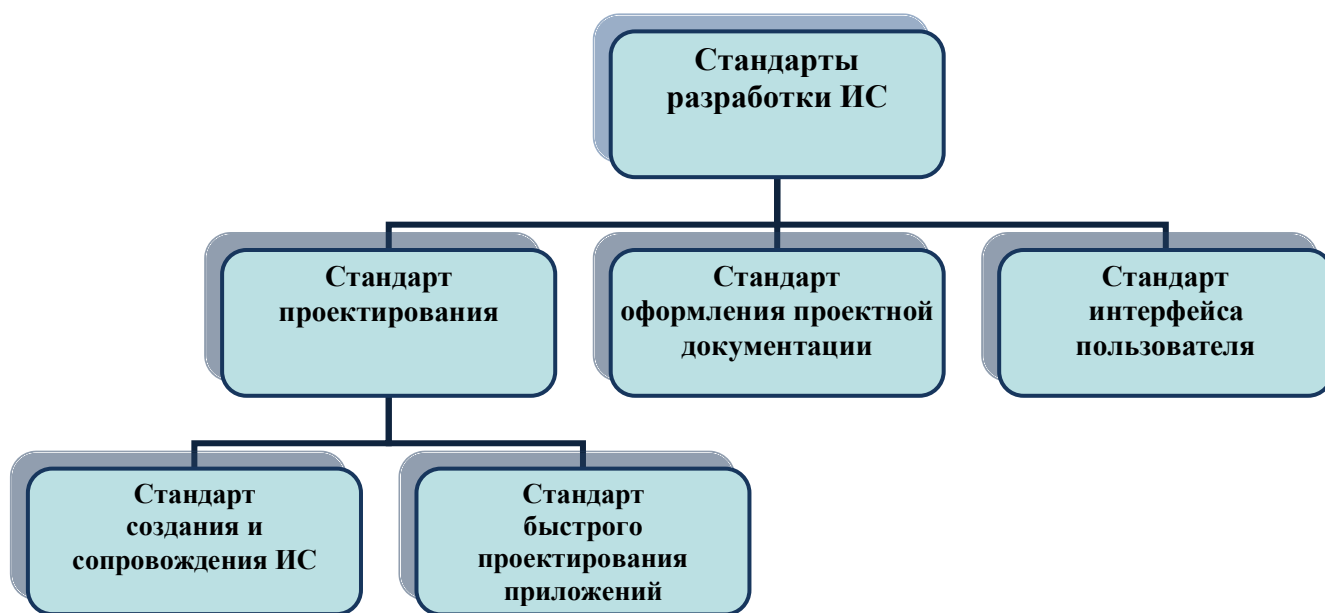


Рис.1.1. Стандарты проектирования информационных систем.

Стандарт проектирования должен устанавливать:

- набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;
- правила фиксации проектных решений на диаграммах, в том числе: правила именования объектов (включая соглашения по терминологии), набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм, включая требования к форме и размерам объектов, и т.д.;
- требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы, настройки CASE-средств, общие настройки проекта и т.д.;
- механизм обеспечения совместной работы над проектом, в том числе: правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии (регламент обмена проектной информацией, механизм фиксации общих объектов и т.д.), правила проверки проектных решений на непротиворечивость и т.д.

Стандарт оформления проектной документации должен устанавливать:

- комплектность, состав и структуру документации на каждой стадии проектирования;
- требования к ее оформлению (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.);
- правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;
- требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;
- требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарт интерфейса пользователя должен устанавливать:

- правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления;
- правила использования клавиатуры и мыши;
- правила оформления текстов помощи;
- перечень стандартных сообщений;
- правила обработки реакции пользователя.

Стандарты в области информационных технологий можно **классифицировать**:

- в зависимости от организации, утверждающей стандарты;
- по уровню утверждающей организации;
- по предмету стандартизации;
- по используемым методическим источникам.

В зависимости от *организации, утверждающей стандарты* [10]:

- официальные международные, официальные национальные или национальные ведомственные стандарты (например, стандарты ISO1, IEC2, ГОСТ);
- стандарты международных консорциумов и комитетов по стандартизации (например, стандарты OMG – Object Management Group (Группа по управлению объектами));
- стандарты «де-факто» – официально никем не утвержденные, но фактически действующие (например, стандартом «де-факто» долгое время были язык взаимодействия с реляционными базами данных SQL4);
- фирменные стандарты (например, Microsoft ODBC5, методика Oracle CDM).

По уровню утверждающей организации:

- верхний уровень – международные стандарты (ISP), признанные соответствующими международными комитетами;
- средний уровень – региональные стандарты, создаваемые для группы стран или континентов;
- нижний уровень – национальные стандарты, действующие в рамках отдельных государств.

По предмету стандартизации:

- функциональные стандарты (стандарты на языки программирования, интерфейсы, протоколы);
- стандарты на организацию жизненного цикла (ЖЦ) информационных систем.

По используемым методическим источникам:

- методические материалы научных центров;
- методические материалы фирм-разработчиков;
- методические материалы фирм-консультантов.

Объектами стандартизации при проектировании информационных систем являются:

- процесс (что делать?): ISO12207 [8], ISO15288 [11], IEEE1220 [12];
- практика (как следует делать?): ISO15504 [13];
- источник (какие данные использовать): ISO10303 [14];
- описание: IDEF [15], UML [16], IEEE 1471 [17].

Стандартизация информационных технологий и систем повышает их прибыльность за счет снижения затрат на создание и особенно модификацию.

1.6. Жизненный цикл информационной системы

Понятие жизненного цикла (ЖЦ) является одним из базовых понятий методологии проектирования информационных систем.

Модель ЖЦ, как правило, представляется в виде диаграммы ЖЦ, на которой изображаются пути перехода из некоторого начального состояния в конечное состояние и события, инициирующие изменения состояния. Модель ЖЦ может быть также представлена в виде текстового описания.

1.6.1. Структура жизненного цикла информационной системы

Структура ЖЦ по стандарту ГОСТ Р ИСО/МЭК 12207 [8] базируется на трех группах процессов:

- основные процессы ЖЦ ПО (приобретение, поставка, разработка, эксплуатация, сопровождение);
- вспомогательные процессы, обеспечивающих выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);
- организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение).

Каждый процесс включает ряд действий. Каждое действие включает ряд задач.

Структура ЖЦ по стандарту ГОСТ Р ИСО/МЭК 15288-2005 [11] базируется на пяти группах процессов:

- договорные процессы: приобретение (внутренние решения или решения внешнего поставщика); поставка (внутренние решения или решения внешнего поставщика);

- процессы предприятия: управление окружающей средой предприятия; инвестиционное управление; управление ЖЦ ИС; управление ресурсами; управление качеством;

- проектные процессы: планирование проекта; оценка проекта; контроль проекта; управление рисками; управление конфигурацией; управление информационными потоками; принятие решений;

- технические процессы, включающие: определение требований; анализ требований; разработка архитектуры; внедрение; интеграция; верификация; переход; аттестация; эксплуатация; сопровождение; утилизация;

- специальные процессы: определение и установка взаимосвязей исходя из задач и целей.

Структура ЖЦ по стандарту ГОСТ 34.601-90 [18] предусматривает следующие стадии и этапы создания автоматизированной системы (АС) (8 групп процессов):

- формирование требований к АС: обследование объекта и обоснование необходимости создания АС; формирование требований пользователя к АС; оформление отчета о выполнении работ и заявки на разработку АС;

- разработка концепции АС: изучение объекта; проведение необходимых научно-исследовательских работ; разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователей; оформление отчета о проделанной работе;

- техническое задание: разработка и утверждение технического задания на создание АС;

- эскизный проект: разработка предварительных проектных решений по системе и ее частям; разработка документации на АС и ее части;

- технический проект: разработка проектных решений по системе и ее частям; разработка документации на АС и ее части; разработка и оформление документации на поставку комплектующих изделий; разработка заданий на проектирование в смежных частях проекта;

- рабочая документация: разработка рабочей документации на АС и ее части; разработка и адаптация программ;

- ввод в действие: подготовка объекта автоматизации; подготовка персонала; комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями); строительно-монтажные работы; пусконаладочные работы; проведение предварительных испытаний; проведение опытной эксплуатации; проведение приемочных испытаний;

- сопровождение АС: выполнение работ в соответствии с гарантийными обязательствами; послегарантийное обслуживание; эскизный, технический

проекты и рабочая документация – это последовательное построение все более точных проектных решений.

Допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в «Технорабочий проект», параллельно выполнять различные этапы и работы, включать дополнительные.

***Замечание:** стандарт ГОСТ 34.601-90 не вполне подходит для проведения разработок в настоящее время: многие процессы отражены недостаточно, а некоторые положения устарели.*

1.6.2. Основные модели жизненного цикла информационной системы

Под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует.

1.6.2.1. Модель кодирования и устранения ошибок («проб и ошибок») (Build-and-fix, Code and fix)

Модель кодирования и устранения ошибок – самая простая модель жизненного цикла, алгоритм которой состоит из следующих шагов, рис.1.2:

- постановка задачи;
- создание программы;
- тестирование;
- анализ результата тестирования и возможный переход к постановке задачи.

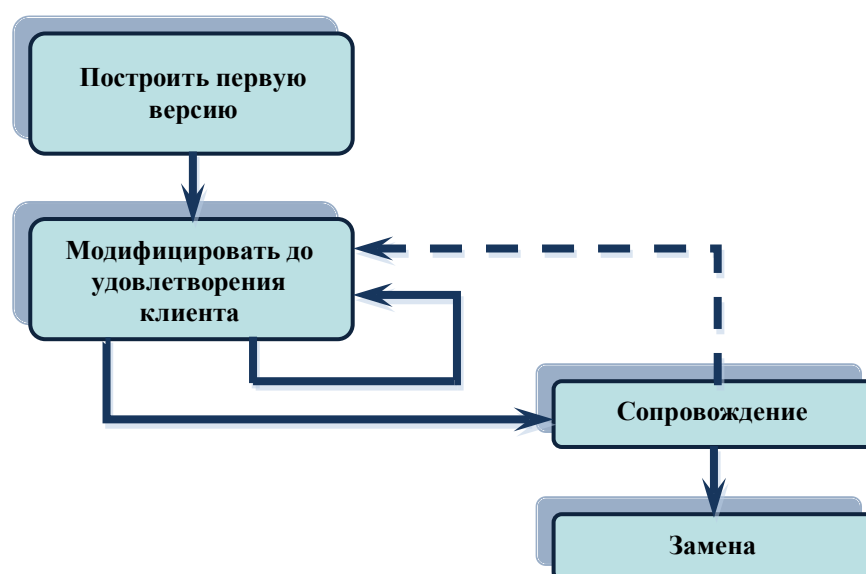


Рис.1.2. Модель кодирования и устранения ошибок.

Эта модель не актуальна при профессиональной разработке программного обеспечения. Модель Build-and-fix хороша для небольших проектов, которые не требуют сопровождения, но абсолютно непригодна для корпоративных или вообще нетривиальных проектов объемом более 1000 строк.

1.6.2.2. Каскадная модель («водопад» – waterfall)

Первой моделью, характерной для 70-85 г.г. прошлого века, получившей широкую известность и структурирующей процесс разработки, является каскадная (однократный проход, водопадная или классическая модель). Она была создана после прошедшей в 1968 г. конференции НАТО по вопросам науки и техники, где рассматривались подобные вопросы.

Модель предполагала разделение процесса создания программного продукта на последовательные этапы (применялась различными разработчиками, однако ни количество, ни содержание этапов не унифицировалось).

То есть каскадная стратегия подразумевает линейную последовательность прохождения стадий создания информационной системы, рис.1.3. Другими словами, переход с одной стадии на следующую, происходит только после того, как будет полностью завершена работа на текущей [10,19].

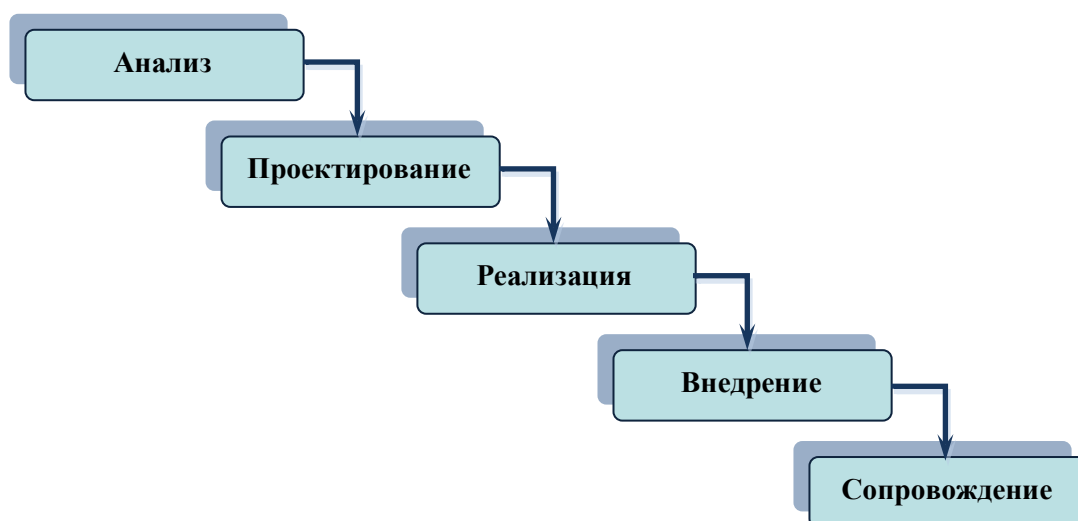


Рис.1.3. Каскадная (водопадная, классическая) модель жизненного цикла.

Данная модель применяется при разработке информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

Достоинства модели:

- на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы (денежные, материальные и людские).

Недостатки модели:

- реальный процесс разработки информационной системы редко полностью укладывается в такую жесткую схему. Особенно это относится к разработке нетиповых и новаторских систем;
- жизненный цикл основан на точной формулировке исходных требований к информационной системе. Реально в начале проекта требования заказчика определены лишь частично;
- основной недостаток – результаты разработки доступны заказчику только в конце проекта. В случае неточного изложения требований или их изменения в течение длительного периода создания ИС заказчик получает систему, не удовлетворяющую его потребностям.

V-образная модель была предложена для того, чтобы устранить недостатки каскадной модели, а название – V-образная, или шарнирная – появилось из-за ее специфического графического представления (рис.1.4).

Рис.1.4. V-диаграмма жизненного цикла.

Однако в целом V-образная модель является всего лишь модификацией каскадной и обладает многими ее недостатками. В частности, и та и другая слабо приспособлены к возможным изменениям требований заказчика. Если процесс разработки занимает продолжительное время (иногда до нескольких лет), то

полученный в результате продукт может оказаться фактически ненужным заказчику, поскольку его потребности существенно изменились [19].

1.6.2.4. Инкрементная модель (поступательная, многопроходная)

Инкрементная стратегия (англ. increment – увеличение, приращение) подразумевает разработку информационной системы с линейной последовательностью стадий, но в несколько инкрементов (версий), т.е. с запланированным улучшением продукта, рис.1.5 [19].

Другими словами, инкрементная разработка представляет собой процесс частичной реализации всей системы и постепенного наращивания функциональных возможностей

В начале работы над проектом определяются все основные требования к системе, после чего выполняется ее разработка в виде последовательности версий. При этом каждая версия является законченным и работоспособным продуктом. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т.д., пока не будет получена полная система.

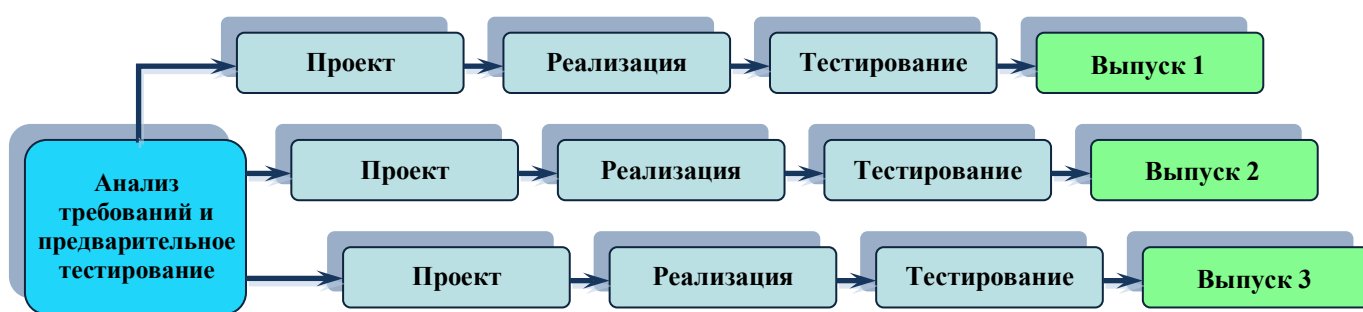


Рис.1.5. Инкрементная модель жизненного цикла.

Данная модель жизненного цикла характерна при разработке сложных и комплексных систем, для которых имеется четкое видение (как со стороны заказчика, так и со стороны разработчика) того, что собой должен представлять конечный результат (информационная система). Разработка версиями ведется в силу разного рода причин:

- отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
- отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
- требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может вызвать у ее пользователей неприятие и только «затормозить» процесс перехода на новые технологии.

Достоинства и недостатки этой стратегии такие же, как и у классической. Но в отличие от классической стратегии заказчик может раньше увидеть результаты. Уже по результатам разработки и внедрения первой версии он может незначительно изменить требования к разработке, отказаться от нее или

предложить разработку более совершенного продукта с заключением нового договора.

1.6.2.5. Спиральная (эволюционная, итерационная) модель

Спиральная стратегия (эволюционная или итерационная модель, автор Барри Боэм, 1988 г.) подразумевает разработку в виде последовательности версий, но в начале проекта определены не все требования. Требования уточняются в результате разработки версий, рис.1.6 [10,19].

Данная модель жизненного цикла характерна при разработке новаторских (нетиповых) систем. В начале работы над проектом у заказчика и разработчика нет четкого видения итогового продукта (требования не могут быть четко определены) или стопроцентной уверенности в успешной реализации проекта (риски очень велики). В связи с этим принимается решение разработки системы по частям с возможностью изменения требований или отказа от ее дальнейшего развития. Из рис.1.6 видно, что развитие проекта может быть завершено не только после стадии внедрения, но и после стадии анализа риска.

Достоинства модели:

- позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;

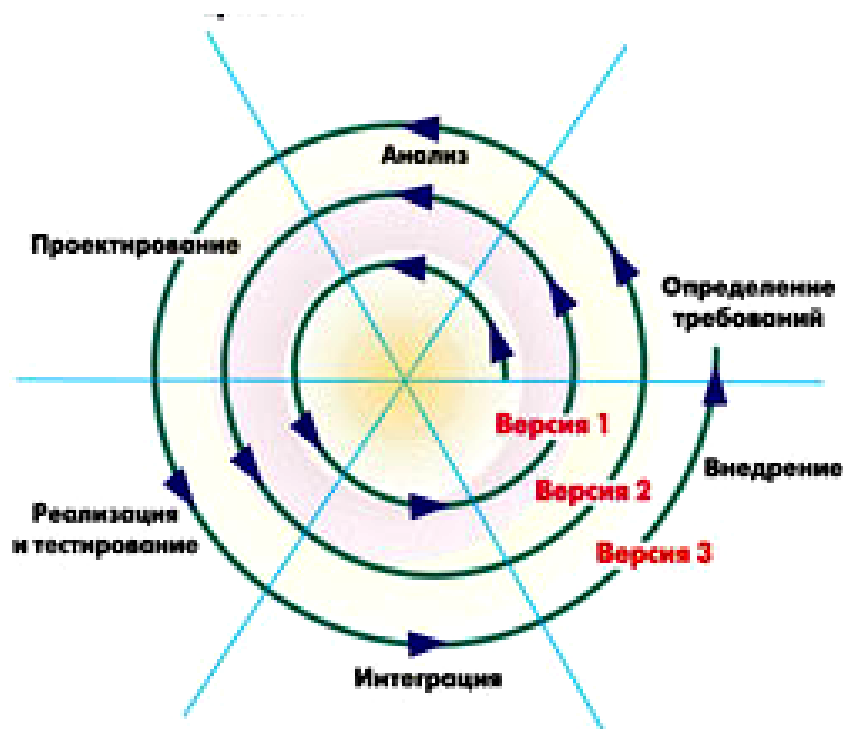


Рис.1.6. Спиральная (эволюционная, итерационная) модель жизненного цикла.

- допускает изменение требований при разработке информационной системы, что характерно для большинства разработок, в том числе и типовых;
- обеспечивает большую гибкость в управлении проектом;
- позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации;

- позволяет совершенствовать процесс разработки – анализ, проводимый в каждой итерации, позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующей итерации;
- уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта.

Недостатки модели:

- увеличивается неопределенность у разработчика в перспективах развития проекта. Этот недостаток вытекает из предыдущего достоинства модели;
- затруднены операции временного и ресурсного планирования всего проекта в целом. Для решения этой проблемы необходимо ввести временные ограничения на каждую из стадий жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа выполнена. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.

1.6.3. Модифицированные модели жизненного цикла информационной системы

1.6.3.1. Поэтапная модель с промежуточным контролем («водоворот»)

В 1970 г. каскадная модель была доработана Уинстом Ройсом с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания [19].

В таком «обратимом» виде каскадная модель просуществовала долгое время и явилась основой для многих проектов (рис.1.7).

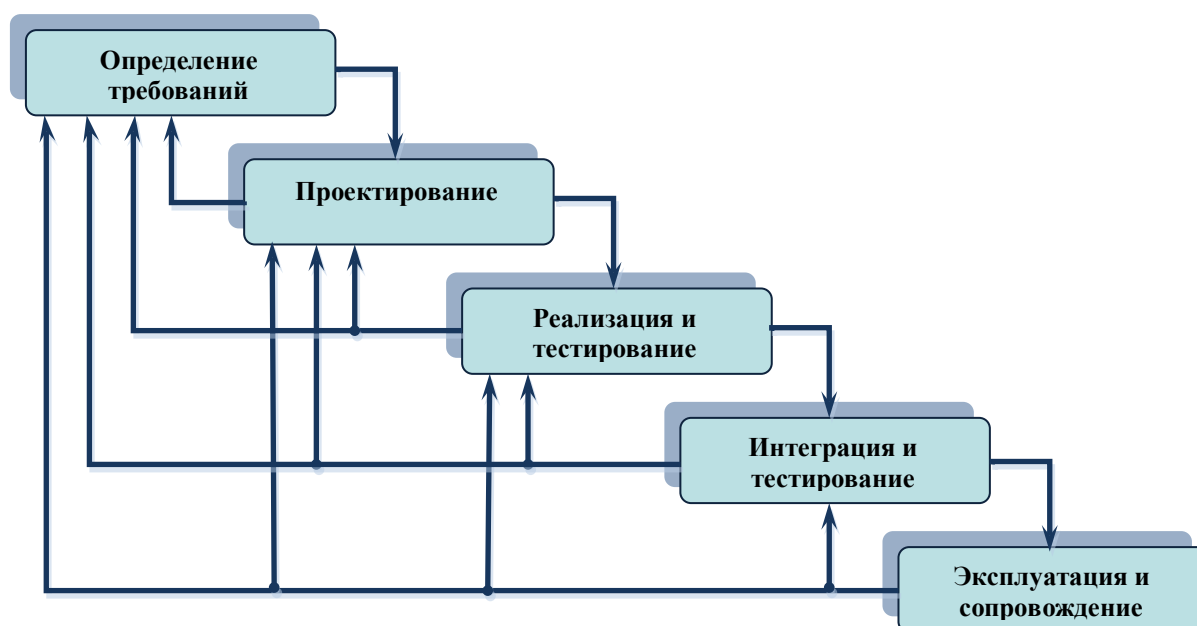


Рис.1.7. Модифицированная каскадная модель (поэтапная модель жизненного цикла с промежуточным контролем, «водоворот»).

Согласно основной идее виды деятельности распределяют таким образом, чтобы каждое действие проходило последовательно отдельным этапом. Ройс разделил процесс на следующие основные этапы (стоит отметить, что перечень этапов варьируется в изложении различных авторов):

- **определение требований.** Этот этап можно разделить на две категории - системный анализ (все то, что окружает конкретное программное обеспечение), и анализ требований. Документируют поведение системы, производительность, интерфейс и т.д.

- **планирование системного и программного обеспечения.** Фокусируется на основных свойствах программы, таких как структуры данных, архитектура программного обеспечения, функции интерфейса, алгоритмические и процедурные детали. Качество проекта возможно оценить. Результат документируется.

- **реализация и тестирование модулей.** Описанная система в проекте программируется в виде модулей и комплектов программ, которые тестируются отдельно. Чем детальнее проект, тем проще и более механическим может быть этап реализации.

- **интеграция и тестирование системы.** Объединяют программы и модули, тестируют всю систему, после тестирования продукт передается заказчику. Во время тестирования сосредотачиваются как на логических деталях, так и на том, отвечает ли система требованиям в отношении функциональности (проверка достоверности).

- **эксплуатация и сопровождение** – это обычно самая длинная фаза. Систему изменяют, если пользователи находят ошибки, либо окружение и рабочая среда изменяются или клиент нуждается в новой функциональности. Фаза повторяет все предыдущие этапы в рамках изменения существующей системы.

Результатом каждой фазы является один или несколько документов, которые утверждаются. Следующая фаза не должна начинаться до того, пока предыдущая не завершена. У фаз имеются определенные совпадения, и информация передается от одной фазы к другой.

Практическое использование данной модели выявило множество ее недостатков, главный из которых состоял в том, что она больше подходит для традиционных видов инженерной деятельности, чем для разработки ПО. В частности, одной из самых больших проблем оказалась ее «предрасположенность» к возможным несоответствиям полученного в результате продукта и требований, которые к нему предъявлялись.

Основная причина этого заключается в том, что полностью сформированный продукт появляется лишь на поздних этапах разработки, но так как работу на разных этапах обычно выполняли различные специалисты и проект передавался от одной группы к другой, то по принципу испорченного телефона оказывалось так, что на выходе получалось не совсем то, что предполагалось вначале.

1.6.3.2. Структурная эволюционная модель быстрого прототипирования (Rapid Prototype Model)

Данная модель является типичным решением для решения конкретных задач системного или стратегического, уровня [20].

Прототипирование – это процесс построения рабочей модели системы.

Прототип – это первоначальная версия системы, которая используется для апробирования возможностей дизайна и демонстрирования идей. Прототипы можно использовать на различных фазах разработки. Например, на этапе анализа требований при их нахождении и проверке; на этапе дизайна при исследовании выбора возможностей и планировании пользовательского интерфейса.

Преимущества прототипов: лучшее удобство при использовании системы, более точная совместимость с реальными потребностями пользователей; более высокое качество и более лучшее удобство сопровождения и меньше трудностей при разработке.

Жизненный цикл разработки программного продукта начинается с разработки плана проекта, затем выполняется быстрый анализ, после чего создаются база данных, пользовательский интерфейс и выполняется разработка необходимых функций. В результате этой работы получается документ, содержащий частичную спецификацию требований к программному продукту. Данный документ в дальнейшем является основой для итерационного цикла быстрого прототипирования.

В результате прототипирования разработчик демонстрирует пользователям готовый прототип, а пользователи оценивают его функционирование. После этого определяются проблемы, над устранением которых совместно работают пользователи и разработчики. Этот процесс продолжается до тех пор, пока пользователи не будут удовлетворены степенью соответствия программного продукта, поставленным перед ним требованиям. Затем прототип демонстрируют пользователям с целью получения предложений по его усовершенствованию, которые включаются в последовательные итерации до тех пор, пока рабочая модель не окажется удовлетворительной. После этого получают от пользователей официальное одобрение (утверждение) функциональных возможностей прототипа и выполняют его окончательное преобразование в готовый программный продукт, рис.1.8.

Достоинства модели:

- пользователь может получить прототип системы;
- сведение к минимуму количества неточностей в требованиях;
- нечувствительна к изменениям требований;
- формальная спецификация в рабочей модели;
- гибкое проектирование и разработка;
- минимизация разногласий с заказчиком;
- управление рисками;
- участие заказчика в проектировании и реализации.

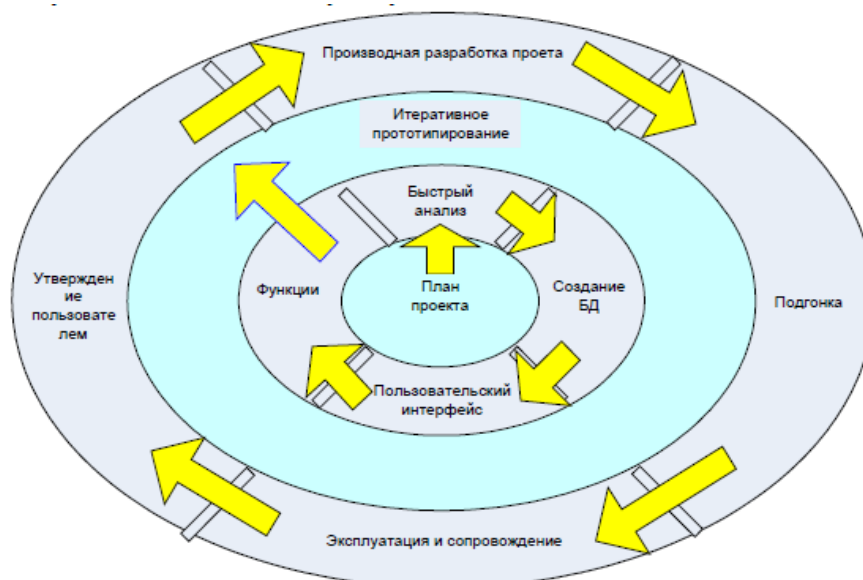


Рис.1.8. Модель быстрого прототипирования.

Недостатки модели:

- неадекватность прототипов;
- недостаток документации;
- недостаточное внимание качеству;
- заказчик может оставить себе прототип;
- заикливание прототипа;
- может быть большое количество итераций;
- процесс без надлежащего контроля может быть длительным;
- не используются методы SADT – минимум документации.

1.6.3.3. Модель быстрой разработки приложений RAD (Rapid Application Development)

Один из подходов к разработке ПО в рамках спиральной модели ЖЦ – получившая широкое распространение методология (технология) быстрой разработки приложений RAD (Rapid Application Development) [21]. Данная модель очень хорошо подходит к разработке учебных программ, т.к. включает в себя три составляющие:

- небольшую команду программистов (от 2 до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
- повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять собой группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств, способных хорошо взаимодействовать с конечными пользователями и трансформировать их

предложения в рабочие прототипы. Жизненный цикл ПО по методологии RAD состоит из четырёх фаз, рис.1.9:

- анализа и планирования требований;
- проектирования;
- построения;
- внедрения.

На **первой фазе** анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности (связи). Формулирование требований к системе осуществляется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, устанавливаются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации проекта в заданных размерах финансирования, на имеющихся аппаратных средствах и т.п.

Результатом фазы должны быть:

- список расставленных по приоритету функций будущей ПС;
- предварительная функциональная модель ПС;
- предварительная информационная модель ПС.

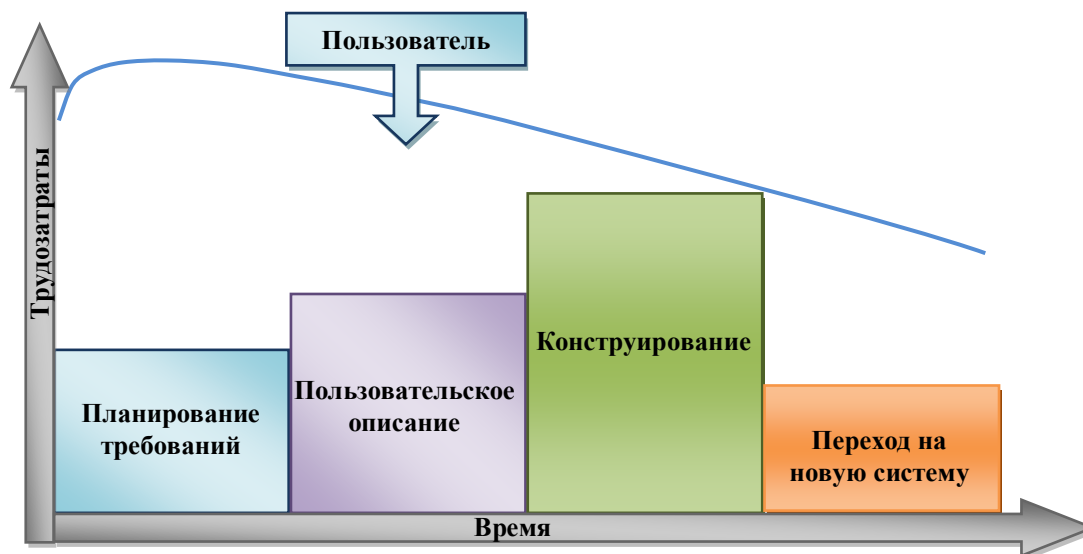


Рис.1.9. Модель быстрой разработки приложений RAD.

На **второй фазе** проектирования часть пользователей принимают участие в техническом проектировании системы под руководством специалистов-разработчиков и, взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. При необходимости корректируется функциональная модель, создаются частичные прототипы: экранов, отчетов, устраняющие неясности или неоднозначности. Устанавливаются требования разграничения доступа к данным. На этой же фазе происходит определение необходимой документации. После детального определения состава процессов

оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении системы на подсистемы.

Результатом данной фазы должны быть:

- общая информационная модель системы;
- функциональные модели системы в целом и подсистем;
- точно определенные интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, отчетов, диалогов.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На **третьей фазе** построения выполняется непосредственно сама быстрая разработка приложения (реализация подсистем). На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется в процессе разработки.

После окончания разработки подсистем производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование системы в целом. Завершается физическое проектирование системы: определяется необходимость распределения данных; осуществляется анализ использования данных; производится физическое проектирование базы данных; определяются требования к аппаратным ресурсам; определяются способы увеличения производительности; завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На **четвертой фазе** внедрения производятся обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы.

Технология RAD (как и любая другая) не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Она неприменима для разработки операционных систем; сложных расчетных программ с большим объемом программного кода и сложными уникальными алгоритмами управления; приложений, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (приложения реального

времени), так как итерационный подход предполагает, что несколько первых версий не будут полностью соответствовать требованиям.

Основные принципы технологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом этапе ЖЦ;
- обязательное вовлечение пользователей на этапе разработки;
- использование прототипирования, позволяющего выяснить и удовлетворить все требования конечного пользователя;
- тестирование и развитие проекта одновременно с разработкой;
- грамотное руководство разработкой, четкое планирование и контроль выполнения работ.

1.6.3.4. Спиральная модель «Win-Win»

Спиральная модель «Win-Win» содержит в себе больше фаз, в которых внимание сконцентрировано на участии заказчика в процессе разработки. Это достигается путем добавления к начальной фазе каждого цикла, так называемых действий Теории W (Theory W activities). Теория W – это принцип менеджмента, при реализации которого особое значение придается ключевым организаторам совместного дела, выполняющим разработку системы (пользователь, заказчик, разработчик, наладчик, создатель интерфейсов и т.д.), которые станут «победителями», если проект окажется успешным.

В этом методе, основанном на постоянном согласовании, циклы состоят из следующих фаз или стадий:

- определение участников следующего уровня;
- определение условий, необходимых для одержания участниками победы;
- согласование «победных» условий;
- формулирование целей, ограничений и альтернативных вариантов следующего уровня;
- оценка альтернативных вариантов на уровне продукта и процесса, разрешение рисков;
- определение следующего уровня продукта и процесса, включая сегментацию;
- обоснование определений продукта и процесса;
- обзор и комментарии.

Важной стадией является последующее планирование следующего цикла и обновление плана жизненного цикла, включая разделение системы на подсистемы, разработка которых осуществляется в ходе выполнения параллельных циклов. Эта стадия может включать в себя план прекращения проекта, если продолжение работы является слишком рискованным или невозможным. Также необходимо обеспечить, чтобы продолжение работы над проектом со стороны руководства осуществлялось согласно составленному плану.

Спиральная модель «Win-Win» имеет следующие **преимущества**:

- более быстрая разработка ПО благодаря содействию, оказываемому участниками проекта;

- уменьшение стоимости программ благодаря уменьшению объема переработок и текущего сопровождения;
- более высокий уровень удовлетворения со стороны участников проекта, достигаемого до разработки самого продукта;
- более высокое качество ПО благодаря использованию компромиссных качественно-атрибутивных моделей на уровне архитектуры;
- исследование большого количества вариантов построения архитектуры на ранних этапах разработки.

1.6.3.5. Модель ROP (Rational Objectory Process) методологии RUP (Rational Unified Process)

Фирма Rational Software, разработавшая язык UML, предложила также и свою модель ЖЦ, которая называется Rational Objectory Process (ROP). Означенная технология прямого перевода не имеет, т.к. rational в данном случае употребляется в значении «рациональный» и как название фирмы одновременно, во-вторых, слова objectory в английском языке не существует.

Rational Objectory Process – итеративный процесс, в течение которого происходит последовательное уточнение результатов.

Rational Objectory Process направлен именно на создание моделей, а не на разработку каких-либо других элементов проекта (например, текстовых документов).

RUP – одна из спиральных методологий разработки программного обеспечения.

Итерационная и инкрементная разработка программного обеспечения в RUP предполагает разделение проекта на несколько проектов, которые выполняются последовательно, и каждая итерация разработки четко определена набором целей, которые должны быть достигнуты в конце итерации. Конечная итерация предполагает, что набор целей итерации должен в точности совпадать с набором целей, указанных заказчиком продукта, то есть все требования должны быть выполнены.

Процесс предполагает эволюционирование моделей; итерация цикла разработки однозначно соответствует определенной версии модели программного обеспечения.

Rational Objectory Process разбит на циклы, каждый из которых, в свою очередь, состоит из четырех фаз:

- начальная стадия (Inception);
- разработка (Elaboration);
- конструирование (Construction);
- ввод в эксплуатацию (Transition).

Результатом работы каждого такого цикла является своя версия программной системы.

В этом смысле RUP является реализацией спиральной модели, хотя довольно часто изображается в виде графика-таблицы, рис.1.10.

Диаграмма имеет два измерения: горизонтальная ось представляет время и показывает временные аспекты жизненного цикла процесса; вертикальная ось представляет дисциплины, которые определяют физическую структуру процесса. На рис.1.10 видно, как с течением времени изменяются акценты в проекте. Например, в ранних итерациях больше времени отводится требованиям; в поздних итерациях больше времени отводится реализации. Горизонтальная ось сформирована из временных отрезков – итераций, каждая из которых является самостоятельным циклом разработки; цель цикла – принести в конечной продукт некоторую заранее определенную осязаемую доработку, полезную с точки зрения заинтересованных лиц.

По *горизонтальной оси* времени жизненный цикл делится на четыре основные фазы.

Начало (Inception) – формирование концепции проекта, понимание того, что мы создаем, представление о продукте (vision), разработка бизнес-плана (business case), подготовка прототипа программы или частичного решения. Это фаза сбора информации и анализа требований, определение образа проекта в целом. Цель – получить поддержку и финансирование. В конечной итерации результат этого этапа – техническое задание.

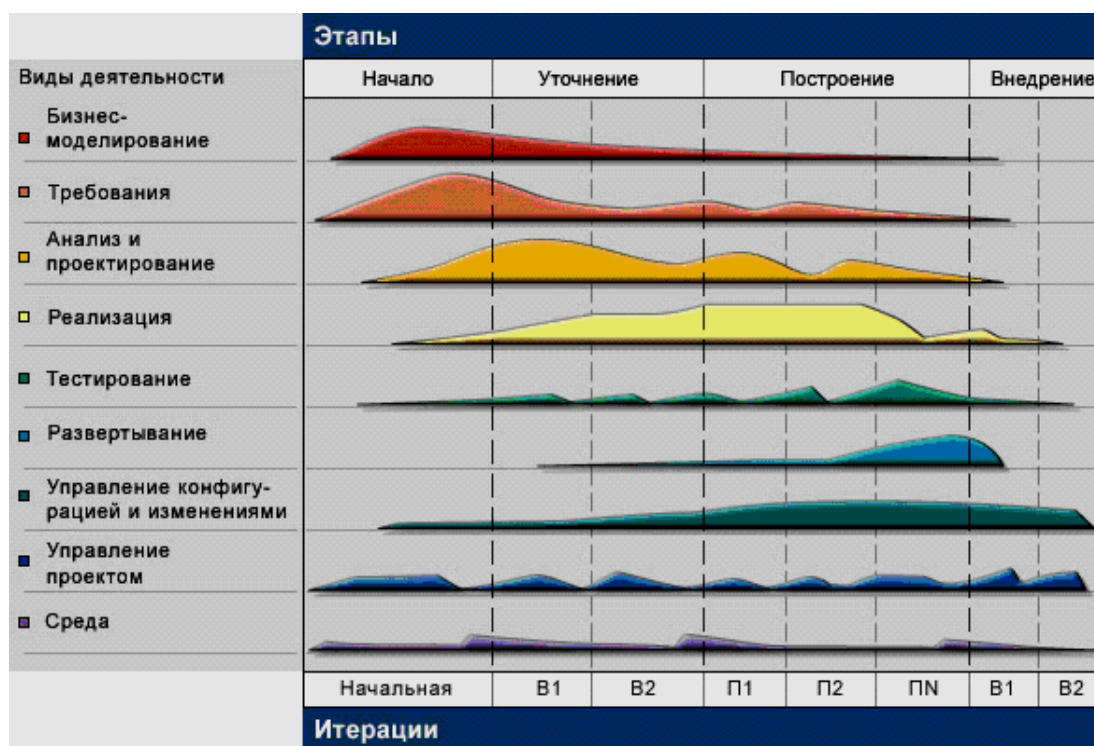


Рис.1.10. Модель жизненного цикла быстрой разработки приложений RUP.

Начальная стадия может принимать множество разных форм. Для крупных проектов – это всестороннее изучение всех возможностей реализации на протяжении нескольких месяцев. Здесь же вырабатывается бизнес-план проекта, определяется его стоимость, примерный доход, а также ограничения ресурсов – иными словами, выполняется некоторый начальный анализ оценки проекта.

Окончанием начального этапа могут служить следующие результаты:

- начальный проектный словарь терминов;
- общее описание системы – основные требования к проекту, его характеристики и ограничения;
- начальная модель вариантов использования;
- начальный бизнес-план;
- план проекта, отражающий стадии и итерации;
- один или несколько прототипов.

Проектирование, разработка (Elaboration) – уточнение плана, понимание того, как мы это создаем, проектирование, планирование необходимых действий и ресурсов, детализация особенностей. На стадии разработки выявляются более детальные требования к системе, выполняется высокоуровневый анализ предметной области и проектирование базовой архитектуры системы, создается план конструирования, и устраняются наиболее рискованные элементы проекта.

Завершается этап исполняемой архитектурой, когда все архитектурные решения приняты и риски учтены. Исполняемая архитектура представляет собой работающее программное обеспечение, которое демонстрирует реализацию основных архитектурных решений. В конечной итерации это – технический проект.

Стадия разработки занимает примерно пятую часть времени создания проекта, результатом которой являются:

- оценка времени реализации каждого варианта использования;
- идентификация всех наиболее серьезных рисков и возможности их ликвидации.

Реализация, создание системы (Construction) – этап расширения функциональности системы, заложенной в архитектуре.

Сущность стадии конструирования заключается в определении последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации, которые являются одновременно инкрементными и повторяющимися.

При этом необходимо отметить следующее:

- итерации являются инкрементными в соответствии с выполняемой функцией. Каждая итерация добавляет очередные конструкции к вариантам использования, реализованным во время предыдущих итераций;
- итерации являются повторяющимися по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода переписывается с целью сделать его более гибким.

Результатом стадии конструирования является продукт, готовый к передаче пользователям и содержащий, как правило, руководство пользователей и готовый к интеграции на требуемых платформах. В конечной итерации это – рабочий проект.

Внедрение, развертывание (Transition) – создание конечной версии продукта.

Данная стадия включает:

- бета-тестирование, позволяющее убедиться, что новая соответствует ожиданиям пользователей;
- параллельное функционирование с существующей системой, которая подлежит постепенной замене;
- оптимизацию производительности;
- обучение пользователей и специалистов службы сопровождения.

Каждая стадия завершается в четко определенной контрольной точке (milestone). В этот момент времени должны достигаться важные результаты и приниматься критически важные решения о дальнейшей разработке.

Вертикальная ось состоит из дисциплин, каждая из них может быть более детально расписана с точки зрения выполняемых задач, ответственных за них ролей, продуктов, которые подаются задачам на вход и выпускаются в ходе их выполнения и т.д.

По этой оси располагаются ключевые дисциплины жизненного цикла RUP, которые часто на русском языке называют процессами (хотя это не совсем верно с точки зрения данной методологии) поддерживаемые инструментальными средствами IBM (и/или третьих фирм):

- бизнес-анализ и моделирование (Business modeling) обеспечивает реализацию принципов моделирования с целью изучения бизнеса организации и накопления знаний о нем, оптимизации бизнес-процессов и принятия решения об их частичной или полной автоматизации;
- управление требованиями (Requirements) посвящено получению информации от заинтересованных лиц и ее преобразованию в набор требований, определяющих содержание разрабатываемой системы и подробно описывающих ожидания от того, что система должна делать;
- анализ и проектирование (Analysis and design) охватывает процедуры преобразования требований в промежуточные описания (модели), представляющие, как эти требования должны быть реализованы;
- реализация (Implementation) охватывает разработку кода, тестирование на уровне разработчиков и интеграцию компонентов, подсистем и всей системы в соответствии с установленными спецификациями;
- тестирование (Test) посвящено оценке качества создаваемого продукта;
- развертывание (Deployment) охватывает операции, имеющие место при передаче продуктов заказчикам и обеспечении доступности продукта конечным пользователям;
- конфигурационное управление и управление изменениями (Configuration management) посвящено синхронизации промежуточных и конечных продуктов и управлению их развитием в ходе проекта и поиском скрытых проблем;
- управление проектом (Management) посвящено планированию проекта, управлению рисками, контролю хода его выполнения и непрерывной оценке ключевых показателей;
- управление средой (Environment) включает элементы формирования среды разработки информационной системы и поддержки проектной деятельности.

1.6.4. Адаптированные модели жизненного цикла

1.6.4.1. Быстрое отслеживание

Методология построения жизненного цикла по принципу быстрого отслеживания заключается в ускоренном прохождении или пропуске одного или нескольких фаз жизненного цикла или процессов разработки. Многие или большинство из стандартных стадий разработки выполняются в обычном режиме, в то время как выполнение других стадий и их область действия могут быть сокращены.

Адаптация жизненного цикла необходима для реализации принципа быстрого отслеживания, который эффективнее всего используется при выполнении второстепенных проектов по разработке и приобретению ПО. Необходимость в применении быстрого отслеживания может возникнуть в случае критической нехватки времени, например, при необходимости первым поставить серийно выпускаемый продукт на рынок или в случае возникновения угрозы национального характера для государственной организации. Помимо сокращения жизненный цикл, адаптированный в целях быстрого отслеживания, обычно является менее формальным. Полный срок службы поставленного продукта может быть коротким, что указывает на короткую фазу эксплуатации.

К проектам, при выполнении которых используется принцип быстрого отслеживания, должны прибегать лишь организации с повышенными дисциплинарными требованиями. Официально установленная и определенная среда разработки сводит до минимума возможные риски. При наличии четко выраженного постоянного набора требований и метода, предусмотренного для внесения изменений, вероятность достижения успеха при выполнении проектов за принципом быстрого отслеживания увеличивается.

1.6.4.2. Параллельный инжиниринг (Concurrent Engineering, CE)

Процесс параллельного инжиниринга заключается в создании продуктов более высокого качества за меньший период времени.

Основной принцип использования этого метода заключается в том, что все аспекты жизненного цикла проекта должны учитываться в процессе от проектирования до производства как можно раньше. Благодаря раннему анализу более поздних этапов жизненного цикла выявляются проблемы, которые возникают далее в процессе разработки, а значит, это будет способствовать принятию продуманных и обоснованных решений на протяжении всего процесса разработки.

Метод параллельного инжиниринга успешно используется для проектирования ПО. При выполнении больших проектов отслеживание состояния на главных фазах жизненного цикла может обеспечить создание в высшей степени упрощенной модели.

В общих чертах можно отметить, что, как правило, параллельный инжиниринг состоит из нескольких действий (сбор требований, разработка проекта, кодирование, тестирование и т.д.), которые осуществляются одновременно. Кроме того, внутренние или внешние продукты проекта могут

находиться в одном из нескольких состояний (в состоянии разработки, анализа, проверки, ожидания следующей стадии и др.)

При использовании этого метода следует оценить возможные технические риски, чтобы определить, совместима ли разрабатываемая технология с методикой ускоренной разработки, оставить свободное место в графике разработки, периодически производить оценку технологического процесса для определения того, является ли он по-прежнему совместимым с построенным планом, и чтобы, как и при использовании более традиционных жизненных циклов, обеспечить основу для проведения оценки и тестирования, поскольку игнорирование этих действия связано с крайним риском.

1.6.4.3. Эволюционный/инкрементный принцип

По своей природе разработка программного продукта при использовании эволюционного/инкрементного принципа часто затруднена. Вопросы возникают потому, что каждая инкрементная конструкция реализует лишь небольшую часть возможностей разрабатываемой системы. Помимо обычных критериев для принятия решений по разработке, может возникнуть необходимость ответа на дополнительные вопросы:

- является ли решение о разработке текущих функциональных свойств хорошей идеей с учетом текущего объема финансирования?
- наступило ли уже время рассматривать функциональные возможности системы (приоритеты пользователя, требования процесса эволюции)?
- стоят ли добавленные функциональные возможности потраченных на них средств (или «покрывается позолотой» лишь одна область функциональных возможностей прежде, чем будут разработаны все необходимые характеристики системы)?
- хватит ли объема денежных средств на разработку требуемой системы в полном объеме?

1.6.4.4. Принцип V-образной инкрементной модели

Разработка хорошей модели жизненного цикла проекта означает заблаговременные капиталовложения, например, создание традиционной V-образной модели, смешанной с инкрементной, итеративной моделью разработки.

В этой модели предпринята попытка сбалансировать потребность в административном контроле с нуждами в технической инновации и ситуативной динамике.

Успешное использование V-образной тесно связано с тем, что происходит в контрольных точках. Эти точки представляют собой формальные механизмы, определяющие совместное принятие определенных решений по переходу к следующей фазе со стороны менеджеров и разработчиков. Вместе с периодическим проведением руководством обзоров и предварительных просмотров, контрольные точки побуждают к обсуждению вопросов, рисков и альтернатив. Значение каждой контрольной точки необходимо четко определить в рамках всего процесса. За такой высокоуровневой моделью кроются конкретные

планы, основанные на точных предварительных оценках и четко определенных контрольных точках проектирования, способствующих достижению успеха.

1.6.4.5. Модель синхронизации и стабилизации MSF (модель синхростабилизации Microsoft – Microsoft Solution Framework)

В 1994 году, стремясь достичь максимальной отдачи от ИТ-проектов, Microsoft выпустила в свет пакет руководств по эффективному проектированию, разработке, внедрению и сопровождению решений, построенных на основе своих технологий. Эти знания базируются на опыте, полученном Microsoft при работе над большими проектами по разработке и сопровождению программного обеспечения, опыте консультантов Microsoft и лучшем из того, что накопила на данный момент ИТ-индустрия. Все это представлено в виде двух взаимосвязанных и хорошо дополняющих друг друга областей знаний: Microsoft Solutions Framework (MSF) и Microsoft Operations Framework (MOF).

Модель процессов MSF (MSF process model) представляет общую методологию разработки и внедрения ИТ-решений. Особенность этой модели состоит в том, что благодаря своей гибкости и отсутствию жестко навязываемых процедур она может быть применена при разработке весьма широкого круга ИТ-проектов. Эта модель сочетает в себе свойства двух стандартных производственных моделей: каскадной (waterfall) и спиральной (spiral). Модель процессов в MSF 3.0 была дополнена еще одним инновационным аспектом: она покрывает весь жизненный цикл создания решения, начиная с его отправной точки и заканчивая непосредственно внедрением. Такой подход помогает проектным группам сфокусировать свое внимание на бизнесотдаче (business value) решения, поскольку эта отдача становится реальной лишь после завершения внедрения и начала использования продукта.

Процесс MSF ориентирован на «вехи» (milestones) – ключевые точки проекта, характеризующие достижение в его рамках какого-либо существенного (промежуточного либо конечного) результата. Этот результат может быть оценен и проанализирован, что подразумевает ответы на вопросы: «Пришла ли проектная группа к однозначному пониманию целей и рамок проекта?», «В достаточной ли степени готов план действий?», «Соответствует ли продукт утвержденной спецификации?», «Удовлетворяет ли решение нужды заказчика?» и т. д.

Модель процессов MSF учитывает постоянные изменения проектных требований. Она исходит из того, что разработка решения должна состоять из коротких циклов, создающих поступательное движение от простейших версий решения к его окончательному виду.

Особенностями модели процессов MSF являются:

- подход, основанный на фазах и вехах;
- итеративный подход;
- интегрированный подход к созданию и внедрению решений.

Модель процессов, рис.1.11 включает такие основные фазы процесса разработки, как:

- выработка концепции (Envisioning);

- планирование (Planning);
- разработка (Developing);
- стабилизация (Stabilizing);
- внедрение (Deploying).



Рис.1.11. Модель ЖЦ решения MSF.

Кроме этого, существует большое количество промежуточных вех, которые показывают достижение в ходе проекта определенного прогресса и расчлняют большие сегменты работы на меньшие, обозримые участки. Для каждой фазы модели процессов MSF определяет:

- что (какие артефакты) является результатом этой фазы;
- над чем работает каждый из ролевых кластеров на этой фазе.

В рамках MSF программный код, документация, дизайн, планы и другие рабочие материалы создаются, как правило, итеративными методами. MSF рекомендует начинать разработку решения с построения, тестирования и внедрения его базовой функциональности. Затем к решению добавляются все новые и новые возможности. Такая стратегия именуется стратегией версионирования. Несмотря на то, что для малых проектов может быть достаточным выпуск одной версии, рекомендуется не упускать возможности создания для одного решения ряда версий. С созданием новых версий эволюционирует функциональность решения.

Итеративный подход к процессу разработки требует использования гибкого способа ведения документации. «Живые» документы (living documents) должны изменяться по мере эволюции проекта вместе с изменениями требований к конечному продукту. В рамках MSF предлагается ряд шаблонов стандартных документов, которые являются артефактами каждой стадии разработки продукта и могут быть использованы для планирования и контроля процесса разработки.

Решение не представляет бизнес-ценности, пока оно не внедрено. Именно по этой причине модель процессов MSF содержит весь жизненный цикл создания решения, включая его внедрение – вплоть до момента, когда решение начинает давать отдачу.

1.6.5. Выбор модели жизненного цикла информационной системы

Выбор модели определяет основные критические параметры проекта – это успех проекта в целом, архитектура проекта, его бюджет, в каких случаях можно сэкономить. Модель должна быть адекватна опыту проектной команды с точки зрения знаний предметной области и знания конкретных технологий, CASE-средств, документирования, подходов к документированию и т.д. Серьезные модели, такие как спиральная или объектно-ориентированная, требуют определенной дисциплины и зрелости. В противном случае они вырождаются в модель проб и ошибок. Универсальной модели не существует. Выбор модели определяется исключительно характером и масштабом проекта. Ряд моделей можно комбинировать. У каждой модели есть свои преимущества и недостатки, которые обнаруживаются и имеют смысл только в контексте проекта, с учетом его особенностей [22].

В особых случаях (при создании сложных систем, систем реального времени, закрытых или секретных разработок) применяется несколько моделей одновременно, что дает возможность, как минимум, выбрать лучшую по заданным критериям.

1.7. Методы и принципы проектирования информационных систем

Метод проектирования в общем случае включает совокупность трёх составляющих:

- 1) пошаговая процедура, определяющая последовательность технологических операций проектирования;
- 2) критерии и правила, используемые для оценки результатов выполнения технологических операций;
- 3) нотации (графические и текстовые средства), используемые для описания проектируемой системы.

Методы проектирования ИС можно классифицировать по:

- по подходу к автоматизации объекта;
- степени использования средств автоматизации;
- степени использования типовых проектных решений;
- степень использования адаптивности к предполагаемым изменениям.

По подходу к автоматизации объекта методы проектирования различают:

– **метод «снизу - вверх»** – разработка подсистем (процедур, функций), в то время когда проработка общей схемы не закончилась, то есть разработка ведется от отдельных задач ко всей системе;

– **метод «сверху - вниз»** – разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация. При нисходящем проектировании задача анализируется с целью определения возможности разбиения ее на ряд подзадач. Затем каждая из полученных подзадач также анализируется для возможного разбиения на подзадачи. Процесс заканчивается, когда подзадачу невозможно или нецелесообразно далее разбивать на подзадачи;

– **принципы «дуализма» и многокомпонентности** – подход к проектированию ИС заключается в сбалансированном сочетании двух предыдущих.

По степени автоматизации методы проектирования разделяются на:

– **методы ручного проектирования**, при которых проектирование компонентов ИС осуществляется без использования специальных инструментальных программных средств, а программирование – на алгоритмических языках;

– **методы компьютерного проектирования**, при которых производится генерация или конфигурирование (настройка) проектных решений на основе использования специальных инструментальных программных средств.

По степени использования типовых проектных решений различают следующие методы проектирования:

– **типовое**, предполагающее конфигурирование ИС из готовых типовых проектных решений (программных модулей). Выполняется на основе опыта, полученного при разработке индивидуальных проектов. Типовые проекты, как обобщение опыта для некоторых групп организационно-экономических систем или видов работ, в каждом конкретном случае связаны с множеством специфических особенностей и различаются по степени охвата функций управления, выполняемым работам и разрабатываемой проектной документации.

– **оригинальное (индивидуальное)**, когда проектные решения разрабатываются «с нуля» в соответствии с требованиями к ИС. Характеризуется тем, что все виды проектных работ ориентированы на создание индивидуальных для каждого объекта проектов, которые в максимальной степени отражают все его особенности;

По степени адаптивности проектных решений выделяют методы:

– **реконструкции**, когда адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);

– **параметризации**, когда проектные решения настраиваются (генерируются) в соответствии с изменяемыми параметрами;

– **реструктуризации модели**, когда изменяется модель проблемной области, на основе которой автоматически заново генерируются проектные решения.

1.8. Технологии проектирования информационных систем

Среди технологий проектирования ИС выделяют два основных класса:

- каноническая технология;
- индустриальная технология.

1.8.1. Технология канонического проектирования

Технология канонического проектирования предполагает использование инструментальных средств универсальной компьютерной поддержки и предназначена для создания индивидуальных (оригинальных) проектов локальных ИС. При этом адаптация проектных решений возможна лишь путем перепрограммирования соответствующих программных модулей.

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в ГОСТ 34.601-90 [18].

1.8.2. Технологии индустриального проектирования

Технологии индустриального проектирования используют инструментальные средства специальной компьютерной поддержки для разработки проектов сложных интегрированных (корпоративных) ИС.

Индустриальная технология проектирования, в свою очередь, разбивается на два подкласса:

- *типовое* (параметрически-ориентированное или модельно-ориентированное) проектирование.
- *автоматизированное проектирование* (использование CASE-технологий).

Замечание: использование индустриальных технологий не исключает использования в отдельных случаях канонических.

1.9. Методологии проектирования информационных систем

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов [6].

В настоящее время существует ряд общих методологий разработки ИС. Главное в них – единая дисциплина работы на всех этапах жизненного цикла системы, учет критических задач и контроль их решения, применение развитых инструментальных средств поддержки процессов анализа, проектирования и реализации ИС.

Основными задачами, решение которых должна обеспечивать методология создания информационных систем, являются следующие:

- обеспечение создания информационных систем, отвечающих целям и задачам предприятия и соответствующих предъявляемым к ним требованиям;
- гарантия создания системы с заданными параметрами в течение заданного времени в рамках оговоренного заранее бюджета;
- простота сопровождения, модификации и расширения системы с целью обеспечения ее соответствия изменяющимся условиям работы предприятия;
- обеспечение создания информационных систем, отвечающих требованиям открытости, переносимости и масштабируемости;
- возможность использования в создаваемой системе разработанных ранее средств информационных технологий (программного обеспечения, баз данных, средств вычислительной техники, телекоммуникаций).

***Замечание:** именно методология определяет, какие средства будут применяться для разработки программного обеспечения и, во многом, рекомендует, какой технологический подход будет при этом использован.*

1.9.1. Методологии структурного подхода (анализа)

Все наиболее распространенные методологии структурного подхода базируются на ряде **общих принципов**. В качестве двух базовых принципов используются следующие:

- принцип «разделяй и властвуй» – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- принцип иерархического упорядочивания – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). Основными из этих принципов являются следующие:

- принцип абстрагирования – заключается в выделении существенных аспектов системы и отвлечения от несущественных;
- принцип формализации – заключается в необходимости строгого методического подхода к решению проблемы;
- принцип непротиворечивости – заключается в обоснованности и согласованности элементов;
- принцип структурирования данных – заключается в том, что данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

- SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;
- DFD (Data Flow Diagrams) диаграммы потоков данных;
- ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь".

Одними из самых известных и широко используемых методологий в области моделирования бизнес-процессов являются методологии семейства IDEF. Семейство IDEF появилось в конце 60-х гг. XX в. под названием SADT (Structured Analysis and Design Technique) [23]. В настоящее время оно включает следующие стандарты, рис.1.12 [24]:

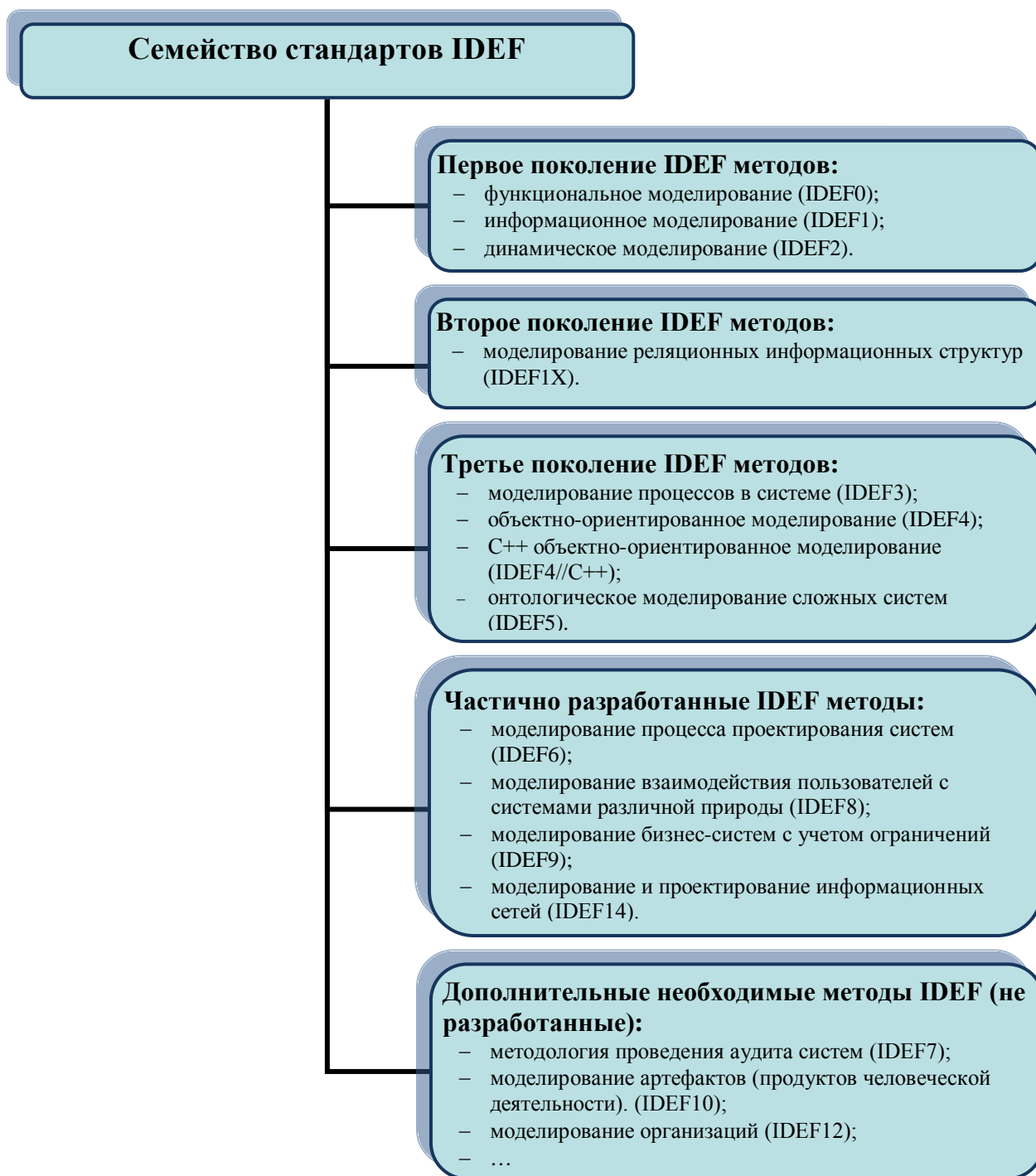


Рис.1.12. Семейство стандартов IDEF [24].

– *IDEF0* – методология функционального моделирования. Используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции;

– *IDEF1* – методология моделирования информационных потоков внутри систем, позволяющая отображать их структуру и взаимосвязи. Методология применяется для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы;

– *IDEF1X (IDEF1X Extended)* – методология построения реляционных информационных структур. IDEF1X относится к типу методологий «сущность-связь» и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;

– *IDEF2* – методология динамического моделирования развития систем, которая позволяет построить динамическую модель меняющихся во времени поведения функций, информации и ресурсов системы. В настоящее время известны алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных сетей Петри» (CPN – Color Petri Nets);

– *IDEF3* – методология документирования процессов, происходящих в системе. С помощью IDEF3 описываются сценарий и последовательность операций для каждого процесса. Функция в диаграмме IDEF3 может быть представлена в виде отдельного процесса средствами *IDEF3*;

– *IDEF4* – методология построения объектно-ориентированных систем. Средства IDEF4 позволяют наглядно отображать структуру объектов и заложенные принципы их взаимодействия, позволяя тем самым анализировать и оптимизировать сложные объектно-ориентированные системы;

– *IDEF5* – методология онтологического исследования сложных систем. С помощью этой методологии онтология системы описывается при помощи определенного словаря терминов и правил, на основе которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы и производится ее оптимизация;

– *IDEF6 (Design Rational Capture)* – метод рационального представления процесса проектирования информационных систем, позволяющий обосновать необходимость проектируемых моделей, выявить причинно-следственные связи и отразить это в итоговой документации системы;

– *IDEF8 (User Interface Modeling)* – Human – System Interaction Design Method – метод проектирования взаимодействия пользователей с системами различной природы (не обязательно информационно-вычислительными).

– *IDEF9 (Business Constraint Discovery Method)* – метод изучения и анализа бизнес-систем в терминах «ограничений». Ограничения иницируют результат, руководят и ограничивают поведение объектов и агентов (автономных программных модулей) для выполнения целей или намерений системы.

– *IDEF14 (Network Design Method)* – метод проектирования вычислительных сетей, позволяющий устанавливать требования, определять сетевые компоненты, анализировать существующие сетевые конфигурации и формулировать желаемые характеристики сети.

Анонсированные корпорацией KBSI (Knowledge Based System Inc.) методы IDEF7 (Information System Audit Method), IDEF10 (Information Artifact Modeling) и IDEF12 (Organization Design) не получили дальнейшего развития.

С помощью методологий семейства IDEF можно эффективно отображать и анализировать модели деятельности широкого спектра сложных систем. К настоящему времени наибольшее распространение и применение имеют методологии IDEF0 и IDEF1 (IDEF1X), получившие в США статус федеральных стандартов.

1.9.1.1. Методология моделирования функциональной структуры объектов – SADT (Structured Analysis and Design Technique)

Методология SADT (Structured Analysis and Design Technique – Технология структурного анализа и проектирования) [23], разработанная Дугласом Т. Россом в 1969-73 годах. С 1973 г. сфера ее использования существенно расширяется для решения задач, связанных с большими системами, такими, как проектирование телефонных коммуникаций реального времени, автоматизация производства (САМ), создание программного обеспечения для командных и управляющих систем, поддержка боеготовности. Она с успехом применялась для описания большого количества сложных искусственных систем из широкого спектра областей (банковское дело, очистка нефти, планирование промышленного производства, системы наведения ракет, организация материально-технического снабжения, методология планирования, технология программирования). Причина такого успеха заключается в том, что SADT является полной методологией для создания описания систем, основанной на концепциях системного моделирования и представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта в какой-либо предметной области.

В основе методологии SADT лежат два основных принципа:

– **SA-блоки** на основе которых создается иерархическая многоуровневая модульная система, каждый уровень которой представляет собой законченную систему (блок), поддерживаемую и контролируемую системой (блоком), находящейся над ней.

– **декомпозиция**. Использование этой концепции позволяет разделить каждый блок, понимаемый как единое целое, на свои составляющие, описываемые на более детальной диаграмме. Процесс декомпозиции проводится до достижения нужного уровня подробности описания. Диаграмма ограничивается 3-6 блоками для того, чтобы детализация осуществлялась постепенно. Вместо одной громоздкой модели используется несколько небольших взаимосвязанных моделей, значения которых взаимно дополняют друг друга, делая понятной структуризацию сложного объекта.

Модель SADT отображает функциональную структуру объекта, т.е. производимые им действия, и связи между этими действиями.

Результатом применения методологии SADT является модель, состоящая из диаграмм, текстов и глоссария, имеющих ссылки друг на друга. Ее главные компоненты – диаграммы, состоящие из блоков, рис.1.13. Блоки на диаграмме изображают системные функции, а дуги (стрелки) изображают множество различных объектов системы.

Блоки обычно располагаются в соответствии с порядком их доминирования, т.е. их важностью относительно друг друга. Дуги, связывающие блоки, изображают наборы объектов и могут разветвляться и соединяться различными сложными способами. Однако, разветвляясь и соединяясь, дуги должны во всех случаях сохранять представляемые ими объекты.

SADT-диаграммы, являются декомпозициями ограниченных объектов. Объект ограничивается блоком и касающимися его дугами.

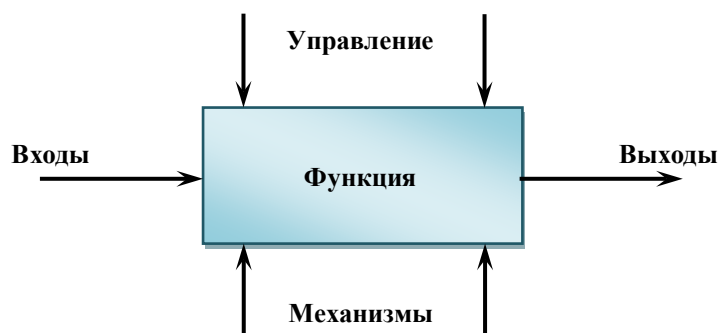


Рис.1.13. Функциональный блок и интерфейсные дуги SADT-диаграммы.

По мере создания диаграмм, отображающих модель SADT, производится постепенное введение в рассмотрение все большего числа уровней детализации, так что в результате модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, представленные в виде блоков (см.п.п.1.8.1.5 – стандарт IDTF0, являющийся конечным этапом методологии описания функциональных систем SADT).

SADT-модель дает полное, точное и адекватное описание системы, имеющее конкретное назначение. Это назначение, называемое целью модели, вытекает из формального определения модели в SADT:

M есть модель системы **S** ,
если **M** может быть использована для получения ответов на
вопросы относительно **S** с точностью **A** .

Наличие собственного графического языка SADT, и его усиленное использование преобразовало SADT в законченную методологию, способную повысить качество продуктов, создаваемых на ранних стадиях развития проекта.

В программе интегрированной компьютеризации производства (ICAM) Министерства обороны США была признана полезность SADT, что привело в 1993 году к *стандартизации и публикации ее части, называемой IDEF0* (п.1.8.1.5) в качестве федерального стандарта в США [25], а в 2000 году – в качестве руководящего документа по стандартизации в Российской Федерации [26]. Под названием IDEF0 SADT применялась тысячами специалистов в военных и промышленных организациях.

1.9.1.2. Методологии функционального моделирования потоков данных DFD (Data Flow Diagrams) и потоков работ – WFD (Work Flow Diagram)

Классическая технология описания процессов, которая была разработана на заре рождения процессных технологий управления, достаточно проста и состоит всего лишь из двух стандартов описания бизнес-процессов – DFD и WFD. Большинство других современных стандартов, несмотря на другие названия, представляют небольшие разновидности и дополнения двух классических подходов DFD и WFD.

Методология DFD (Data Flow Diagrams – диаграммы потоков данных) представляет модель системы как иерархию диаграмм потоков данных, описывающих процессы верхнего уровня (процессы, получающиеся на начальных этапах процессной декомпозиции). К данной группе процессов, как правило, относят процессы преобразования информации от момента ее ввода в систему до выдачи конечному пользователю и описывает: операции обработки информации; документы и информацию; объекты, организационные единицы сотрудников и т.д., которые участвуют в обработке информации; внешние объекты, которые участвуют в процессе, но находятся за его границами; хранилища документов, данных и информации.

Методология DFD [27] чаще всего может быть представлен следующими нотациями:

- нотация Йордона – де Марко (Yourdon and Coad Process Notation), рис.1.14;

- нотация Гейна-Сарсона (Gane and Sarson Process Notation), рис.1.15.

Диаграммы потоков данных используются для описания движения документов и обработки информации как дополнение к IDEF0. В отличие от IDEF0, где система рассматривается как взаимосвязанные работы и стрелки представляют собой жесткие взаимосвязи, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой.

Главная цель – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между процессами.

DFD отражает функциональные зависимости значений, вычисляемых в системе, включая входные значения, выходные значения и внутренние хранилища данных.

DFD – это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.



Рис.1.14. Пример DFD –диаграммы в нотация Йордона – де Марко.

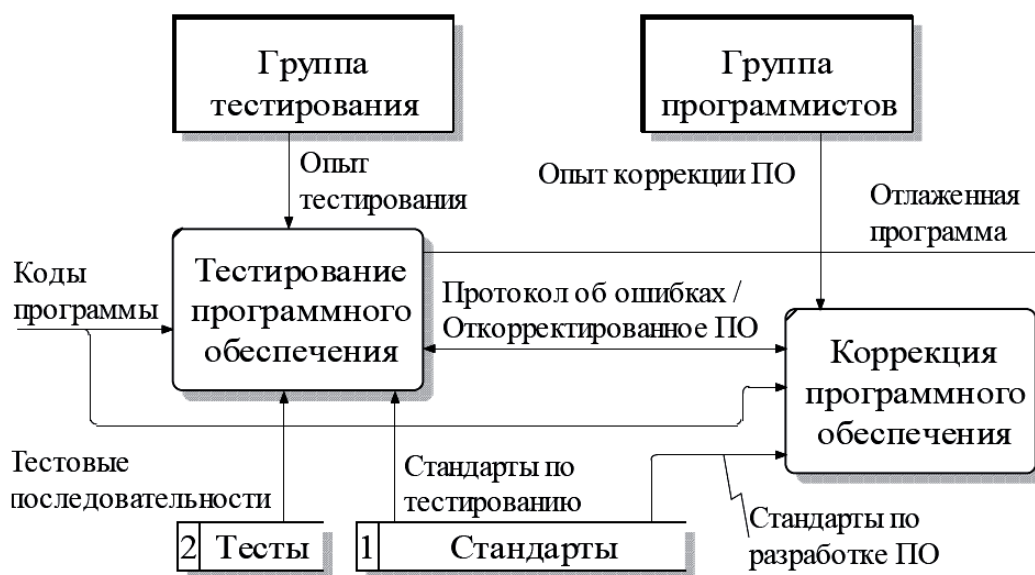


Рис.1.15. Пример DFD-диаграммы в нотации Гейна-Сарсона.

DFD содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

Построение DFD-модели базируется на принципе декомпозиции. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня.

DFD-модель включает в себя три документа, которые ссылаются друг на друга:

- графические диаграммы, рис.1.14,1.15;

- миниспецификация;
- словарь данных.

В соответствии с концепцией потоков данных (DFD) архитектура организационной системы изображается такими типами графических объектов, как: процесс, хранилище, внешняя сущность, поток данных.

Механизмы, используемые для моделирования передачи энергии, материи или информации из одной части системы в другую называют – *потоками данных*. Они обычно изображаются именованными стрелками. *Хранилище данных* служит для определения данных, которые сохраняются между процессами. *Процессы* (функции) предназначены для продуцирования выходных потоков из входных в соответствии с действием, задаваемым именем процесса. *Внешняя сущность* представляет собой сущность вне контекста системы, которая является отправителем или получателем системных данных.

Методология WFD (Work Flow Diagram – диаграмма потоков работ) [28] и представляет модель системы как диаграмму потоков работ, которая используется для описания процессов нижнего уровня (процессы, получающиеся на последующих этапах процессной декомпозиции и являющиеся составной частью процессов верхнего уровня). У диаграммы потоков работ имеются и другое название – диаграмма алгоритмов.

На диаграмме потоков работ появляются дополнительные объекты, с помощью которых описывается процесс: логические операторы, события начала и окончания процесса, а также элементы, показывающие временные задержки, рис.1.16.

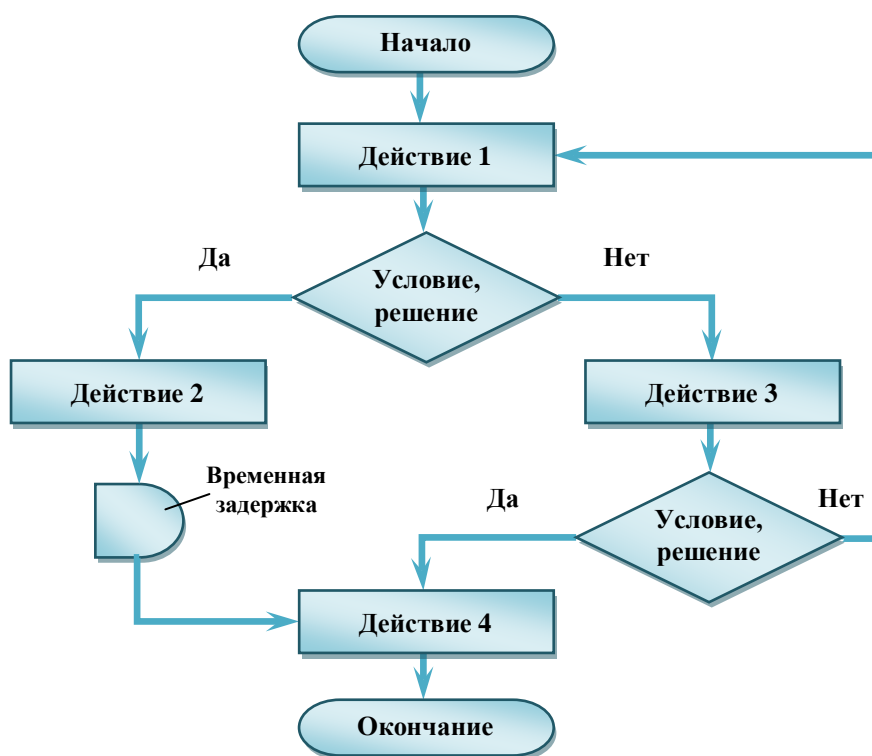


Рис.1.16. Диаграмма потоков работ – WFD.

С помощью логических операторов, которые еще называют блоками принятия решений, показывают альтернативы, которые происходят в процессе, показывается в каких случаях процесс протекает по одной технологии, а в каких случаях по другой.

С помощью событий начала и окончания процесса показывается, когда процесса начинается и когда заканчивается.

Для жестко формализованных процессов, в качестве событий может выступать время. В случаях, когда описание бизнес-процесса проводится с целью его дальнейшей временной оптимизации, используют элементы задержки времени, которые показывают места, в которых между последовательно выполняемыми работами имеется временной разрыв. В данном случае последующая работа начинается только через некоторое время после завершения предшествующей.

В классическом подходе WFD на данной схеме не показывают документы, так эти схемы используются для описания процессов нижнего уровня, содержащие детальные работы, по названию которых понятно, что является входом и что является выходом.

Отличительной особенностью WFD-диаграммы является то, что стрелки между операциями бизнес-процесса обозначают не потоки объектов (информационные и материальные), а потоки или временную последовательность выполнения работ.

***Замечание:** DFD (построение диаграмм потоков данных) и WFD (построение диаграммы потоков работ) являются ключевыми методологиями описания бизнес-процессов.*

1.9.1.3. Методология функционального моделирования процессов – IDEF0 (Integrated DEfinition Function)

Основу методологии IDEF0 (Integrated Definition Function), считающейся конечным этапом методологии описания функциональных систем SADT (п.1.8.1.1), составляет графический язык описания (моделирования) систем – графическая нотация, предназначенная для формализации и описания бизнес-процессов.

Стандарт IDEF0 был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий ICAM (Integrated Computer Aided Manufacturing), предложенной департаментом Военно-Воздушных Сил США. Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF – ICAM DEFinition).

С 1981 года стандарт IDEF0 претерпел несколько незначительных изменений, в основном ограничивающего характера. Последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST). Стандарт IDEF0 с 1993 г. принят в качестве Федерального стандарта для функционального моделирования и обработки информации США, используется в Министерстве обороны Великобритании,

НАТО и множеством других различных корпораций, осуществляющих в своей практике функциональное моделирование [25].

В 2000 году Госстандарт России принял Руководящий документ «Методология функционального моделирования IDEF0» для целей реинжиниринга деловых процессов и процессов менеджмента качества [26]. В настоящее время методология IDEF0 рассматривается ИСО на предмет международного стандарта IPS (стандарты по обработке информации).

IDEF0-методология – это методология функционального моделирования, согласно которой система представляется как совокупность взаимодействующих процессов/работ/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. Поэтому исследование или разработка любой сложной системы начинается с функционального анализа и моделирования как системы в целом, так и всех ее подсистем.

Данная методология при описании функционального аспекта информационной системы конкурирует с методами, ориентированными на потоки данных (DFD). В отличие от них IDEF0 позволяет:

- описывать любые системы, а не только информационные (DFD предназначена для описания программного обеспечения);
- создать описание системы и ее внешнего окружения до определения окончательных требований к ней. Иными словами, с помощью данной методологии можно постепенно выстраивать и анализировать систему даже тогда, когда трудно еще представить ее воплощение.

Таким образом, IDEF0 может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению. Основу методологии IDEF0 составляет графический язык описания процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

В терминах IDEF0 система представляется в виде комбинации блоков и дуг, рис.1.17.

Стандарт IDEF0 получил большое распространение в США и активно используется в России. Но в виду того, что в стандарте IDEF0 появилась дополнительная аналитика по сравнению с классическим DFD-стандартом, схемы процессов, получаемые при описании в стандарте IDEF0, выглядят более сложными. IDEF0 является излишне информационно насыщенным и сложным стандартом.

Второй недостаток стандарта IDEF0 заключается в том, что он дает больше поводов и возможностей сторонникам сопротивления изменениям притормозить проект по описанию и оптимизации процессов за счет возникновения вопросов типа: «А правильно ли, что этот объект отнесен ко входу? Может его лучше отнести к управлению?», что так же вызвано усложненной аналитикой стандарта IDEF0.

Тем не менее, стандарт IDEF0 имеет большое распространение в России, так как по нему существует множество книг и информационно-методических материалов. Также существуют программные продукты, поддерживающие данный стандарт.

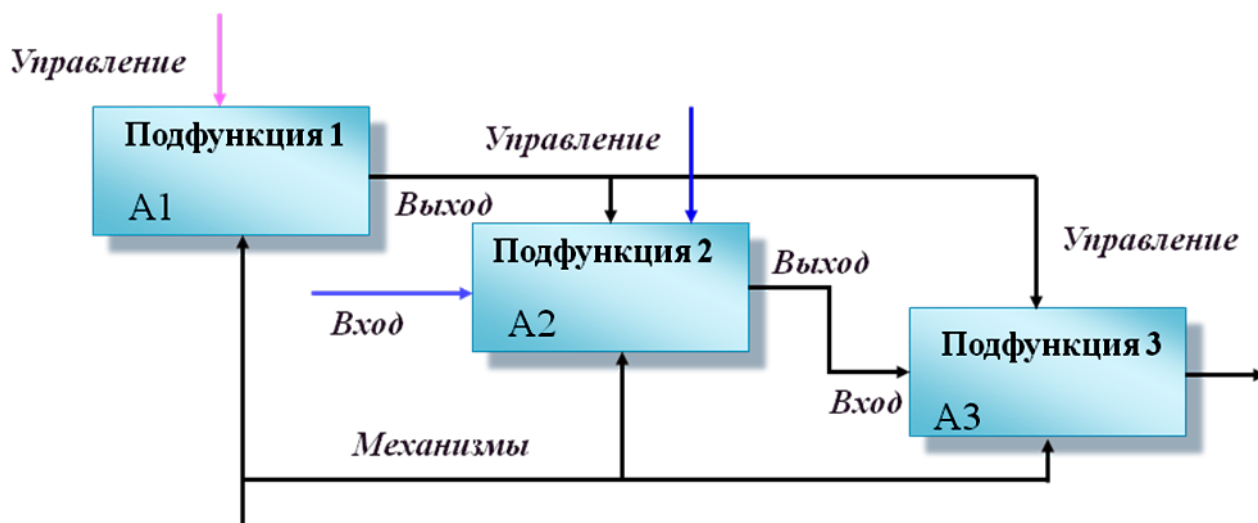


Рис.1.17. Декомпозиция бизнес-процесса на составляющие его операции в стандарте IDEF0.

Практика показала, что стандарт IDEF0 наиболее часто применяется как технология исследования и проектирования систем на логическом уровне и его целесообразно использовать в проектах по описанию и оптимизации локальных процессов, а так же в небольших проектах в которых больше участвуют и принимают решения специалисты предметных областей.

1.9.1.4. Методология моделирования данных – ERD (Entity-Relationship Diagrams) (case-метод Баркера)

Наиболее распространенной методологией моделирования данных являются диаграммы «сущность-связь» – ERD (Entity-Relationship Diagrams). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П.Ченом (Chen) и получила дальнейшее развитие в работах Баркера [29]. Она включает в себя:

- сущности, которые являются существительными, и представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и т.п.), обладающих общими атрибутами или характеристиками;
- отношения, в самом общем виде представляют собой связи между двумя и более сущностями. Именованное отношение осуществляется с помощью грамматического оборота глагола;
- связи, каждая из которых соединяет сущность и отношение и может быть направлена только от отношения к сущности. Значение связи характеризует ее тип (1:1 – один-к-одному; 1:*n* – один-к-многим; *n*:*m* – многие-к-многим). Связь

соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности;

– атрибуты, которые являются прилагательными или модификаторами. Каждая сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности. При этом любой атрибут может быть определен как ключевой. Детализация сущности осуществляется с использованием диаграмм атрибутов, которые раскрывают ассоциированные сущностью атрибуты.

ERD-диаграмма, рис.1.18, позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации.

ERD-диаграммы можно подразделить на отдельные части, соответствующие отдельным задачам, решаемым проектируемой системой. Это позволяет рассматривать систему с точки зрения функциональных возможностей, делая процесс проектирования управляемым.

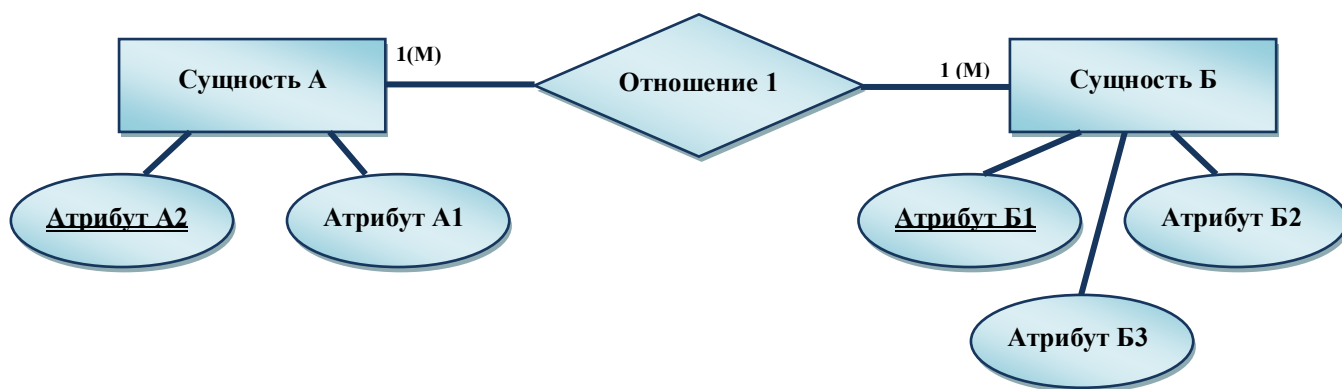


Рис.1.18. Пример ER-диаграммы в нотации П.Чена.

Существуют и другие нотации для представления ER-диаграмм: нотация Мартина, нотация IDEF1X (см.п.1.8.1.6), нотация Баркера.

1.9.1.5. Методология моделирования работы в реальном времени – STD (State Transition Diagrams)

Методология STD (State Transition Diagram) предназначена для моделирования аспектов функционирования системы, зависящих от времени или реакции на события (так называемая работа в реальном времени) [30].

Диаграммы переходов состояний STD обычно используются для описания отношения между входными и выходными управляющими потоками на управляющем процессе-предке и позволяют осуществлять декомпозицию управляющих процессов.

Такие диаграммы позволяют осуществить декомпозицию управляющих процессов, происходящих в системе, и описать отношение между управляющими потоками. С помощью диаграмм перехода состояний можно моделировать последующее функционирование системы исходя из предыдущих и текущего состояний.

Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены. Для перехода в состояние нужно какое-либо условие – условие перехода. Оно может быть информационным (условие появления информации) или временным. STD состоит из следующих объектов, рис.1.19.



Рис.1.19. Пример STD узлов.

Состояние – рассматривается как устойчивое значение некоторого свойства в течение определенного времени. Начальное состояние – узел STD, являющийся стартовой точкой для начального системного перехода. STD имеет ровно одно начальное состояние, соответствующее состоянию системы после ее инсталляции, но перед началом реальной обработки, а также любое (конечное) число завершающих состояний;

Переход определяет перемещение моделируемой системы из одного состояния в другое. При этом имя перехода идентифицирует событие, являющееся причиной перехода и управляющее им. Это событие обычно состоит из управляющего потока (сигнала), возникающего как во внешнем мире, так и внутри моделируемой системы при выполнении некоторого условия. Управляющий поток – это «трубопровод», через который проходит управляющая информация.

Имеются следующие типы управляющих потоков:

- Т-поток (trigger flow) – поток управления процессом, который может вызвать выполнение процесса. При этом процесс включается одной короткой операцией;
- А-поток (activator flow) – поток управления процессом, который может изменять выполнение отдельного процесса. Используется для обеспечения непрерывности выполнения процесса до тех пор, пока поток «включен», с «выключением» потока выполнение процесса завершается;

– E/D-поток (enable/disable flow) –поток управления процессом, который может переключать выполнение отдельного процесса. Течение по E-линии вызывает выполнение процесса, которое продолжается до тех пор, пока не возбуждается течение по D-линии. Можно использовать 3 типа таких потоков: E-поток, D-поток, E/D-поток.

Условие представляет собой событие (или события), вызывающее переход и идентифицируемое именем перехода. Если в условии участвует входной управляющий поток управляющего процесса-предка, то имя потока должно быть заключено в кавычки, например. Кроме условия с переходом может связываться действие или ряд действий, выполняющихся, когда переход имеет место.

Действие – это операция, которая может иметь место при выполнении перехода. Если действие необходимо для выбора выходного управляющего потока, то имя этого потока должно заключаться в кавычки. Для спецификации A-, T-, E/D-потоков имя запускаемого или переключаемого процесса также должно заключаться в кавычки.

Фактически *условие* есть некоторое внешнее или внутреннее событие, которое система способна обнаружить и на которое она должна отреагировать определенным образом, изменяя свое *состояние*. При изменении состояния система обычно выполняет одно или более *действий* (производит вывод, выдает сообщение на терминал, выполняет вычисления). Таким образом, действие представляет собой отклик, посылаемый во внешнее окружение, или вычисление, результаты которого запоминаются в системе (обычно в хранилищах данных на DFD), для того, чтобы обеспечить реакцию на некоторые из планируемых в будущем событий.

На STD *состояния* представляются узлами, а *переходы* – дугами. *Условия* (по-другому называемые стимулирующими событиями) идентифицируются именем перехода и возбуждают выполнение перехода. *Действия* или отклики на события привязываются к переходам и записываются под соответствующим условием.

Начальное состояние на диаграмме должно иметь входной переход, изображаемый потоком из подразумеваемого стартового узла (иногда этот стартовый узел изображается небольшим квадратом и привязывается к входному состоянию). При построении STD рекомендуется следовать нижеперечисленным правилам:

- строить STD на как можно более высоком уровне детализации DFD;
- строить как можно более простые STD;
- по возможности детализировать STD;
- использовать те же принципы именований состояний, событий и действий, что и при именовании процессов и потоков.

Применяются **два способа построения STD**.

Первый способ заключается в идентификации всех возможных состояний и дальнейшем исследовании всех не бессмысленных связей (переходов) между ними.

По *второму способу* сначала строится начальное состояние, затем следующие за ним и т.д.

Результат (оба способа) – предварительная STD, для которой затем осуществляется контроль состоятельности, заключающийся в ответе на следующие вопросы:

- все ли состояния определены и имеют уникальное имя?
- все ли состояния достижимы?
- все ли состояния имеют выход?
- (для каждого состояния) реагирует ли система соответствующим образом на все возможные условия (особенно на ненормальные)?
- все ли входные (выходные) потоки управляющего процесса отражены в условиях (действиях) на STD?

В ситуации, когда число состояний и/или переходов велико, для проектирования спецификаций управления могут использоваться *таблицы* и *матрицы переходов состояний*. Обе эти нотации позволяют зафиксировать ту же самую информацию, что и диаграммы переходов состояний, но в другом формате.

Первая колонка таблицы содержит список всех состояний проектируемой системы, во второй колонке для каждого состояния приведены все условия, вызывающие переходы в другие состояния, а в третьей колонке - совершаемые при этих переходах действия. Четвертая колонка содержит соответствующие имена состояний, в которые осуществляется переход из рассматриваемого состояния при выполнении определенного условия.

Матрица переходов состояний содержит по вертикали перечень состояний системы, а по горизонтали список условий. Каждый ее элемент содержит список действий, а также имя состояния, в которое осуществляется переход.

Используется и другой вариант данной нотации: по вертикали указываются состояния, из которых осуществляется переход, а по горизонтали - состояния, в которые осуществляется переход. При этом каждый элемент матрицы содержит соответствующие условия и действия, обеспечивающие переход из «вертикального» состояния в «горизонтальное».

1.9.1.6. Методология анализа взаимосвязей между информационными потоками – IDEF1 (IDEF1X) (Integrated DEfinition Function)

Методология IDEF1 [31] разработанная Т.Рэмеем, основана на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

Методология IDEF1 позволяет на основе наглядных графических представлений моделировать информационные взаимосвязи и различия между:

- реальными объектами;
- физическими и абстрактными зависимостями, существующими среди реальных объектов;
- информацией о реальных объектах;
- структурой данных, используемой для приобретения, накопления и управления информацией.

Центральным понятием IDEF1 является понятие «сущность», рис.1.22.

Каждая сущность имеет своё имя и атрибуты.

Под связями в IDEF1 понимаются ссылки, соединения и ассоциации между сущностями.

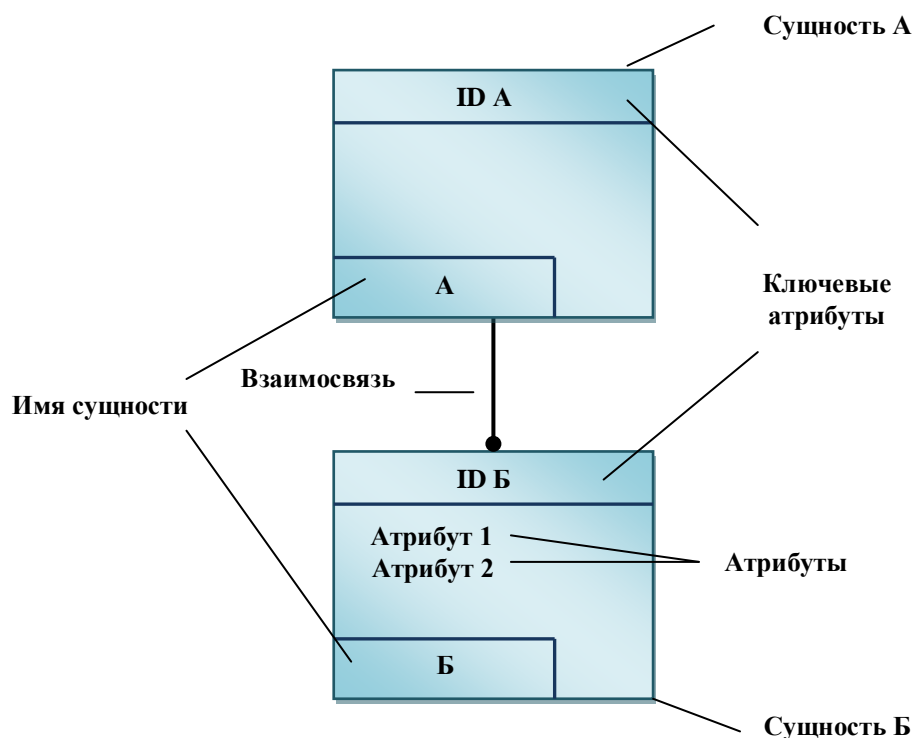


Рис.1.20. Стандарт IDEF1.

Разработка стандарта IDEF1 предполагала его использование в виде некоего инструмента для детального анализа, а также изучения взаимосвязей среди информационных потоков коммерческой деятельности фирмы. Целью исследования – частичное дополнение и формирование структуры уже существующей информации и создание отличного менеджмента. Такая необходимость возникает обычно на первоначальном этапе построения корпоративной системы информации.

Методология IDEF1 наглядно может представить образовавшиеся слабые и пустые места в современной инфраструктуре потоков.

Данная модель включает не только рассмотрение компонентов, которые уже автоматизированы, базу данных и соответствующую информацию, но и реальных объектов деятельности (телефоны, кабинеты, штат сотрудников и т.д.). Основная задача методологии IDEF1 – выявление и четкое разграничение потребностей в данном информационном менеджменте внутри коммерческой деятельности фирмы. IDEF1 – это аналитический метод.

В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия – методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации.

IDEF1X – методология, предназначена для построения концептуальной схемы логической структуры реляционной базы данных, которая была бы независимой от программной платформы её конечной реализации. По сравнению с IDEF1 она дополнительно оперирует рядом понятий, правил и ограничений, такими как домены, представления, первичные, внешние и суррогатные ключи и другими, пришедшими из реляционной алгебры [32,33].

Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того как все информационные ресурсы исследованы и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято. Средства моделирования IDEF1X специально разработаны для построения реляционных информационных систем, и если существует необходимость проектирования другой системы, например, объектно-ориентированной, то лучше избрать другие методы моделирования.

Концепция и семантика IDEF1X. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друг от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира.

Классификация сущностей в IDEF1X. При разработке модели, зачастую, приходится сталкиваться с сущностями, уникальность которых зависит от значений атрибута внешнего ключа. Для этих сущностей (для уникального определения каждой сущности) внешний ключ должен быть частью первичного ключа дочернего объекта.

Дочерняя сущность, уникальность которой зависит от атрибута внешнего ключа, называется зависимой сущностью.

Зависимые сущности далее классифицируются на сущности, которые не могут существовать без родительской сущности и сущности, которые не могут быть идентифицированы без использования ключа родителя (сущности, зависящие от идентификации).

Сущности, независимые при идентификации от других объектов в модели, называются независимыми сущностями.

Связи между сущностями в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи это суть глаголы, которые показывают, как соотносятся сущности между собой.

Набор атрибутов, выбранных для идентификации уникальных экземпляров сущности – это первичный ключ.

Выбор первичного ключа для сущности является очень важным шагом, и требует большого внимания. В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов.

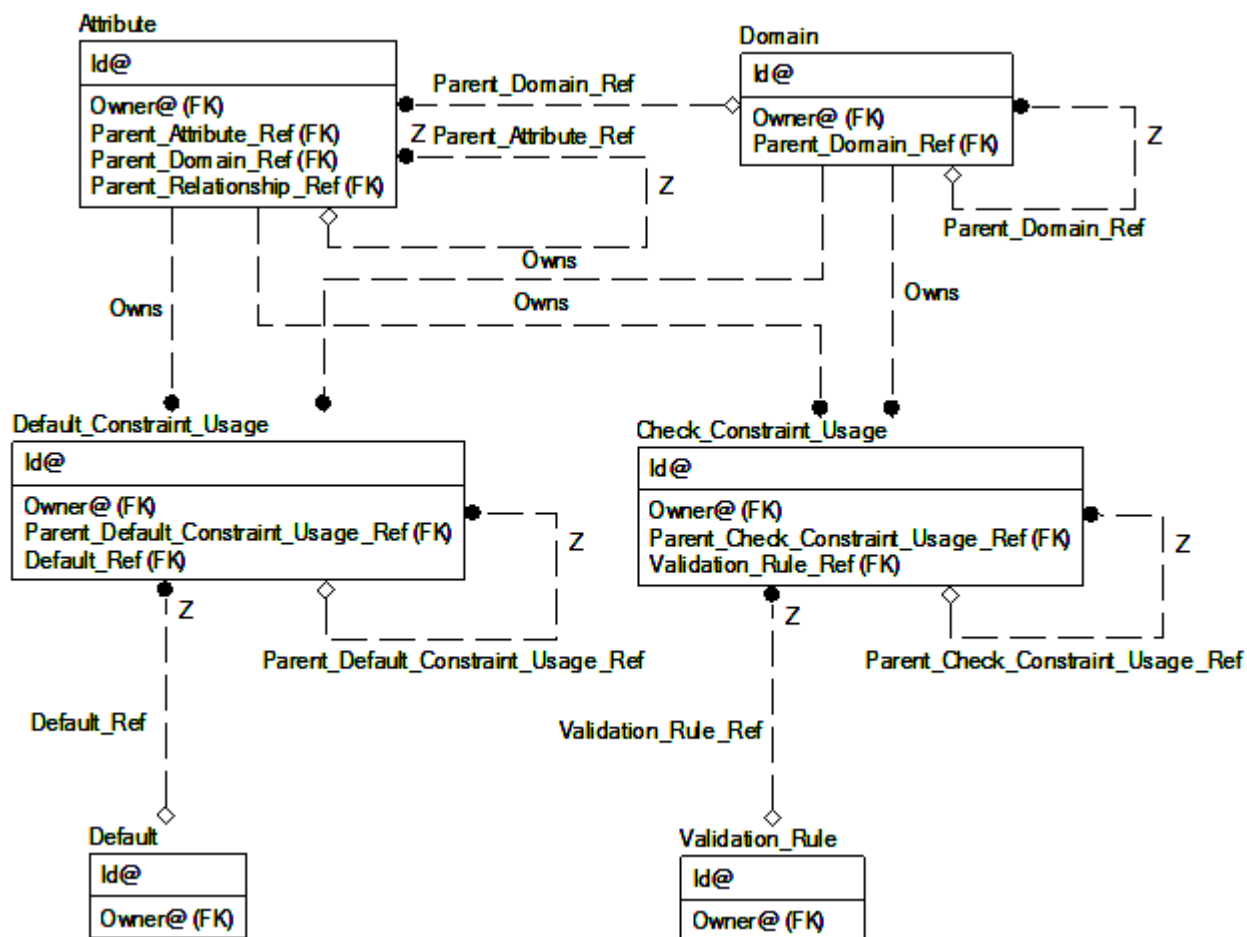


Рис.1.21. Пример диаграммы IDEF1X.

В IDEF1X концепция зависимых и независимых сущностей усиливается типом взаимосвязей между двумя сущностями. Если требуется, чтобы внешний ключ передавался в дочернюю сущность (и, в результате, создавал зависимую сущность), то необходимо создать идентифицирующую связь между родительской и дочерней сущностью.

Неидентифицирующие связи, являющиеся уникальными для IDEF1X, также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения другого типа передачи атрибутов внешних ключей – передача в область данных дочерней сущности (под линией).

Основным *преимуществом IDEF1X*, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая, несомненно, является значительным недостатком ERD.

1.9.1.7. Методология описания (документирования) и моделирования процессов – IDEF3 (Integrated DEfinition Function)

Нотация IDEF3, вторая важнейшая нотация (после IDEF0), предназначена для описания потоков работ (Work Flow Modeling) [34,35]. IDEF3 широко используется для создания моделей бизнес-процессов организации на нижнем

уровне – при описании работ, выполняемых в подразделениях и на рабочих местах.

Описание процесса по нотации IDEF3 представляет собой структурированную базу знаний, которая состоит из набора диаграмм описания процесса, объектных диаграмм изменения состояний объектов и уточняющих форм. Цель такого описания может состоять как в документальном оформлении и распространении знаний о процессе, так и в идентификации противоречивости или несовместимости выполнения отдельных операций.

Для эффективного управления любым процессом, необходимо иметь детальное представление о его сценарии и структуре сопутствующего документооборота. Средства документирования и моделирования IDEF3 позволяют выполнять следующие задачи:

- документировать имеющиеся данные о технологии выполнения процесса, выявленные, в процессе опроса специалистов предметной области, ответственных за организацию рассматриваемого процесса или участвующих в нем;
- анализировать существующие процессы и разрабатывать новые;
- определять и анализировать точки влияния потоков сопутствующего документооборота на сценарий технологических процессов;
- определять ситуации, в которых требуется принятие решения, влияющего на ЖЦ процесса, например изменение конструктивных, технологических или эксплуатационных свойств конечного продукта;
- содействовать принятию оптимальных решений при реорганизации процессов;
- разрабатывать имитационные модели технологических процессов, по принципу «как будет, если...».

Моделирование в стандарте IDEF3 производится с использованием графического представления процесса, материальных и информационных потоков в этом процессе, взаимоотношений между операциями и объектами в процессе.

Нотация визуального моделирования IDEF3 позволяет создать графическое описание логики взаимодействия информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов, очередность их запуска и завершения. IDEF3 предоставляет инструмент моделирования сценариев действий сотрудников организации, отделов, и т.п.

Нотация IDEF3 была взята за основу при создании методики описания процессов ARI-eEPC – «расширенной цепочки процесса, управляемого событиями».

IDEF3-диаграмма (диаграмма потока, process flow diagram, IDEF3-diagram, workflow) – основная единица описания в IDEF3, являющаяся графическим представлением назначения системы или процесса и применяются для анализа завершенности процесса обработки информации (проверка модели ИС на целостность).

Обычно IDEF3-диаграммы являются дополнением к IDEF0-диаграммам, т.к. содержат все необходимые сведения для построения моделей, которые в

дальнейшем могут быть использованы для имитационного анализа. С помощью диаграмм IDEF3 можно анализировать сценарии из реальной жизни, например, как закрывать магазин в экстренных случаях или какие действия должны выполнить менеджер и продавец при закрытии. Каждый такой сценарий содержит в себе описание процесса и может быть использован, что бы наглядно показать или лучше задокументировать бизнес-функции организации.

Существуют **два типа диаграмм** в стандарте IDEF3, представляющие описание одного и того же сценария процесса в разных ракурсах:

- *диаграммы Описания Последовательности Этапов Процесса* – *PFDD* (Process Flow Description Diagrams).

- *диаграммы Состояния Объекта и его Трансформаций в Процессе* – *OSTN* (Object State Transition Network).

С помощью диаграмм PFDD документируется последовательность и описание стадий в рамках процесса.

Диаграмма PFDD, рис.1.22 является графическим отображение сценария процесса.

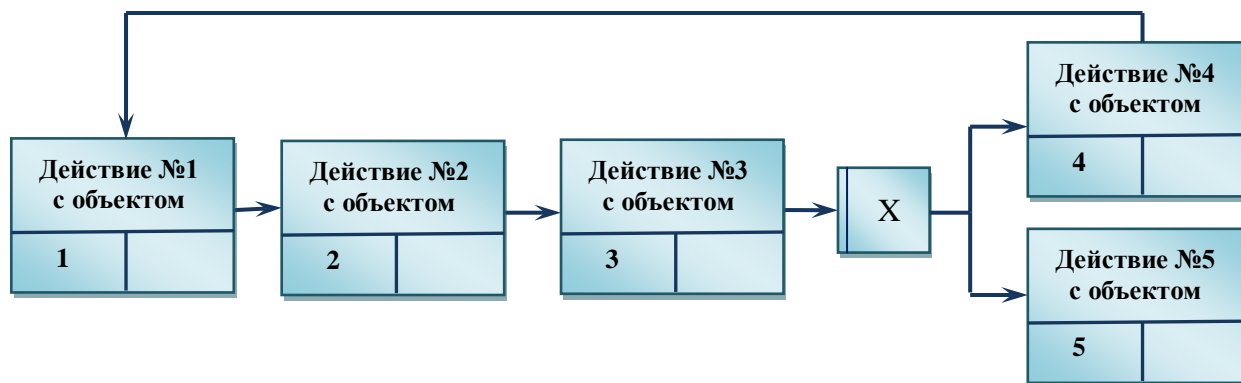


Рис.1.22. Пример PFDD диаграммы.

Прямоугольники на диаграмме PFDD называются *функциональными элементами* или *элементами поведения* – UOB (Unit of Behavior,) и обозначают событие, стадию процесса или принятие решения. Каждый UOB имеет свое имя, отображаемое в глагольном наклонении и уникальный номер.

Стрелки или линии являются отображением перемещения объекта между UOB-блоками в ходе процесса.

Объект, обозначающийся префиксом **J** – называется *перекрестком* (*Junction*). Перекрестки используются для отображения логики взаимодействия стрелок (потоков) при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы.

Все перекрестки в PFDD диаграмме нумеруются, каждый номер имеет префикс «J».

Каждый функциональный блок UOB может иметь последовательность декомпозиций, и, следовательно, может быть детализирован с любой необходимой точностью. Под декомпозицией понимается представление каждого UOB с помощью отдельной IDEF3 диаграммы. При этом эта диаграмма будет

называться дочерней, по отношению к детализируемой, а исходная, соответственно родительской.

Если диаграммы PFDD технологический процесс «С точки зрения наблюдателя», то другой класс диаграмм IDEF3 OSTN позволяет рассматривать тот же самый процесс «С точки зрения объекта». На рис.1.23 представлено пример изображения OSTN диаграммы.

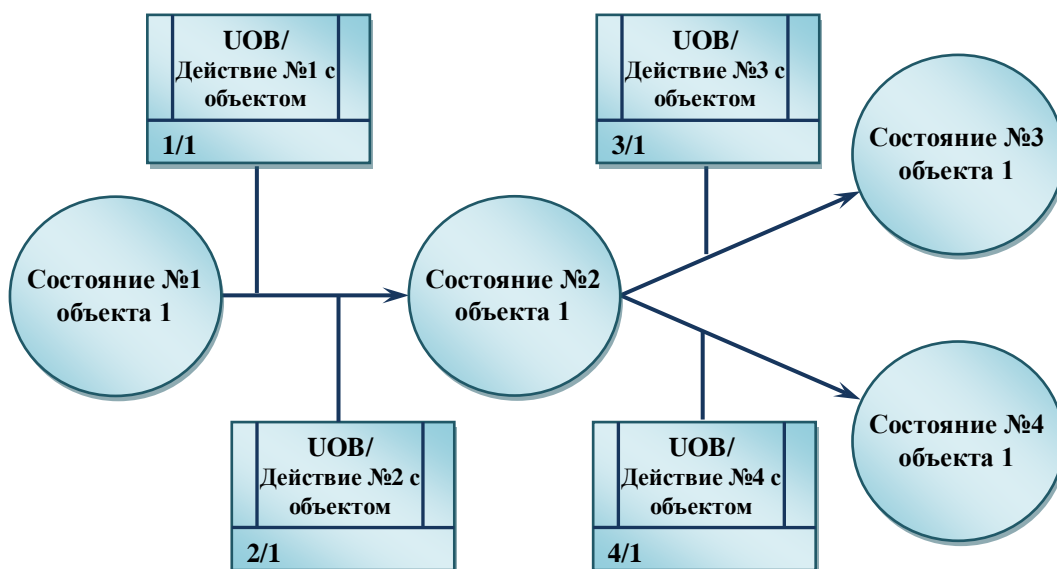


Рис.1.23. Пример OSTN диаграммы.

Состояния объекта и Изменение состояния являются ключевыми понятиями OSTN диаграммы. Состояния объекта отображаются окружностями, а их изменения направленными линиями. Каждая линия имеет ссылку на соответствующий функциональный блок UOB, в результате которого произошло отображаемое ею изменение состояния объекта.

Другими словами диаграммы OSTN используются для иллюстрации трансформаций объекта, которые происходят на каждой стадии процесса.

Нотацию IDEF3 целесообразно применять в случае относительно простых процессов на нижнем уровне декомпозиции, т.е. процессов уровня рабочих мест. В этом случае схема процесса может служить основой для создания документов, регламентирующих работу исполнителей.

Очевидно, что процесс в нотации IDEF3 является «плоским». При помощи этой нотации достаточно сложно создавать комбинированные модели, в которых бы сочетались описания потоков работ и процессы управления этими работами.

1.9.1.8. Методология моделирования, анализа и реорганизации бизнес-процессов – BPMN (Business Process Model and Notation)

Модель и нотация бизнес-процессов (BPMN, Business Process Model and Notation) – методология моделирования, анализа и реорганизации бизнес-процессов. Разработана Business Process Management Initiative (BPMI), с 2005 г. поддерживается и развивается Object Management Group (OMG). В отличие от других методологий бизнес-моделирования, имеющих статус «фирменного»

(EPC) или «национального» (IDEF0) стандарта, BPMN получила «международный» статус – Международная организация по стандартизации опубликовала стандарт «ISO/IEC 19510:2013. Information technology – Object Management Group. Business Process Model and Notation» [36, 37].

Основной целью BPMN является обеспечение доступной нотацией описания бизнес-процессов всех пользователей: от аналитиков, создающих схемы процессов, и разработчиков, ответственных за внедрение технологий выполнения бизнес-процессов, до руководителей и обычных пользователей, управляющих этими бизнес-процессами и отслеживающих их выполнение. Таким образом, BPMN нацелена на устранение расхождения между моделями бизнес-процессов и их реализацией.

Диаграмма процесса в нотации BPMN представляет собой алгоритм выполнения процесса. На диаграмме могут быть определены события, исполнители, материальные и документальные потоки, сопровождающие выполнение процесса. Каждый процесс может быть декомпозирован на более низкие уровни. Декомпозиция может производиться в нотациях BPMN или EPC. При декомпозиции процесса BPMN, расположенного на диаграмме SADT, стрелки с диаграммы SADT на диаграмму BPMN не переносятся.

В нотации BPMN выделяют пять основных категорий элементов:

- элементы потока (Flow Objects) (события, процессы и шлюзы);
- данные (объекты данных и базы данных) (товарно-материальные ценности (ТМЦ) или информация);
- соединяющие элементы (Connecting Objects) (потоки управления, потоки сообщений и ассоциации);
- зоны ответственности (Swimlanes) (пулы и дорожки);
- артефакты (сноски) (Artifacts).

Элементы потока являются важнейшими графическими элементами, определяющими ход бизнес-процесса. Элементы потока, в свою очередь, делятся на:

- события (Events);
- действия (Activities);
- шлюзы (Gateways).

Выделяют три вида **соединяющих элемента** потока, связывающихся друг с другом и с другими элементами:

- поток операций (Sequence Flow);
- поток сообщений (Message Flow);
- ассоциация (Association).

Существуют два способа группировки основных элементов моделирования с помощью **зон ответственности**:

- группировка с помощью Пула (Pool);
- группировка с помощью Дорожки (Lane).

Артефакты используются для добавления дополнительной информации о Процессе.

Выделяют три типовых Артефакта, что, однако, не запрещает разработчикам моделей бизнес-процессов либо программам моделирования добавлять необходимое количество Артефактов. Для широкого круга пользователей, а также для вертикальных рынков существует возможность стандартизации более полного перечня Артефактов. Таким образом, текущий перечень Артефактов включает в себя следующие элементы:

- объект данных (Data object);
- группа (Group);
- аннотация (Annotation).

Все многообразие процессов и способов взаимодействия между их участниками в BPMN поделено на типы (sub-model). Каждому из типов соответствует своя семантика и набор отображаемых элементов.

Разновидности диаграмм (типы процессов) BPMN:

- диаграмма процессов (Process Diagram):
 - частный (внутренний) бизнес-процесс (Private (internal) Business Process), рис.1.24;
 - публичный (открытый) процесс (Public Process), рис.1.25;
- диаграмма хореографии (Choreography Diagram), рис.1.26;
- диаграмма взаимодействия (Collaboration Diagram):
 - процессов (Process), рис.1.27;
 - посредством обмена сообщениями (A view of Conversations), рис. 1.28.

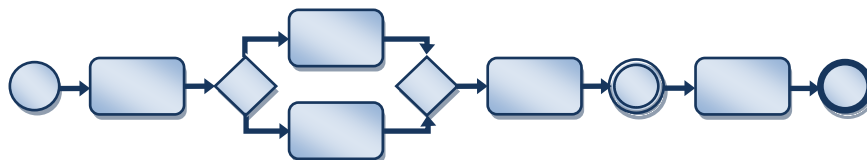


Рис. 1.24. Примерный вид BPMN диаграммы частного (внутреннего) бизнес-процесса.

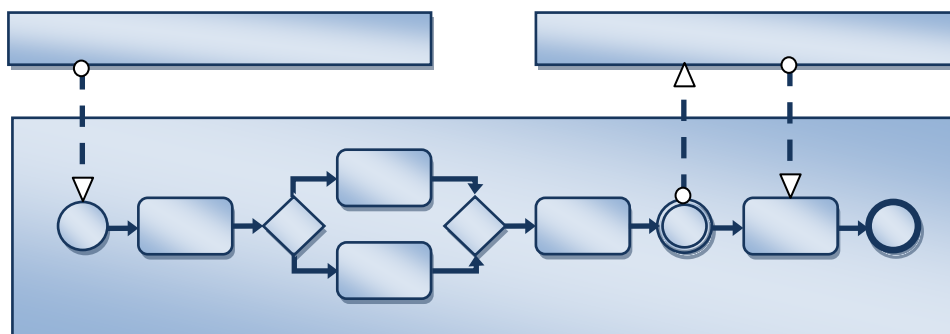


Рис.1.25. Примерный вид BPMN диаграммы публичного (открытого) процесса.

Процесс моделирования процессов с помощью BPMN подчиняется классическим принципам моделирования: декомпозиции и иерархического упорядочивания. Декомпозиция, с отображением на отдельных диаграммах, выполняется для участников (пулов) и отдельных подпроцессов, подобно работам на диаграммах IDEF0 или предопределенным процессам на блок-схемах.

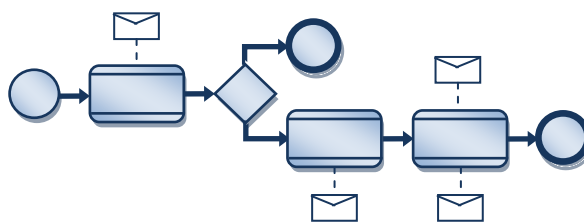


Рис.1.26.Примерный вид BPMN диаграммы хореографии.

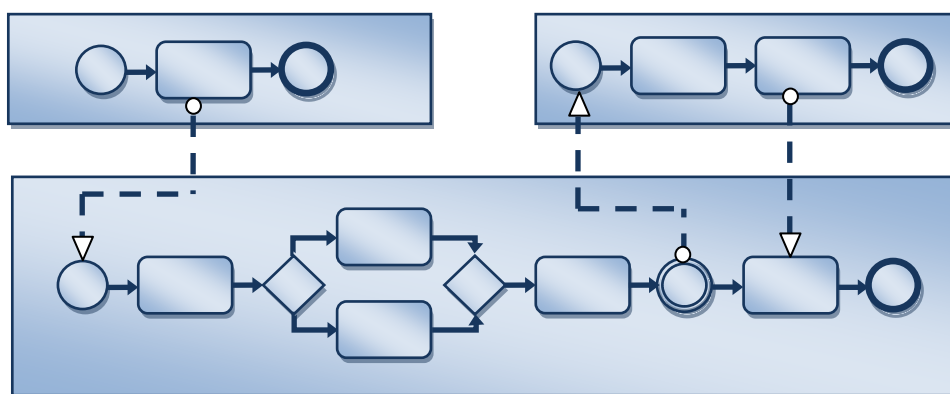


Рис.1.27. Примерный вид BPMN диаграммы взаимодействия процессов.

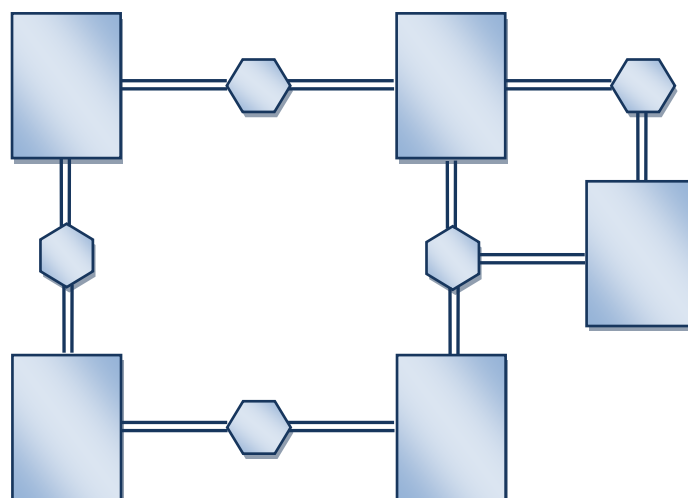


Рис.1.28.Примерный вид BPMN диаграммы посредством обмена сообщениями.

По заявлению разработчиков стандарта BPMN, он вообрал в себя лучшие идеи, что имеются в следующих нотациях и методологиях моделирования:

- UML (Unified Modeling Language, унифицированный язык моделирования):
- Activity Diagram (диаграмма деятельности);
- EDOC (Enterprise Distributed Object Computing, корпоративная распределенная обработка объектов) – Business Processes (бизнес-процессы);

- IDEF (SADT);
- ebXML (Electronic Business eXtensible Markup Language), расширяемый язык разметки для электронного бизнеса) BPSS (Business Process Specification Schema), схемы спецификации бизнес-процессов;
- ADF (Activity-Decision Flow, поток «деятельность-результат») Diagram;
- RosettaNet;
- LOVEM (Line of Visibility Engineering Methodology, визуальная методология проектирования);
- EPC.

Поддержка и дальнейшее развитие BPMN организацией OMG наложило свой «отпечаток» на данную методологию. Одним из ключевых направлений OMG является продвижение UML, предназначенного для моделирования объектно-ориентированных систем. В связи с этим, в BPMN при моделировании (разработке диаграмм), помимо понятий и концепций структурного подхода (действие, поток управления, объект данных и т.д.), используются такие характерные для объектно-ориентированного подхода понятия, как сообщение, обмен сообщениями и поток сообщений.

1.9.2. Методологии модельно-ориентированного подхода (анализа)

Модельно-ориентированное проектирование (МОП) – эффективный и экономически выгодный способ разработки систем управления и создания встраиваемых систем. Вместо физических прототипов и текстовых спецификаций в модельно-ориентированном проектировании применяется исполняемая модель. Эта модель используется во всех этапах разработки. При таком подходе можно разрабатывать и проводить имитационное моделирование как всей системы целиком, так и ее компонентов. Автоматическая генерация программного кода позволяет избежать большинства ошибок связанных с человеческим фактором и уменьшить время разработки более чем в два раза.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретной системы.

Типовая ИС в специальной базе метаинформации – *репозитории* – содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Таким образом, модельно-ориентированное проектирование ИС предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler). Возможно также создание системы на базе типовой модели ИС из репозитория, который поставляется вместе с программным продуктом и

расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.

Репозиторий содержит базовую (ссылочную) модель ИС, типовые (референтные) модели определенных классов ИС, модели конкретных ИС предприятий.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.

Модель конкретной системы строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями объекта автоматизации (BAAN Enterprise Modeler), либо путем автоматизированной адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).

Построенная модель в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе этой модели автоматически осуществляется конфигурирование и настройка информационной системы.

Принципы МОП существенно отличаются от традиционной методологии проектирования. Вместо создания сложных программных кодов разработчики могут применять МОП для улучшения характеристик модели, используя стандартные функциональные блоки с непрерывным и дискретным временем. Построенные таким образом модели вместе с использованием инструментов для моделирования, могут привести к созданию прототипа системы управления, тестированию и верификации программного обеспечения. В некоторых случаях аппаратно-программное моделирование может быть использовано в качестве инструмента проектирования для более быстрого и эффективного тестирования динамических воздействий на систему, в отличие от традиционного метода проектирования.

Преимущества МОП перед традиционным подходом проектирования:

- МОП предоставляет общую среду разработки, что способствует взаимодействию группы разработчиков в процессе анализа данных и проверки системы;
- инженеры могут найти и исправить ошибки на ранних стадиях проектирования системы, когда затраты времени и финансовые последствия изменения системы сводятся к минимуму;
- МОП способствует повторному использованию моделей для улучшения системы и создания производных систем с расширенными возможностями.

1.9.2.1. Методология прототипного создания приложений – RAD (Rapid Application Development)

Методология разработки информационных систем, основанная на использовании средств быстрой разработки приложений, получила широкое

распространение и приобрела название методологии быстрой разработки приложений – RAD (Rapid Application Development) [10].

Данная методология *охватывает все этапы жизненного цикла* современных информационных систем.

RAD – это комплекс специальных инструментальных средств быстрой разработки прикладных информационных систем, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений.

Под методологией быстрой разработки приложений обычно понимается процесс разработки информационных систем, основанный на трех основных элементах:

- небольшой команде программистов (обычно от 2 до 10 человек);
- тщательно проработанный производственный график работ, рассчитанный на сравнительно короткий срок разработки (от 2 до 6 мес.);
- итерационная модель разработки, основанная на тесном взаимодействии с заказчиком – по мере выполнения проекта разработчики уточняют и реализуют в продукте требования, выдвигаемые заказчиком.

При использовании методологии RAD большое значение имеют опыт и профессионализм разработчиков. Группа разработчиков должна состоять из профессионалов, имеющих опыт в анализе, проектировании, программировании и тестировании программного обеспечения.

Основные принципы методологии RAD можно свести к следующему:

- используется итерационная (спиральная) модель разработки;
- полное завершение работ на каждом из этапов жизненного цикла не обязательно;
- в процессе разработки информационной системы необходимо тесное взаимодействие с заказчиком и будущими пользователями;
- необходимо применение CASE-средств и средств быстрой разработки приложений;
- необходимо применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимо использование прототипов, позволяющее полнее выяснить и реализовать потребности конечного пользователя;
- тестирование и развитие проекта осуществляются одновременно с разработкой;
- разработка ведется немногочисленной и хорошо управляемой командой профессионалов;
- необходимы грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

1.9.2.2. Методология синхронизации и стабилизации MSF (модель синхростабилизации Microsoft – Microsoft Solution Framework)

Microsoft Solutions Framework (модель разработки приложений Microsoft) – это набор концепций и рекомендуемых моделей, которые позволяют

разрабатывать и внедрять информационные системы на основе технологий и инструментальных средств Microsoft. Многие концепции MSF хорошо известны. MSF является одной из интерпретаций спиральной (циклической) модели разработки приложений и базируется на практических результатах организации распределенных вычислений и применения технологий «клиент-сервер» компании Microsoft, ее партнеров и заказчиков [38].

Главной целью MSF, как и любой методологии проектирования приложений, является создание рабочего приложения вовремя и в рамках установленного бюджета. MSF предлагает хорошо зарекомендовавшие себя практики планирования, разработки и внедрения информационных технологий. В то же время MSF не является простым набором инструкций, которым полагается следовать безоговорочно – этот процесс достаточно гибок и расширяем. Методология MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов.

Основные компоненты и модели MSF

MSF содержит следующие модели:

- Team Model (Модель команды) – описывает коллектив, в котором работа одного сотрудника зависит от другого;
- Process Model (Модель процесса) – позволяет определить принципы планирования и контроля проектов;
- Application Model (Модель приложения) – помогает создавать приложения, максимально используя готовые компоненты;
- Enterprise Architecture Model (Модель архитектуры корпорации) – обеспечивает принятие решения по технологиям; она очень важна для эффективного использования новых технологий;
- Solution Design Model (Модель проектирования решений) – показывает, каким должно быть приложение с точки зрения пользователя. Эта модель связывает приложение, команду разработчиков и процесс разработки;
- Infrastructure Model (Модель управления инфраструктурой) – определяет принципы управления пользователями в больших сетях;
- Total Cost of Ownership Model (Модель стоимости владения продуктом) – позволяет оценивать расходы на информационные технологии.

Базовыми компонентами методологии являются:

– Solution Development Discipline (SDD) – дисциплина разработки решений. Содержание этой дисциплины связано с уникальными моделями: моделью команды и моделью процесса, которые рекомендуется использовать для организации эффективных команд проектов и управления жизненным циклом проекта. SDD включает три фундаментальные модели MSF:

- масштабируемая модель команды разработки – описывает принципы организации группы людей, ответственных за разработку приложения,
- итеративная модель процесса разработки – описывает, как должен быть организован процесс,

- сетевая трехслойная модель приложения – описывает, какой должна быть структура приложения, удовлетворяющего современным требованиям;
- Designing Component Solutions (DCS) – проектирование компонентного ПО. Эта дисциплина направлена на поддержку процесса проектирования сложных моделей распределенных вычислений;
- Enterprise Architecture Planning – планирование архитектуры предприятия. С точки зрения Microsoft, это итеративный процесс, сосредоточенный на долгосрочном планировании, но при этом направленный на достижение результатов в максимально сжатые сроки;
- Infrastructure Deployment and Management – управление технологической инфраструктурой. Эта дисциплина содержит подход к процессу внедрения в масштабах предприятия, как новых информационных технологий, так и отдельных программных продуктов и приложений.

1.9.2.3. Методология рационального унифицированного процесса RUP (Rational Unified Process)

Еще одной ведущей методологией, в которой инструментально *поддерживаются все этапы жизненного цикла* разработки ПО, является методология Rational Unified Process (RUP). Она опирается на проверенные практикой методы анализа, проектирования и разработки ПО, методы управления проектами. RUP обеспечивает прозрачность и управляемость процесса и позволяет создавать ПО в соответствии с требованиями заказчика на момент сдачи ПО, а также в соответствии с возможностями инструментальных средств поддержки разработки.

В основе методологии RUP, как и многих других программных методологий, объединяющих инженерные методы создания ПО, лежит «пошаговый подход». Он определяет этапы жизненного цикла, контрольные точки, правила работ для каждого этапа и, тем самым, упорядочивает проектирование и разработку ПО.

Для каждого этапа жизненного цикла методология задает:

- состав и последовательность работ, а также правила их выполнения;
- распределение полномочий среди участников проекта (роли);
- состав и шаблоны формируемых промежуточных и итоговых документов;
- порядок контроля и проверки качества.

Методология RUP позволяет объединить проектную команду, предоставляя в ее распоряжение проверенные мировой практикой лучшие подходы к разработке ИС. К ним относятся такие процессы жизненного цикла создания ПО, как управление проектами, бизнес-моделирование, управление требованиями, анализ и проектирование, тестирование и контроль изменений.

Вся разработка ПО рассматривается в RUP как процесс создания артефактов. Любой результат работы проекта, будь то исходные тексты, объектные модули, документы, передаваемые пользователю, модели – это

подклассы всех артефактов проекта. Каждый член проектной группы создает свои артефакты и несет за них ответственность. Программист создает программу, руководитель – проектный план, а аналитик – модели системы. RUP позволяет определить когда, кому и какой артефакт необходимо создать, доработать или использовать.

Одним из классов артефактов проекта являются модели, которые позволяют разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем. Каждая модель является самодостаточным взглядом на разрабатываемую систему и предназначена как для очерчивания проблем, так и для предложения решения. Самодостаточность моделей означает, что аналитик или разработчик может из конкретной модели почерпнуть всю необходимую ему информацию, не обращаясь к другим источникам.

Модели позволяют рассмотреть будущую систему, ее объекты и их взаимодействие еще до вкладывания значительных средств в разработку, позволяют увидеть ее глазами будущих пользователей снаружи и разработчиков изнутри еще до создания первой строки исходного кода. Большинство моделей представляются UML диаграммами [39].

Не следует забывать, что язык моделирования дает только нотацию – инструмент описания и моделирования системы, а унифицированный процесс определяет методику использования этого инструмента

Унифицированный язык моделирования (Unified Modeling Language) появился в конце 80-х в начале 90-х годов в основном благодаря усилиям Гради Буча, Джима Рамбо и Ивара Якобсона. В настоящее время консорциум OMG принял этот язык как стандартный язык моделирования, который предоставляет разработчикам четкую нотацию, позволяющую отображать модели общепринятыми и понятными каждому члену проекта графическими элементами.

Определение требований. Унифицированный процесс – это процесс, управляемый прецедентами, которые отражают сценарии взаимодействия пользователей. Фактически, это взгляд пользователей на программную систему снаружи. Таким образом, одним из важнейших этапов разработки, согласно RUP, будет этап определения требований, который заключается в сборе всех возможных пожеланий к работе системы, которые только могут прийти в голову пользователям и аналитикам. Позднее эти данные должны будут систематизированы и структурированы, но на данном этапе в ходе интервью с пользователями и изучения документов, аналитики должны собрать как можно больше требований к будущей системе, что не так просто, как кажется на первый взгляд.

Пользователи часто сами не представляют, что они должны получить в конечном итоге. Для облегчения этого процесса аналитики используют Диаграммы Прецедентов (Вариантов использования) (Use Case Diagram), рис.1.29. На диаграмме отображаются варианты использования (1) и действующие лица (2), между которыми устанавливаются следующие основные типы отношений: ассоциация между действующим лицом и вариантом использования (3); обобщение между действующими лицами (4); обобщение между вариантами

использования (5); зависимости (различных типов) между вариантами использования (6).

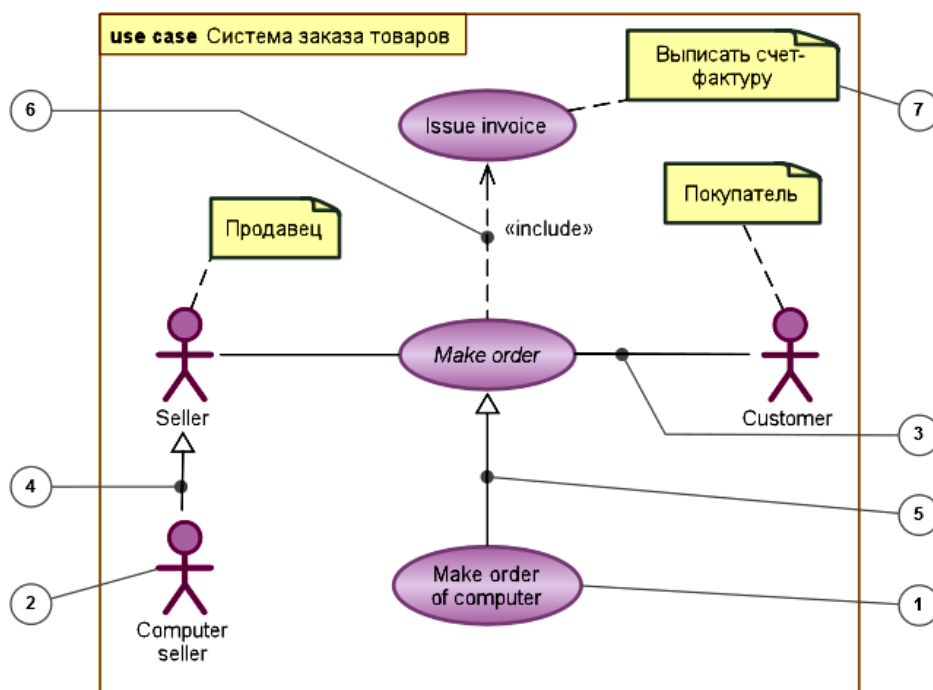


Рис.1.29. Диаграммы Прецедентов (Вариантов использования) (Use Case Diagram).

Диаграмма представляет собой отражение действующих лиц (актантов, акторов), которые взаимодействуют с системой, и реакцию программных объектов на их действия. Актантами могут быть как пользователи, так и внешние агенты, которым необходимо передать, или получить информацию. Значок варианта использования отражает реакцию системы на внешнее воздействие и показывает, что должно быть сделано для актанта.

Для детализации конкретного прецедента используется Диаграмма Активности (Activity Diagram), пример которой дан на рис.1.30. На диаграмме применяют один основной тип сущностей – действие (1), и один тип отношений – переходы (2) (передачи управления и данных). Также используются такие конструкции как развилки, слияния, соединения, ветвления (3), которые похожи на сущности, но таковыми на самом деле не являются, а представляют собой графический способ изображения некоторых частных случаев многоместных отношений.

Для того чтобы верно определить требования, разработчики должны понимать контекст (часть предметной области) в котором будет работать будущая система. Для этого создаются модель предметной области и бизнес-модель, что является различными подходами к одному и тому же вопросу. Часто создается что-то одно: модель предметной области или бизнес-модель.

Отличия этих моделей в том, что модель предметной области описывает важные понятия, с которыми будет работать система и связи их между собой. Тогда как бизнес-модель описывает бизнес-процессы (существующие или будущие), которые должна поддерживать система. Поэтому кроме определения

бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности и действия, которые они должны выполнять.

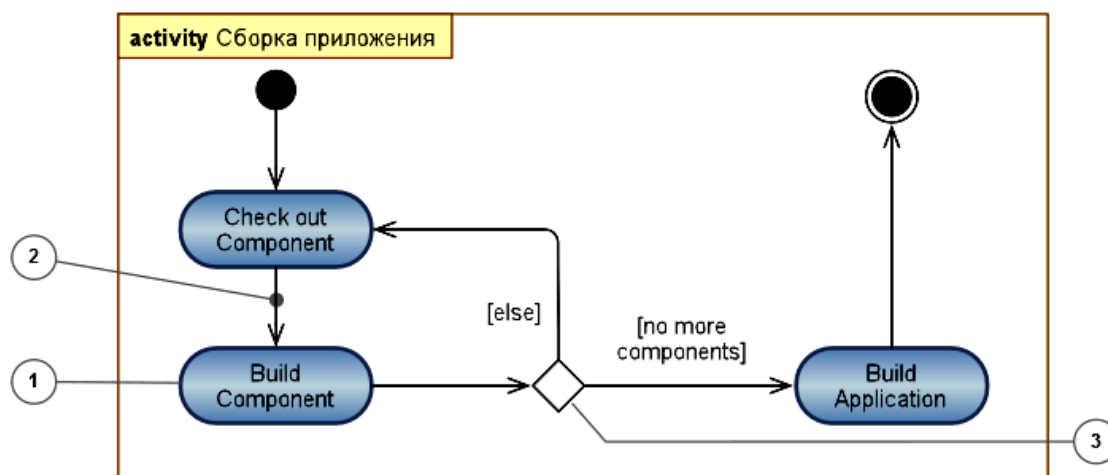


Рис.1.30. Диаграмма Активности (Activity Diagram).

Для создания модели предметной области используется обычная Диаграмма Классов (Class Diagram), рис.1.31, на которой отображаются: основной тип сущностей: классы (1) (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений: ассоциация между классами (2) (с множеством дополнительных подробностей); обобщение между классами (3); зависимости (различных типов) между классами 4 и между классами и интерфейсами.).

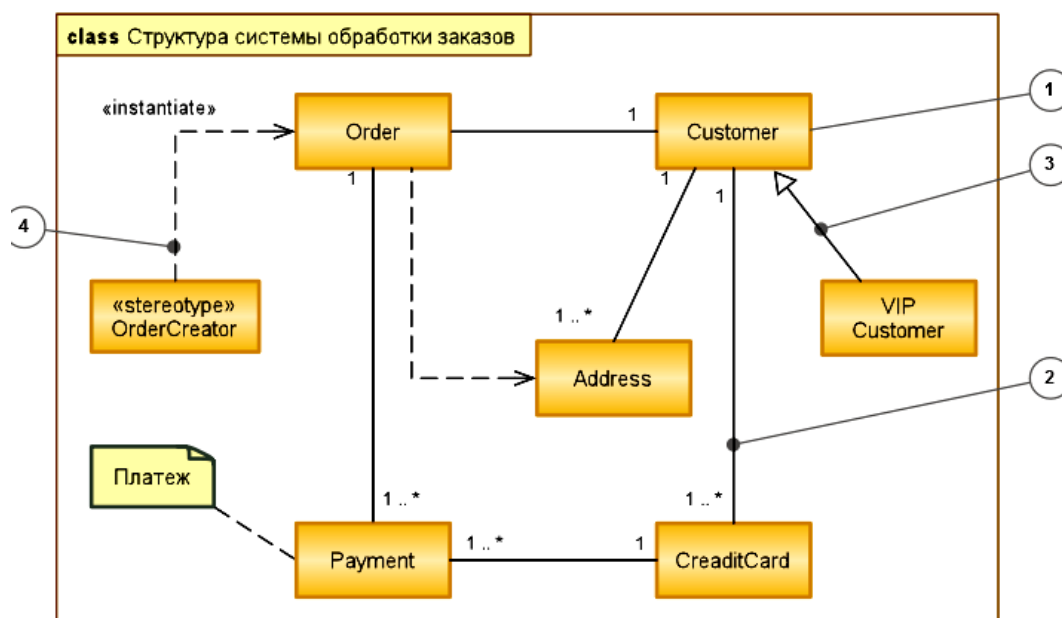


Рис.1.31. Диаграмма Классов (Class Diagram).

Однако для создания бизнес-модели Диаграммы Классов уже явно недостаточно. В этом случае применяется диаграмма прецедентов с использованием дополнительных значков, которые отражающие сущность

бизнес-процессов – это бизнес-актант, бизнес-прецедент, бизнес-сущность и бизнес-управление. Эта модель намного ближе к следующей модели, создаваемой в процессе разработки – модели анализа.

Анализ. После определения требований и контекста, в котором будет работать система, наступает черед анализа полученных данных. В процессе анализа создается аналитическая модель, которая подводит разработчиков к архитектуре будущей системы. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели прецедентов, которая показывает, как система будет выглядеть снаружи.

Эта модель позволяет понять, как система должна быть спроектирована, какие в ней должны быть классы и как они должны взаимодействовать между собой. Основное ее назначение – определить направление реализации функциональности, выявленной на этапе сбора требований и сделать набросок архитектуры системы.

В отличие от создаваемой в дальнейшем модели проектирования, модель анализа является в большей степени концептуальной моделью и только приближает разработчиков к классам реализации. Эта модель не должна иметь возможных противоречий, которые могут встретиться в модели прецедентов.

Для отображения модели анализа при помощи UML используется Диаграмма Классов со стереотипами (образцами поведения) «граничный класс», «сущность», «управление», а для детализации используются Диаграммы Сотрудничества (Collaboration), рис.1.32.

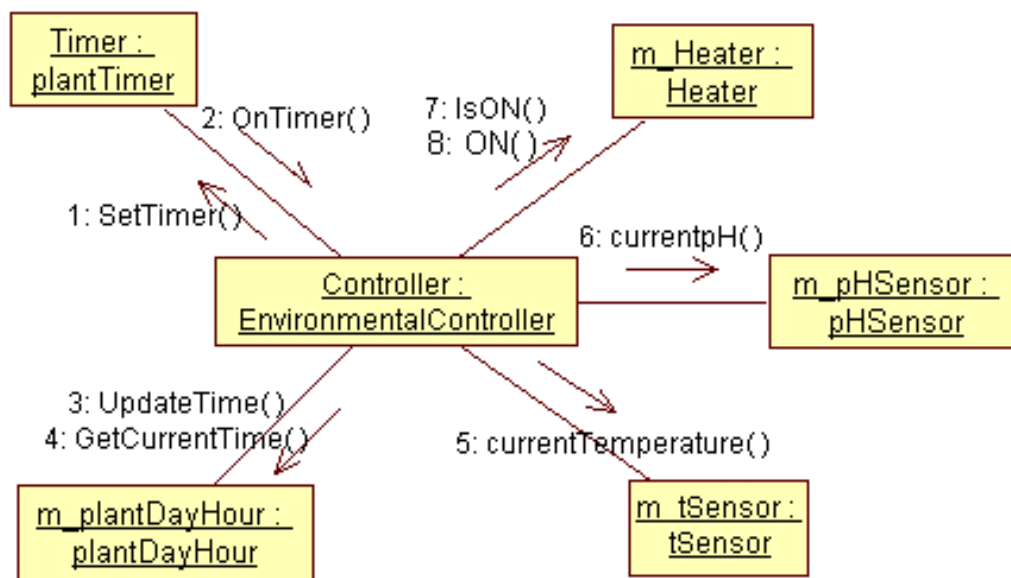


Рис.1.32. Диаграммы Сотрудничества (Collaboration).

Стереотип «граничный класс» отображает класс, который взаимодействует с внешними актантами, «сущность» – отображает классы, которые являются хранилищами данных, а «управление» – классы, управляющие запросами к сущностям.

Нумерация сообщений показывает их порядок, однако назначение диаграммы не в том, чтобы рассмотреть порядок обмена сообщениями, а в том, чтобы наглядно показать связи классов друг с другом.

Если акцентировать внимание на порядке взаимодействия, то другим его представлением будет Диаграмма Последовательности (Sequence Diagram), рис.1.33.

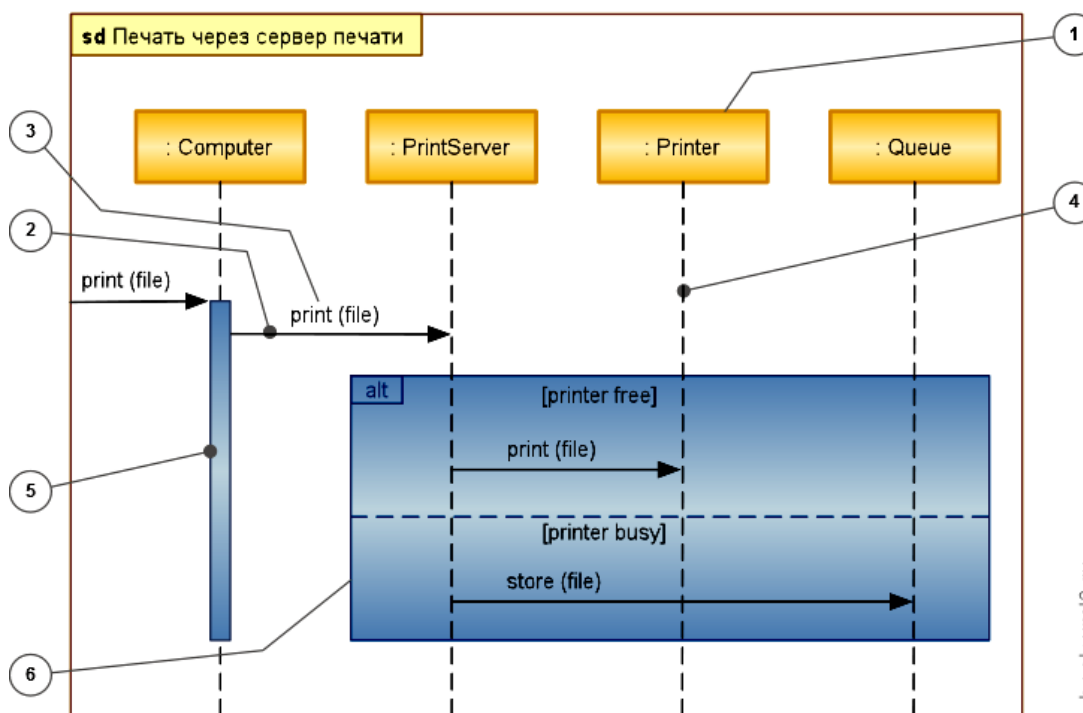


Рис.1.33. Диаграмма Последовательности (Sequence Diagram).

Эта диаграмма позволяет взглянуть на обмен сообщениями во времени, наглядно отобразить последовательность процесса. На диаграмме применяют один основной тип сущностей – экземпляры взаимодействующих классификаторов (1) (в основном классов, компонентов и действующих лиц), и один тип отношений – связи (2), по которым происходит обмен сообщениями (3). Предусмотрено несколько способов посылки сообщений, которые в графической нотации различаются видом стрелки, соответствующей отношению. Пунктирная линия, выходящая из прямоугольник с именем экземпляра классификатора (1), называется линией жизни (lifeline) (4). Фигуры в виде узких полосок, наложенных на линию жизни, также не являются изображениями моделируемых сущностей. Это графический комментарий, показывающий отрезки времени, в течении которых объект владеет потоком управления (execution occurrence) (5) или другими словами имеет место активация (activation) объекта. Составные шаги взаимодействия (combined fragment) (6) позволяют на диаграмме последовательности, отражать и алгоритмические аспекты протокола взаимодействия.

Решение о том, какую из двух диаграмм: Сотрудничества или Последовательности нужно создавать первой, зависит от предпочтений конкретного разработчика. Поскольку эти диаграммы являются отображением

одного и того же процесса, то и та и другая позволяют отразить взаимодействие между объектами.

Проектирование. Следующим этапом в процессе создания системы будет проектирование, в ходе которого на основании моделей, созданных ранее, создается модель проектирования. Эта модель отражает физическую реализацию системы и описывает создаваемый продукт на уровне классов и компонентов. В отличие от модели анализа, модель проектирования имеет явно выраженную зависимость от условий реализации, применяемых языков программирования и компонентов. Для максимально точного понимания архитектуры системы, эта модель должна быть максимально формализована, и поддерживаться в актуальном состоянии на протяжении всего жизненного цикла разработки системы.

Для создания модели проектирования используются целый набор UML диаграмм: Диаграммы Активности, рис.1.30, Диаграммы Классов, рис.1.31, Диаграммы Коммуникации, рис.1.34, Диаграммы Взаимодействия, рис.1.35.

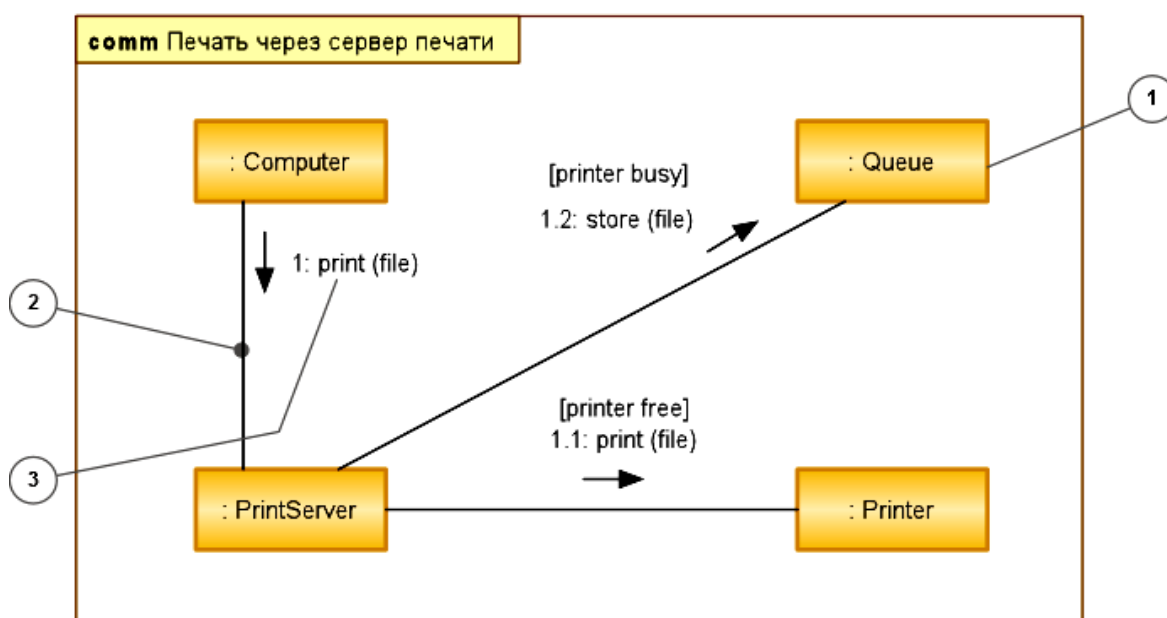


Рис.1.34. Диаграммы Коммуникации (Communication Diagram).

Дополнительно в этом рабочем процессе может создаваться модель развертывания, которая реализуется на основе Диаграммы Развертывания (Deployment Diagram), рис.1.36. Это самый простой тип диаграмм, предназначенный для моделирования распределения устройств в сети. На диаграмме присутствует два типа сущностей: артефакт (1), который является реализацией компонента (2) и узел (3) (может быть как классификатор, описывающий тип узла, так и конкретный экземпляр), а также отношение ассоциации между узлами (4), показывающее, что узлы физически связаны во время выполнения. Если одна сущность является частью другой, применяется либо отношение зависимости «deploy» (5), либо фигура одной сущности помещается внутри фигуры другой сущности (6).

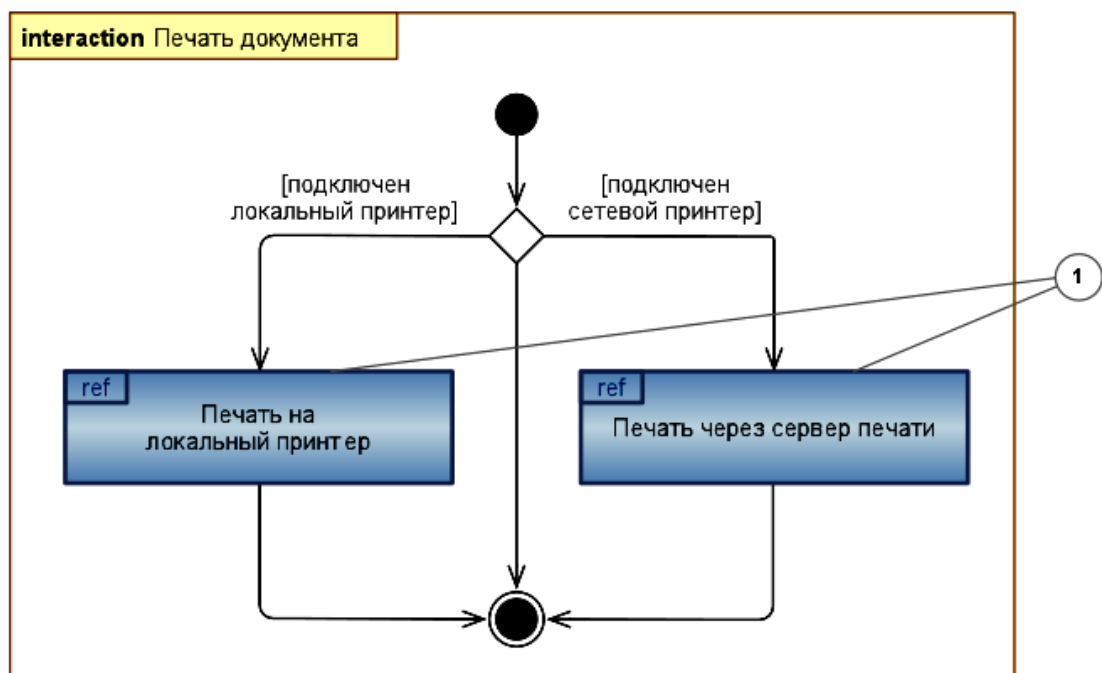


Рис.1.35. Диаграммы Взаимодействия.

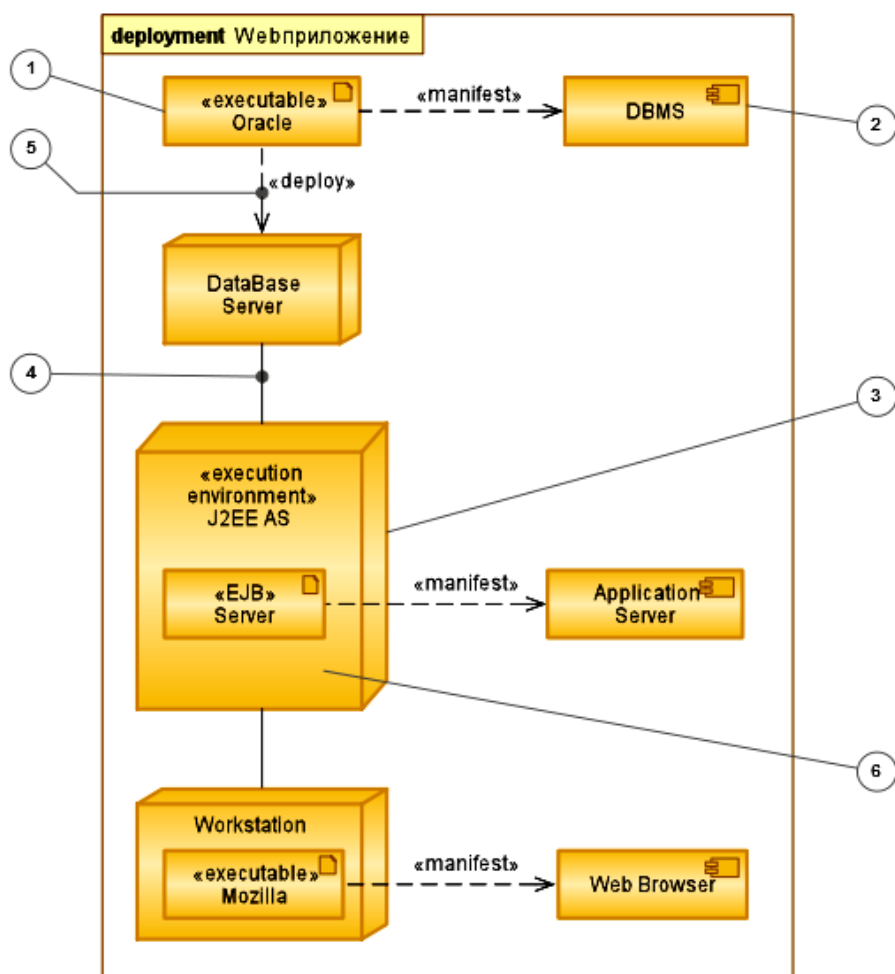


Рис.1.36. Диаграммы Развертывания (Deployment Diagram).

Реализация. Основная задача процесса реализации – создание системы в виде компонентов – исходных текстов программ, сценариев, двоичных файлов,

исполняемых модулей и т.д. На этом этапе создается модель реализации, которая описывает то, как реализуются элементы модели проектирования, какие классы будут включены в конкретные компоненты. Данная модель описывает способ организации этих компонентов в соответствии с механизмами структурирования и разбиения на модули, принятыми в выбранной среде программирования и представляется Диаграммой Компонентов (Component Diagram), рис.1.37.

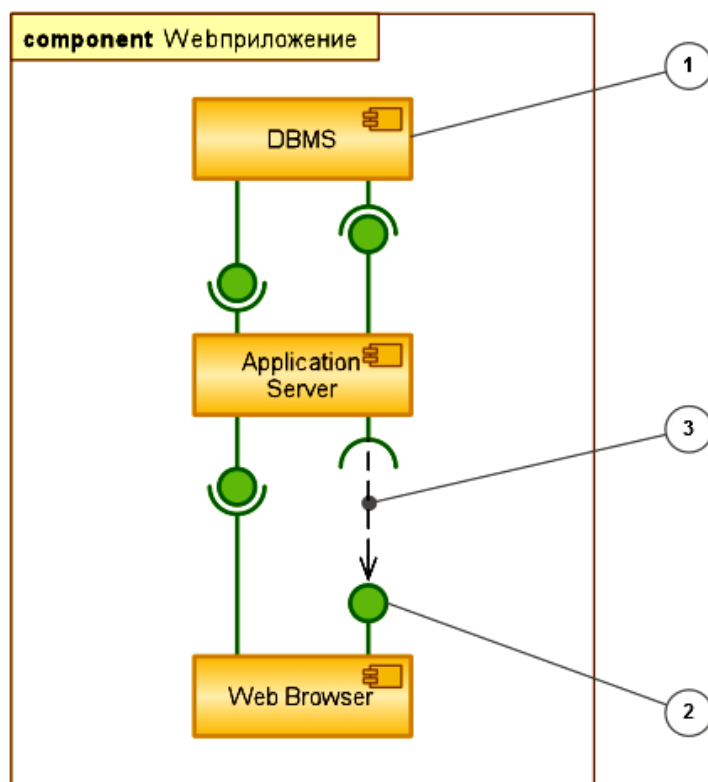


Рис.1.37. Диаграммы Компонентов (Component Diagram).

Основной тип сущностей на диаграмме компонентов – это сами компоненты (1), а также интерфейсы (2), посредством которых указывается взаимосвязь между компонентами. На диаграмме компонентов применяются следующие отношения:

- реализации между компонентами и интерфейсами (компонент реализует интерфейс);
- зависимости между компонентами и интерфейсами (компонент использует интерфейс) (3).

Тестирование. В процессе тестирования проверяются результаты реализации. Для данного процесса создается модель тестирования, которая состоит из тестовых примеров, процедур тестирования, тестовых компонентов, однако не имеет отображения на UML диаграммы.

RUP довольно обширен, и содержит рекомендации по ведению различных программных проектов, от создания программ группой разработчиков в несколько человек, до распределенных программных проектов, объединяющих тысячи человек на разных континентах. Однако, несмотря их колоссальную разницу, методы применения моделей, создаваемых при помощи UML будут одни

и те же. Диаграммы UML, создаваемые на различных этапах разработки, неотделимы от остальных артефактов программного проекта и часто являются связующим звеном между отдельными процессами RUP.

Решение о применении конкретных диаграмм зависит от поставленного в компании процесса разработки, который, хотя и называется унифицированным, однако не есть нечто застывшее. Компания Rational не только предлагает его улучшать и дорабатывать, но и предоставляет специальные средства внесения изменений в базу данных RUP.

1.9.2.4. Методология разработки информационных систем DATARUN

DATARUN является наиболее распространенной в мире электронной методологией [21].

Методология DATARUN преследует две цели:

- определить стабильную структуру, на основе которой будет строиться ИС. Такой структурой является модель данных, полученная из первичных данных, представляющих фундаментальные процессы организации;
- спроектировать ИС на основании модели данных.

Сущность методологии DATARUN – построение комплекса взаимосвязанных моделей будущей системы. В методологии DATARUN разработка ИС рассматривается как процесс последовательной детализации и спецификации свойств системы в виде комплекса взаимосвязанных моделей. Для быстрого построения прототипов используются кодогенераторы.

В соответствии с методологией DATARUN ЖЦ ПО разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом ISO 12207 [8]:

- стадия формирования требований и планирования;
- стадия концептуального проектирования;
- стадия спецификации приложений;
- стадия разработки, интеграции и тестирования;
- стадия внедрения;
- стадии сопровождения и развития.

Каждую стадию кроме ее результатов должен завершать план работ на следующую стадию.

Сущность работ, проводимых на каждом шаге процесса проектирования ИС, сводится к построению комплекса взаимосвязанных моделей, каждая из которых, фиксирует принятые разработчиками проектные решения, специфицирующие определенные характеристики будущей системы.

Комплекс моделей, создаваемых в рамках методологии DATARUN, включает следующие модели, рис.1.38:

- *модель бизнес-процессов BPM* (Business Process Model) – функциональная модель выполняемых в организации операций. В этой модели документируются правила работы организации, выполнение которых должна обеспечить создаваемая информационная система. Строится функциональная (DFD) модель предметной области;

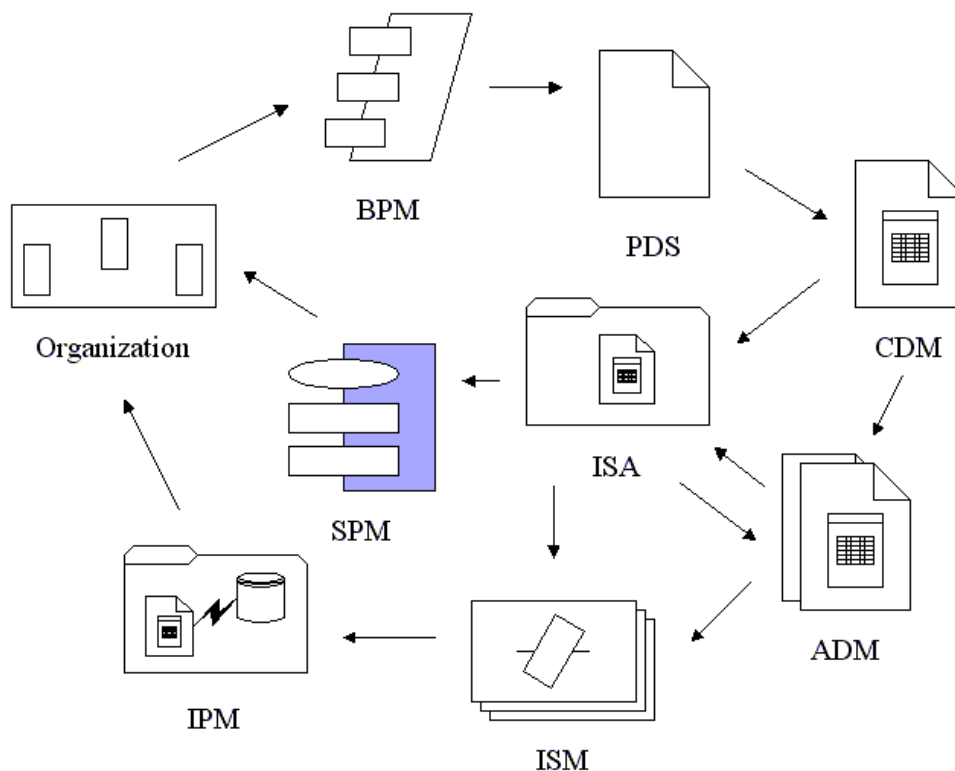


Рис. 1.38. Модели, создаваемые с помощью подхода DATARUN.

- *структуры первичных данных PDS (Primary Data Structure)* – структуры данных, описывающие основные сведения, необходимые для работы организации. Эти структуры данных являются основой построения базы данных информационной системы;

- *концептуальная модель данных CDM (Conceptual Data Model)* – модель «сущность-связь», описывающая систему понятий предметной области. Строится информационная (ER) модель предметной области;

- *модель процессов системы SPM (System Process Model)* – функциональная модель, описывающая структуру системы в виде комплекса взаимосвязанных процессов. Строятся функциональные (SADT, IDEF0, IDEF3, DFD, STD) модели предметной области;

- *модель архитектуры информационной системы ISA (Information System Architecture)* – функциональная модель, описывающая структуру приложений (подсистем) информационной системы, а также состав используемых ими данные;

- *модель данных приложения ADM (Application Data Model).*

- *модель представления интерфейса IPM (Interface Presentation Model)* – описание внешнего вида интерфейса, как его видит конечный пользователь системы. Это может быть и документ, показывающий внешний вид экрана или структуру отчета, и сам экран (отчет), созданные с помощью средств визуальной разработки приложений;

- *модель спецификации интерфейса ISM (Interface Specification Model)* – графическая модель для спецификации и синтеза интерфейса.

Информационная система создается последовательным построением ряда моделей, начиная с модели бизнес-процессов и заканчивая моделью программы, автоматизирующей эти процессы, рис.1.39.

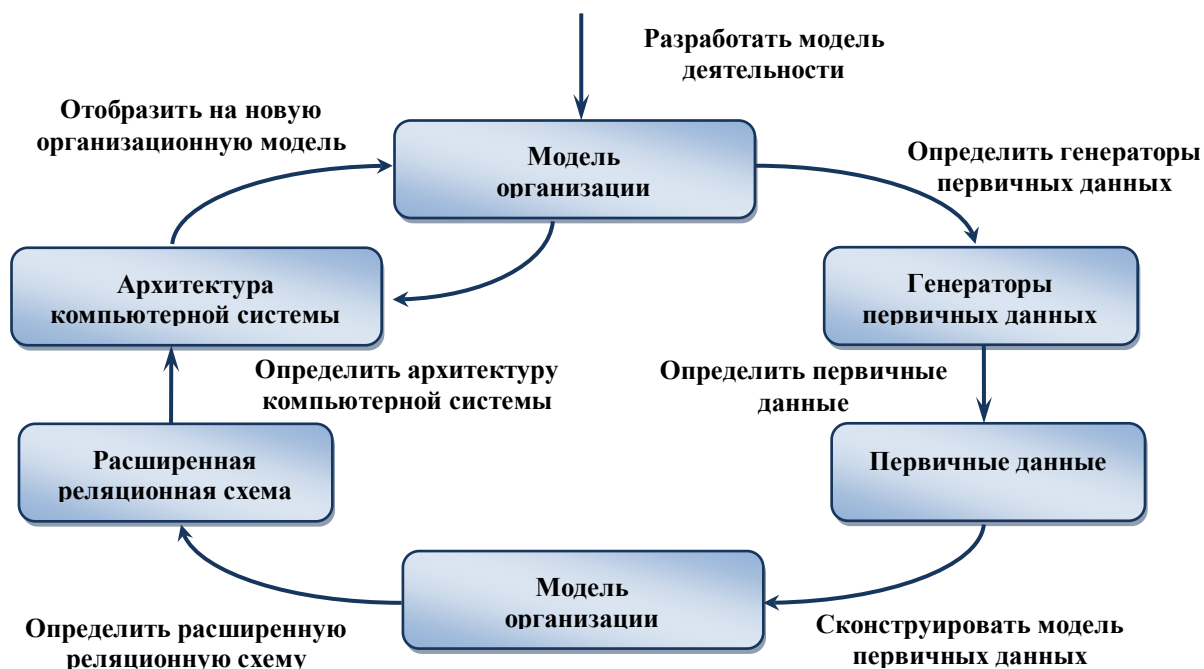


Рис.1.39. Последовательность шагов проектирования системы.

Характерными чертами методологии DATARUN являются:

- использование элементов объектно-ориентированного подхода;
- повторное использование спецификаций;
- графическое представление;
- непроцедурные спецификации;
- итеративный процесс разработки, реляционная основа;
- совместимость с технологией RAD.

1.9.2.5. Методология описания бизнес-процессов ORACLE (Oracle Method)

Методология Oracle (Oracle Method) – комплекс методов, охватывающий большинство процессов ЖЦ ПО [40]. В состав комплекса входят:

- CDM (Custom Development Method) – разработка прикладного ПО;
- PJM (Project Management Method) – управление проектом;
- AIM (Application Implementation Method) – внедрение прикладного ПО;
- BPR (Business Process Reengineering) – реинжиниринг бизнес-процессов;
- OCM (Organizational Change Management) – управление изменениями, и

др.

Custom Development Method (методика Oracle) по разработке прикладных информационных систем – технологический материал, детализированный до

уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle.

В соответствии с этими факторами в CDM выделяются два основных подхода к разработке:

- классический подход (Classic);
- подход быстрой разработки (Fast Track).

Классический подход (Classic) применяется для наиболее сложных и масштабных проектов, он предусматривает последовательный и детерминированный порядок выполнения задач. Для таких проектов характерно большое количество реализуемых бизнес-правил, распределенная архитектура, критичность приложения. Применение классического подхода также рекомендуется при нехватке опыта у разработчиков, неподготовленности пользователей, нечетко определенной задаче. Продолжительность таких проектов от 8 до 36 месяцев.

Подход быстрой разработки (Fast Track) в отличие от каскадного классического, является итерационным и основан на методе DSDM (Dynamic Systems Development Method). В этом подходе четыре этапа – стратегия, моделирование требований, проектирование и генерация системы и внедрение в эксплуатацию. Подход используется для реализации небольших и средних проектов с несложной архитектурой системы, гибкими сроками и четкой постановкой задач. Продолжительность проекта от 4 до 16 месяцев.

1.9.2.6. Методология проектирования интегрированных информационных систем ARIS (ARchitecture of Integrated Information Systems)

Концепция Архитектура Интегрированных Информационных Систем - ARIS (ARchitecture of Integrated Information Systems) разработана профессором А.В. Шеером (Scheer).

Эта концепция имеет два основных преимущества:

- позволяет выбрать методы и интегрировать их, опираясь на основные особенности моделируемого объекта;
- служит базой для управления сложными проектами, поскольку благодаря структурным элементам содержит встроенные модели процедур для разработки интегрированных информационных систем.

Такая архитектура дает возможность вводить в применяемые методы элементы стандартизации. Новые методы моделирования, а также те, в основе которых лежит концепция ARIS, были интегрированы в рамках архитектуры, что позволило создать комплексный метод моделирования бизнес-процессов.

Архитектура ARIS явилась основой ARIS Toolset – инструментальной среды, разработанной компанией IDS Scheer AG.

В методологии ARIS [41] для описания различных подсистем организации используется более ста типов моделей, отражающих различные аспекты деятельности и реализующих различные методы моделирования (в том числе событийная цепочка процесса EPC (Event driven Process Chain):

- модель «сущность-связь» ERM (Entity Relationship Model);
- модели методики объектно-ориентированного моделирования OMT (Object Modeling Technique);
- модели BSC (Balanced Scorecard) – система сбалансированных показателей;
- модели UML и многие другие.

Взаимосвязь моделей различных типов, образующих модель деятельности организации, обеспечивается за счет использования декомпозиции, а также применения принципа множественности экземпляров структурных объектов ARIS, представленных на моделях разных типов (определение объекта в репозитории ARIS всегда единственно).

Все многообразие типов моделей ARIS подразделяется на пять видов описания в соответствии с основными подсистемами предприятия:

- организационной;
- функциональной,
- подсистемой данных;
- подсистемой процессов;
- подсистемой продуктов/услуг,

Остальные подсистемы могут моделироваться с использованием типов объектов, входящих в перечисленные виды описания.

В свою очередь типы моделей внутри каждого вида описания подразделяются на три уровня в соответствии с этапами жизненного цикла КИС: определение требований к системе, спецификация проекта и описание реализации. Такая концепция обеспечивает целостное описание системы управления бизнесом, вплоть до ее технической реализации.

1.9.2.7. Методологии модельно-ориентированного проектирования интегрированных информационных систем (MathWorks)

Модельно-ориентированное проектирование способствует представлению проекта на основании требований, а также улучшенной степени интеграции и повторному использованию на этапах концептуального и детализированного моделирования и проектирования [42].

Отправной точкой модельно-ориентированного проектирования является применение инструмента Simulink® компании MathWorks, с помощью которого на стадии концептуального проектирования создаются модели всей системы, включающие алгоритмы и внешнюю среду. Эти модели можно симулировать и анализировать на протяжении всего процесса проектирования для обеспечения соответствия алгоритмов и спецификаций, на основании которых разрабатываются алгоритмы. Такой подход дает два преимущества:

- обнаружение и исправление ошибок на ранних стадиях проектирования обходится гораздо дешевле по сравнению с выявлением ошибок во время реализации и тестирования;

– результаты проектирования, тесты и анализ можно повторно использовать в течение всего процесса разработки.

На рис.1.40 представлен типичный процесс разработки встроенного программного обеспечения, включающий в себя составление требований, проектирование, реализацию, интеграцию и этапы тестирования. Непрерывная верификация, управление конфигурацией и изменениями распространяются на каждый этап разработки.

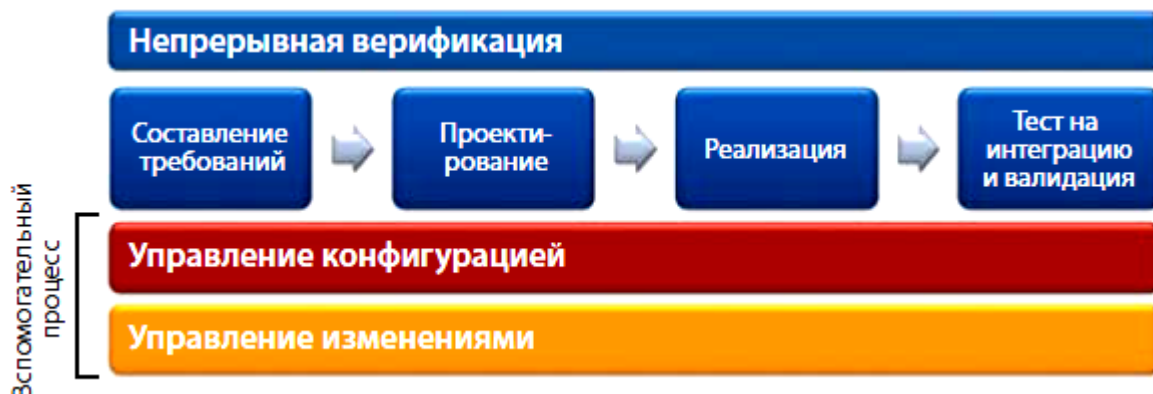


Рис.1.40. Типичный процесс разработки встроенных систем

Модельно-ориентированное проектирование предоставляет большой диапазон инструментов и методов для облегчения каждого этапа процесса разработки встроенного программного обеспечения.

Составления требований. Составление спецификации по требованиям – это процесс анализа и документирования требований и ограничений, которым должна удовлетворять разрабатываемая система. Возможность использовать модель для создания выполняемой спецификации и моделирования позволяет анализировать и проводить валидацию требований на ранних этапах и на ранее не достижимом глубоком уровне.

Проведение валидации требований перед переходом к рабочему проекту. Перед началом рабочего проекта большое количество времени следует потратить на получение, анализ и верификацию требований. Верификация часто предусматривает не только анализ требований. В случае новых или комплексных приложений она будет включать в себя моделирование и быстрое прототипирование для проверки правильности и полноты требований. В случае неудачного исхода необходимо повторить процесс проектирования. Если верификация невозможна без полностью продуманного проекта, можно использовать модель для документирования ожидаемого выхода для каждого из возможных входных воздействий и завершить проектирование упрощенной реализацией.

Взаимодействие с внутренними и внешними клиентами и поставщиками с помощью модели. Спецификации разрабатываются для того, чтобы облегчить взаимодействие между изготовителем комплектного оборудования (ОЕМ) и его поставщиком или между двумя подразделениями одной компании. Модель представляет собой недвусмысленно выполняемую спецификацию, которая

может служить идеальным инструментом взаимодействия. Инженеры могут использовать выполняемую спецификацию, чтобы избавиться от одного из наиболее общих источников ошибок – неправильного толкования или неправильного перевода спецификации.

Когда выполняемая спецификация используется совместно изготовителем комплектного оборудования и его поставщиком, поставщик должен в качестве контрольных точек проекта проводить проверку выполняемой спецификации вместе с изготовителем комплектного оборудования, показывая, как следует исполнять каждое требование, и подтверждая соответствие системы требованиям.

Модель как источник документации. Модель выполняет функцию ядра спецификации, но также ее можно использовать для производства другой документации, такой как обзорный тестовый отчет и, с использованием точной полученной информации, пользовательское руководство по настройке. Существуют инструменты для автоматизации процессов извлечения, перекомпоновки и представления информации, содержащейся в модели. Автоматизация помогает избежать необходимости подготовки многих не связанных с моделью проектных документов, которые могут плохо отражать текущее состояние проекта.

Этап проектирования. Проектирование – это процесс определения архитектуры и интерфейсов программного обеспечения и разработки детализированных функций и операций, удовлетворяющих требованиям. В модельно-ориентированном проектировании для описания проекта вместо проектных документов используются модели. Элементы, разработанные на этапе составления требований, такие как выполняемая спецификация, непосредственно являются отправной точкой для начального проектирования, тем самым снижая ошибки трансляции и ускоряя разработку. Проектные документы генерируются из модели. Компьютерное моделирование и быстрое прототипирование используются для быстрого повторения процедур разработки и подтверждения того, что проектирование идет согласно плану, и разработка будет удовлетворять требованиям как при моделировании, так и в реальных условиях

Проектирование концептуальной модели. По ранее собранным требованиям инженер-проектировщик конструирует исполняемую версию проекта. Simulink дает возможность инженерам создавать эти алгоритмические модели в интуитивно понятной графической среде. Дополнением к системе Simulink является инструмент Stateflow®, который используется для разработки конечных автоматов и логики. Дополнительные наборы блоков обеспечивают функционал более высокого уровня для специфических прикладных задач.

Симуляция работы крупной системы и среды, в которой функционирует аппаратура, позволяет инженерам выполнять полное тестирование системы до начала ее реализации. Например, рассмотрим проект алгоритма прерывания взлета. Этот алгоритм можно разрабатывать независимо в среде Simulink или как часть более высокоуровневой модели самолета. Такая модель системного уровня может включать в себя алгоритм логики, модель динамики летательного аппарата с шестью степенями свободы и учетом воздействия внешних факторов, моделями датчиков и моделями исполнительных механизмов.

Наличие системной модели позволяет специалистам тестировать свои проекты на более ранней стадии и оперативно оценивать сценарии по принципу «что, если». По мере возрастания уверенности в проектных решениях происходит детализация моделей с целью задания архитектуры изделия и включения эффектов реализации. Simulink предоставляет возможность симуляции этих эффектов и сравнения их с базовым проектом в плавающей точке, чтобы удостовериться в соблюдении заданных требований.

Трассируемость требований в концептуальной модели. В рабочем процессе при использовании модельно-ориентированного проектирования все элементы концептуального проекта должны трассироваться к требованиям, которые они удовлетворяют. MathWorks обеспечивает базовую поддержку трассируемости при помощи средства Simulink Verification and Validation™.

Верификация концептуальной модели. Концептуальный проект должен анализироваться для верификации выполнения заданных требований. В этой задаче могут оказать помощь несколько продуктов компании MathWorks. Например, система MATLAB® может применяться для выполнения скриптов, подбора значений параметров и выполнения анализа выходных результатов симуляции. Эти задачи могут запускаться параллельно на многоядерных компьютерах или кластерах с использованием возможностей параллельных и распределенных вычислений

Компания MathWorks разработала также специальные инструменты, ориентированные на верификацию систем. Средство SystemTest™ представляет собой платформу для тестирования, которую можно применять для создания и выполнения тестов для моделей в среде Simulink. Тесты можно создавать для демонстрации выполнения конкретных функциональных требований. SystemTest автоматически генерирует отчеты, которые можно рассматривать как артефакты верификации.

Симуляция помогает убедиться, что заданные требования выполняются, для чего проект подвергается испытаниям при различных условиях. Хотя проведение симуляции имеет важное значение, существенной проблемой в данном случае является вопрос полноты испытаний модели проекта при всех возможных условиях. Для обеспечения полного покрытия при функциональном тестировании можно использовать формальный анализ в сочетании с симуляцией для генерации тестовых векторов. Такие методики основываются на математически строгих процедурах для упрощения и поиска возможных путей выполнения модели, позволяя создавать тестовые вектора и контрпримеры. Этот систематический анализ обеспечивает более глубокое понимание поведения проектируемой системы.

Во время тестирования покрытие модели тестами может служить полезной метрикой, позволяющей оценить, насколько полно тесты охватывают модель. Инструмент Simulink Verification and Validation может анализировать покрытие модели и формировать соответствующие отчеты. Метрики по покрытию должны сначала собираться с помощью функциональных тестов, выполняемых на модели. Хотя функциональные тесты используются для подтверждения выполнения проектных требований, они зачастую не позволяют проверить проект на 100%.

Инструмент Simulink Design Verifier использует формальные методы для автоматической генерации тестовых векторов для дополнения функциональных тестов и достижения 100% модифицированного покрытия условий/решений (MC/DC) на уровне модели.

Этап реализации. Реализация – это процесс перевода разработки во встроенное программное обеспечение, которое можно запускать на целевом аппаратном обеспечении. В модельно-ориентированном проектировании перевод из разработки во встроенное программное обеспечение автоматизирован за счет генерации кода, которая значительно снижает количество ошибок и экономит время на написание кода. Основное направление разработки может сместиться в сторону создания интеллектуальной собственности, такой как модели разработки, которая является основой конкурентного преимущества и прибыльности компании.

Изменение модели вместо редактирования кода. К модели стоит относиться как к единственному эталону разработки. Изменять сгенерированный код напрямую не рекомендуется. Вместо этого следует вносить изменения в модель и повторно генерировать код, чтобы избежать лишних затрат и проблем с синхронизацией. В традиционных процессах это равносильно тому, чтобы воспринимать код C как источник и воздерживаться от внесения изменений на уровне ассемблера. Затраты на верификацию, возникающие при внесении изменений, можно снизить, убедившись в том, что разработка разделена на подходящие по размеру компоненты ссылок на модели (model reference).

Этап верификации и валидации. При модельно-ориентированном проектировании модели и имитационное моделирование используются для ранней и непрерывной верификации и валидации разработок на протяжении всего процесса разработки.

Тестирование интеграции программного обеспечения и системы на этапе верификации и валидации (V&V) осуществляется в конце цикла разработки.

Основные цели верификации и валидации состоят в том, чтобы обеспечить соответствие разработки (верификация разработки) и реализации (верификация кода) заявленным требованиям. При использовании модельно-ориентированного проектирования, разработка представляется в виде модели и в завершение процесса демонстрируется, что модель соответствует требованиям, и код, сгенерированный из модели (реализация) также удовлетворяет требованиям, как показано на рис.1.41. Интеграция программного обеспечения обычно включает в себя интеграцию и валидацию программного обеспечения, сгенерированного из моделей и существующего программного обеспечения. Одним из методов тестирования интеграции программного обеспечения является тестирование с процессором в контуре (PIL). Системная интеграция в общем случае включает в себя использование программно-аппаратного моделирования (HIL) и тестирования на готовом к выпуску аппаратном обеспечении.

Верификация действительно нужных элементов. Первый шаг по внедрению верификации и валидации в проект с использованием модельно-ориентированного проектирования состоит в том, чтобы оценить, какие действия по верификации и валидации необходимы для одновременного удовлетворения

требований и временной экономии за счет модельно-ориентированного проектирования. Это определение должно опираться на цели проекта по времени, качеству и стоимости. Очевидная стратегия – выполнять ровно столько операций, сколько необходимо. Кроме того, чтобы снизить затраты на проект и время выхода на нормальный режим работы, нужно тщательно выбирать объемы автоматизации.



Рис.1.41. Процесс верификации и валидации в модельно-ориентированном проектировании.

Переключение внимания на проверку моделей. Проверка кода является важной частью традиционного процесса разработки. В модельно-ориентированном проектировании модели являются основным источником ошибок, ошибки трансляции при генерации кода минимальны, и автоматизированные технологии, такие как тестирование с программой в контуре (SIL), предоставляют возможности дополнительной проверки того, что код соответствует проектным требованиям. Больше внимание уделяется проверке моделей и формальной верификации и валидации. Как результат, время разработки тратится более продуктивно.

1.9.3. Гибкие методологии проектирования (agile-методы)

Agile – это семейство методологий разработки программного обеспечения, для которых характерны:

- итеративный процесс разработки;
- динамичное формирование требований и их реализация;
- тесная взаимосвязь, активное взаимодействие, как между абсолютно всеми участниками команды разработчиков, так и между командой и заказчиком.

Agile Manifesto [43] разработан и принят 11-13 февраля 2001 года на лыжном курорте The Lodge at Snowbird в горах Юты представителями таких методологий как Scrum, Crystal Clear, Extreme Programming (XP), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM).

Agile Manifesto содержит 4 основные идеи и 12 принципов.

Основные идеи:

Люди и взаимодействие важнее процессов и инструментов

Работающий продукт важнее исчерпывающей документации

Сотрудничество с заказчиком важнее согласования условий контракта

Готовность к изменениям важнее следования первоначальному плану

То есть, **не отрицая важности того, что справа,**

мы всё-таки больше ценим то, что слева.

Основные принципы:

- наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения;
- изменение требований приветствуется, даже на поздних стадиях разработки (Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества);
- работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев;
- на протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе;
- над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им;
- непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды;
- работающий продукт – основной показатель прогресса;
- инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно (Agile помогает наладить такой устойчивый процесс разработки);
- постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта;
- простота – искусство минимизации лишней работы – крайне необходима;
- самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд;
- команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Примечательно, что Agile Manifesto не содержит практических советов.

Agile Modeling (AM) – это набор понятий, принципов и приемов (практик), позволяющих быстро и просто выполнять моделирование и документирование в проектах разработки программного обеспечения (ПО).

AM описывает стиль моделирования, который позволит повысить качество и сократить сроки. AM не является технологическим процессом. Это не детальная инструкция по проектированию, он не содержит описаний, как строить

диаграммы на UML. АМ сосредоточен на эффективном моделировании и документировании. Он не охватывает программирование и тестирование, хотя в нем и говорится о проверке модели кодом и рассматривается тестируемость моделей. АМ также не включает вопросы управления проектом, развертывания и сопровождения системы.

АМ должен рассматриваться как дополнение к существующим методам, а не самостоятельная технология. Этот метод должен использоваться для повышения эффективности труда разработчиков, использующих процессы eXtreme Programming (XP), Dynamic Systems Development Method (DSDM), или RUP.

1.9.3.1. Методология экстремального программирования XP (eXtreme Programming)

Экстремальное программирование сформировалось в 1996 году. Экстремальное программирование было анонсировано Кентом Бекем в 1995 году в компании Chrysler. Экстремальное программирование основывается на идее адаптации изменений в программном проекте вместе с отказом от детального планирования. XP отходит от традиционного процесса создания программной системы и вместо единоразового планирования, анализа и проектирования с расчетом на долгосрочную перспективу, в XP все эти операции реализуются постепенно в ходе разработки, добиваясь тем самым значительного сокращения времени разработки и стоимости изменений в программе [44].

XP годится для создания программ коллективом от 2-х до 10 программистов, – это оптимальное число людей, когда они могут свободно общаться друг с другом.

Extreme Programming стремится, хоть и не явно, к специализации по выпуску кода высокого качества для оправдания постоянно меняющихся ожиданий пользователей.

В отличие от RAD, Extreme Programming не стремится к выпуску всеобъемлющего продукта за 2-3 месяца, а, напротив, стремится к выпуску множества небольших продуктов за больший промежуток времени. Extreme Programming придает большое значение написанию тестов перед написанием самого кода программного продукта, чтобы гарантировать, что код, будучи уже выпущен, не будет иметь одних и тех многократно возникающих ошибок; это также позволяет программистам писать код программ, опираясь на созданные тесты, тем самым повышая его эффективность, поскольку цели написания написания кода уже ясны. В отличие от многих других процессов, XP не заставляет программистов писать кучу отчетов и строить множество моделей.

Обычно XP характеризуют набором из 12 практик.

Практики – это действия, которые надо выполнять для достижения хорошего результата. Однако, следует помнить, что выполнение всех практик не гарантирует результат.

Практики XP:

- 1) планирование процесса.
- 2) частые релизы.

- 3) метафора системы.
- 4) простая архитектура.
- 5) тестирование.
- 6) рефакторинг.
- 7) парное программирование.
- 8) коллективное владение кодом.
- 9) частая интеграция.
- 10) 40-часовая рабочая неделя.
- 11) стандарты кодирования.
- 12) тесное взаимодействие с заказчиком.

В последнее время наметилась тенденция расширения набора практик, так как в XP недостаточно описывается процесс управления проектом.

Схема потоков работ в XP представлена на рис.1.42.



Рис.1.42. Схема потоков работ в XP.

Преимуществами XP являются:

- простота в использовании, адаптации, обучении;
- устойчивость к внешним факторам, таким как текучесть кадров, нехватка денег, внезапное завершение финансирования;
- учёт изменчивости требований и кардинального пересмотра всей системы;
- равномерная загруженность всех членов коллектива;
- быстрое включение в работу новичков с минимальным уровнем риска;
- эффективный контроль работоспособности разрабатываемых систем;
- увеличение производительности разработчиков;

- предоставление дополнительных благ членам коллектива;
- естественный профотбор членов команды;
- низкая степень бюрократизма;
- широкая известность и популярность.

ХР имеет свои проблемы. Так устранение формальных письменных требований к разработке делает его уязвимым к постоянно возникающим изменениям программы. Избыточность парного программирования может иногда замедлять процесс разработки программного обеспечения. Отсутствие детальных требований и непродуманное планирование приводят к тому, что система может иметь в целом негодный дизайн. Пересмотр кода – это очень хорошо, но это только попытка устранить проблемы, возникшие на фундаментальном уровне разработки дизайна.

Экстремальное программирование эффективно применяется:

- в проектах, над которыми может работать от двух до десяти программистов;
- в проектах с постоянно изменяющимися требованиями;
- в проектах с высокой степенью риска;
- при заранее заданных сроках сдачи проекта;

1.9.3.2. Методология управления проектами – Scrum

Методика Scrum представляет собой альтернативный подход к разработке проекта, в противоположность стандартной модели заполнения многостраничного ТЗ. Адаптивная по своему характеру, методика позволяет сделать процесс разработки более гибким, контролируемым, а значит эффективным.

Впервые новый подход к разработке проекта был описан Икудзиро Нонака и Хиротака Такэути в их совместной статье The New Product Development Game, опубликованной в Гарвардском Деловом Обзоре зимой 1986 года. В статье было сделано важное наблюдение, что лучшие результаты приносят проекты, над которыми работают небольшие, кросс-функциональные команды.

Вскоре идея была подхвачена и продолжена другими авторами. В начале 1990-х годов новый подход подробно описали, задокументировали и успешно применили на практике Кен Швабер и Джеф Сазерленд [45]. Методика получила название – Scrum.

Scrum – это методология управления разработкой информационных систем, в которой делается жесткий акцент на качественном контроле процессом разработки. Помимо управления проектами по разработке программного обеспечения, методика используется командами поддержки программного обеспечения, а также как подход управления разработкой и сопровождением программ.

Следуя методике Scrum, весь процесс разработки делится на небольшие временные промежутки, которые называют спринтами (Sprint), рис.1.43.

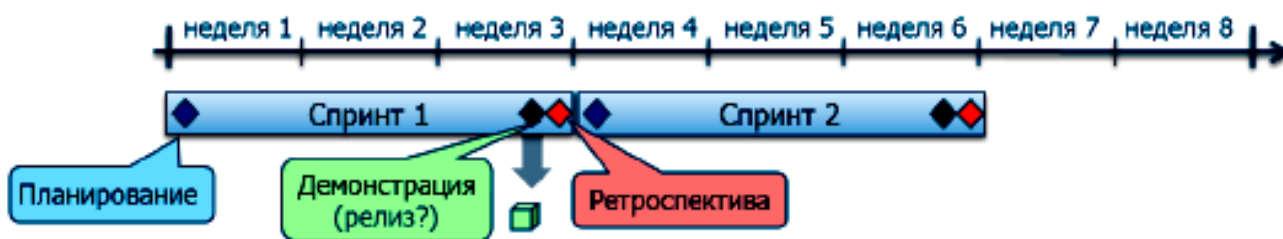


Рис.1.43. Схема процесса Scrum [46].

В течении спринта выполняется работа над продуктом. По его окончанию должна быть получена новая рабочая версия продукта. Спринт всегда ограничен по времени (1-4 недели) и имеет одинаковую продолжительность на протяжении все жизни продукта.

Перед началом каждого спринта производится Sprint Planning, на котором производится оценка содержимого Product Backlog и формирование Sprint Backlog, который содержит задачи (Story, Bugs, Tasks), которые должны быть выполнены в текущем спринте. Каждый спринт должен иметь цель, которая является мотивирующим фактором и достигается с помощью выполнения задач из Sprint Backlog.

Каждый день производится Daily Scrum, на котором каждый член команды отвечает на вопросы.

По окончании Sprint'a производятся Sprint Review и Sprint Retrospective, задача которых оценить эффективность (производительность) команды в прошедшем Sprint'e, спрогнозировать ожидаемую эффективность (производительность) в следующем спринте, выявлении имеющихся проблем, оценки вероятности завершения всех необходимых работ по продукту и другое.

Схематическое изображение процесса приведено на рис.1.44:

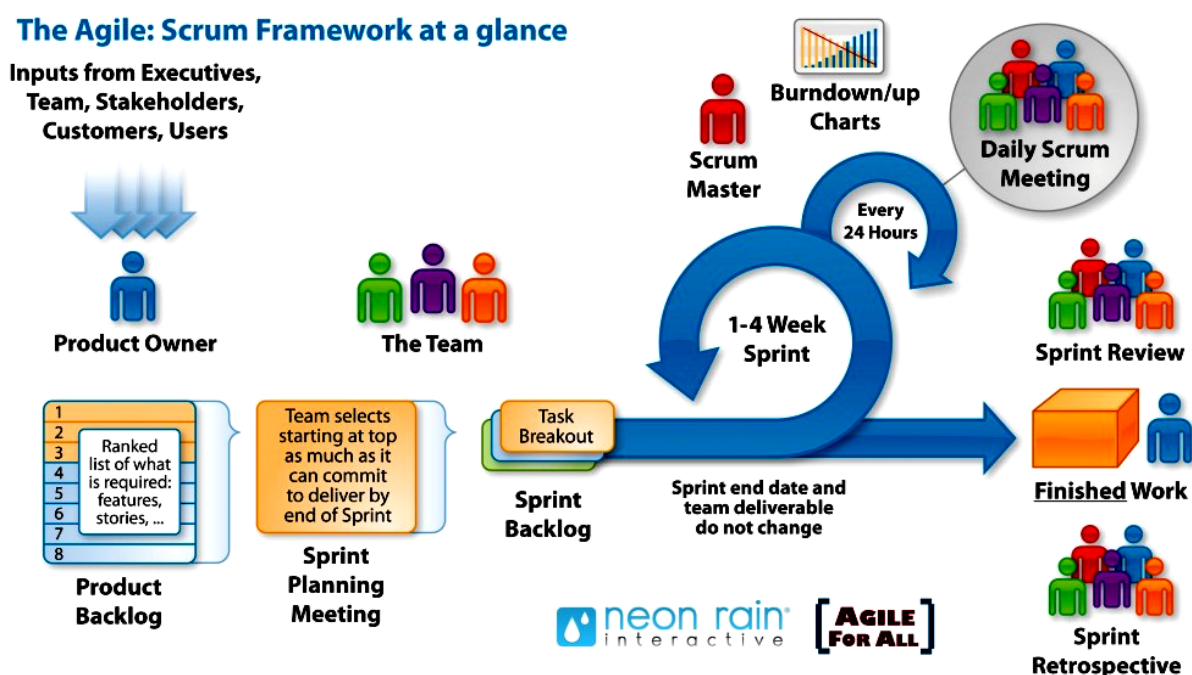


Рис.1.44. Схема процесса Scrum.

Дробление задач на спринты, позволяет делать процесс разработки более предсказуемым и гибким.

Еще два термина, которые важны в методике Scrum – резерв проекта и резерв спринта. Под резервом проекта понимается упорядоченный по степени важности список требований к функциональности, в резерве спринта содержится функциональность, выбранная владельцем проекта из резерва проекта. Соответственно, процессом разработки проекта занимается Scrum-команда, участники которой наделяются строго очерченными «ролями».

В классическом Scrum существует 3 базовых роли:

- Product owner (PO) – является связующим звеном между командой разработки и заказчиком. Задача PO – максимальное увеличение ценности разрабатываемого продукта и работы команды. Одним из основных инструментов PO является Product Backlog. Product Backlog содержит необходимые для выполнения рабочие задачи (такие как Story, Bug, Task и др.), отсортированные в порядке приоритета (срочности);

- Scrum master (SM) является «служашим лидером» (англ. servant-leader). Задача Scrum Master – помочь команде максимизировать ее эффективность посредством устранения препятствий, помощи, обучении и мотивации команде, помощи PO;

- команда разработки (Development team, DT) состоит из специалистов, производящих непосредственную работу над производимым продуктом. Согласно The Scrum Guide (документу, являющимся официальным описанием Scrum от его авторов), DT должны обладать следующими качествами и характеристиками:

- быть самоорганизующейся. Никто (включая SM и PO) не может указывать команде, как им преобразовать Product Backlog в работающий продукт;

- быть многофункциональной, обладать всеми необходимыми навыками для выпуска работающего продукта;

- за выполняемую работу отвечает вся команда, а не индивидуальные члены команды.

Рекомендуемый размер команды – 7 (плюс-минус 2) человека. Согласно идеологам Scrum, команды большего размера требуют слишком больших ресурсов на коммуникации, в то время как команды меньшего размера повышают риски (за счет возможного отсутствия требуемых навыков) и уменьшают размер работы, который команда может выполнить в единицу времени.

1.9.3.3. Методология разработки динамических систем – DSDM (Dynamic Systems Development Method)

DSDM (Dynamic Systems Development Method) [47] появился в Великобритании в 1994 г. Его основателем стал консорциум из 17 английских компаний, которые хотели работать с использованием RAD и принципов итеративной разработки. Сейчас число его членов перевалило за тысячу, причем многие из них находятся за пределами Соединенного королевства.

Все начинается с изучения осуществимости программы и области ее применения. Далее процесс делится на три взаимосвязанных цикла: цикл

функциональной модели отвечает за создание аналитической документации и прототипов, цикл проектирования и конструирования – за приведение системы в рабочее состояние, и наконец, последний цикл – цикл реализации – обеспечивает развертывание программной системы.

Методология DSDM основана на подходе RAD (Rapid Application Development) и включает в себя три стадии, рис.1.45:

- предпроектная стадия, на которой авторизуется реализация проекта, определяются финансовые параметры и команда;
- жизненный цикл проекта представляет собой реализации проекта и включает в себя пять этапов(первые две фактически объединяются):
 - определение реализуемости;
 - экономическое обоснование;
 - создание функциональной модели;
 - проектирование и разработка;
 - реализация;
- постпроектная стадия обеспечивает качественную эксплуатацию системы.



Рис.1.45. Схема процесса методологии DSDM.

Базовые принципы, на которых строится DSDM, это активное взаимодействие с пользователями, частые выпуски версий, самостоятельность разработчиков в принятии решений и тестирование в течение всего цикла работ. Как и большинство других гибких методологий разработки мобильных приложений, DSDM использует короткие итерации, продолжительностью от двух до шести недель каждая. Особый упор делается на высоком качестве работы и адаптируемости к изменениям в требованиях.

1.9.3.4. Методология разработки, управляемой функциональностью FDD (Feature driven development)

Методология (кратко именуемая FDD) была разработана Джеффом Де Люка (Jeff De Luca) и Питером Коадом (Peter Coad). Как и остальные адаптивные методологии, она делает основной упор на коротких итерациях, каждая из которых служит для проработки определенной части функциональности системы. Согласно FDD, одна итерация длится две недели.

FDD насчитывает пять процессов. Первые три из них относятся к началу проекта.

- разработка общей модели
- составление списка требуемых свойств системы
- планирование работы над каждым свойством
- проектирование каждого свойства
- конструирование каждого свойства

Последние два шага необходимо делать во время каждой итерации. При этом каждый процесс разбивается на задачи и имеет критерии верификации.

Все разработчики делятся на два вида: «class owners» (владельцы классов) и «chief programmers» (старшие программисты). Старшие программисты – это наиболее опытные разработчики. Именно им поручается разработка конкретных свойств системы. Однако они не занимаются этим самостоятельно: старший программист определяет, какие классы заняты в реализации данного конкретного свойства, после чего собирает команду из владельцев необходимых классов, которая и будет заниматься разработкой. Сам он действует как координатор, главный проектировщик и руководитель, а на долю владельцев классов остается, по большей части, непосредственное кодирование [48].

1.9.3.5. Методология оптимизации потока работ Kanban (Kanban Development)

Методика разработки программного обеспечения Kanban придумана Дэвидом Андерсеном (David Anderson). Многие из этих практик и подходов использовались разными Agile командами, прежде чем были описаны Дэвидом как единое целое.

Канбан (Kanban «Kan» – видимый, визуальный; «ban» – карточка или доска), японский термин, известный благодаря производственной системе Тойота, основными принципами которой являются: бережливое производство, постоянное развитие, ориентация на клиента и т.п. [49]

Эти принципы позволяют:

- не тратить время на оценку трудоемкости, поскольку нет горизонта планирования - любая работа должна быть выполнена максимально быстро;
- предотвращать перепроизводство за счет визуализации и ограничения незавершенной работы (WIP);
- организовать автономный (самоподдерживающийся) процесс, не требующий микроуправления;
- визуализировать весь процесс, выявить узкие места и мотивировать к его оптимизации.

Канбан разработка крутится вокруг визуальной доски, которая используется для управления незаконченной работой, рис.1.46.



Рис.1.46. Визуальная доска Канбан.

Основной идеей является то, что истории стартуют с левого края доски и проходят всю доску слева направо, попутно проходя через те фазы разработки, без которых эти истории нельзя будет назвать завершенными.

Завершенные истории, готовые к запуску в продакшн окружении, копятся на правом краю доски. И, поскольку это Канбан доска и необходимо ограничивать количество незавершенной работы, то ограничивают количество пользовательских историй, одновременно находящихся на доске.

Числа, написанные снизу каждой колонки, представляют собой максимальное количество историй, допустимое в данной конкретной колонке.

Канбан доска – слева направо:

Goals – цели. С самого левого края Канбан доски расположена колонка целей. Здесь находятся глобальные цели – крупные вещи, к которым стремятся и под которые поднастраивается программное обеспечение. Оставляя цели на самом краю доски, все члены команды фокусируются на одном: на достижении

этих целей. Эта идея была взята у Арло Белши, который опытным путем определил, что размещение целей на левой стороне доски Канбан существенно снижает объем «мусора» на доске. Под «мусором» здесь понимается частое изменение приоритетов отдельных Канбан карточек.

Stories queue – очередь историй. В следующей колонке хранится очередь историй, готовых к запуску. Это именно очередь, поскольку истории в ней упорядочены по времени поступления. Первая пришедшая история будет начата раньше всех, при этом она переместится в другие колонки, а остальные истории сместятся вверх по очереди.

Стадии разработки истории. Справа от очереди историй расположены колонки со стадиями разработки, через которые должна пройти история, чтобы считаться выполненной.

Первая колонка часто используется для стадии прорабатывания истории. Следующая колонка часто используется для собственно разработки, а следующая за ней – для тестирования. Эти колонки не фиксированы.

Каждая стадия разделена на две части: верх используется для историй, находящихся в разработке, а низ – для буфера историй. Когда работа над определенной стадией определенной истории заканчивается, историю перемещают из блока «в процессе» в буфер, где она будет дожидаться перевода на следующую стадию. У последней стадии нет буфера, поскольку эта стадия истории завершается при завершении истории. Когда ограничивается количество невыполненной работы, устанавливаются рамки для общего количества историй в стадии, которое включает в себя как текущие истории, так и истории в буфере.

Done! – Сделано! Когда история завершается, она готова к поставке на продакшн.

Измерение параметров процесса, построенного на базе Kanban, осуществляется с использованием простых метрик:

- *среднее время цикла* – характеризует производительность команды, то есть время от начала работы над некоторой функциональностью (пожеланием) до момента ее полной готовности к передачи заказчику.

- *диаграмма потока* – отражает динамику реализации доработок и исправления ошибок, возникающие проблемы, узкие места и другие негативные сигналы.

Канбан можно описать тремя основными правилами:

1. Визуализация производства:

- разделить работу на задачи, каждую задачу написать на карточке и поместить на стену или доску.

- использовать названные столбцы, чтобы показать положение задачи в производстве.

2. Ограничение WIP (work in progress или работу, выполняемую одновременно) на каждом этапе производства.

3. Измерение времени цикла (среднее время на выполнение одной задачи) и оптимизировать постоянно процесс, чтобы уменьшить это время.

1.10. Средства проектирования информационных систем

Для конкретных классов технологии проектирования свойственно применение соответствующих средств проектирования информационных систем.

1.10.1. Традиционные средства проектирования

Использование технологии канонического проектирования предполагает применение различных средств на традиционных носителях, к которым принято относить:

- нормативно-правовые документы (положения о структурном подразделении, должностных инструкций и т.п.);
- нормативно-технические документы (стандарты, руководящие документы и т.п.);
- системы классификации и кодирования информации;
- системы документации;
- модели входных и выходных потоков информации и методики их анализа;
- др.

1.10.2. Средства автоматизации проектирования (CASE-средства проектирования)

При автоматизированном проектировании наряду с вышеназванными средствами могут быть использованы разнообразные программные средства, которые подразделяют на четыре группы:

- операционные средства (алгоритмические языки, макрогенераторы, генераторы программ типовых операций обработки данных, утилиты, библиотеки стандартных подпрограмм и классов объектов и др.);
- прикладные программные средства общего назначения (СУБД, текстовые и графические редакторы, табличные процессоры, методо-ориентированные пакеты прикладных программ, оболочки экспертных систем, интегрированные пакеты прикладных программ);
- функциональные средства проектирования (функциональные пакеты прикладных программ, типовые проекты и проектные решения);
- средства автоматизации проектирования (CASE-средства).

Проектирование ИС с применением компьютерной поддержки, называется CASE – технологии проектирования. CASE – технологии применяются не только для автоматизации проектирования ИС, но и для разработки моделей бизнес-процессов при проведении бизнес-анализа. CASE – технологии применяются в ситуациях, когда проблематика предметной области отличается большой сложностью.

Можно выделить следующие *основные принципы* создания ИС на основе CASE – технологий:

- принцип всесторонней компьютерной поддержки проектирования;

- принцип модельного подхода. CASE – система может поддерживать методологию функционально-ориентированного или объектно-ориентированного подхода;

- принцип иерархического представления модели предметной области. Данный принцип выражается в возможности последовательной детализации (декомпозиции) описания системы в соответствии с нисходящим подходом проектирования;

- принцип наглядности представления модели – означает наличие в составе CASE-технологий визуальных средств проектирования. Система графических изображения и правила, предназначенные для описания структуры системы, элементов данных и т.д., называются нотацией CASE-средства;

- принцип декомпозиции процесса проектирования ИС с применением CASE-технологий на стадии и этапы.

Большинство современных CASE-средств поддерживает методологии структурного и/или объектно-ориентированного анализа и проектирования информационных систем. Выбор того или иного подхода (парадигмы) подразумевает следование ему и на стадии кодирования (согласно принципу концептуальной общности). Их отличие друг от друга заключается в выборе способа декомпозиции системы (задачи). Если за основу принимается функциональная (алгоритмическая) декомпозиция, то речь идет о структурном подходе, если объектная – об объектно-ориентированном.

Наиболее распространенными вариантами классификации CASE-средств являются разделение по типам и категориям.

Классификация по уровню проектирования в жизненном цикле создания ИС:

- *средства верхнего уровня (Upper CASE)* – анализ предметной области, определение места ИС в контуре бизнес-системы;

- *средства среднего уровня (Middle CASE)* – разработка архитектуры ИС, создание проектных спецификаций;

- *средства нижнего уровня (Lower CASE)* – поддержка разработки программного обеспечения.

Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ и включает следующие типы:

- *средства анализа (соответствуют Upper CASE)*, предназначенные для построения и анализа моделей предметной области – (CA ERwin Process Modeler (бывший BPWin, затем AllFusion Process Modeler), Design/IDEF, ARIS, ORACLE Designer);

- *средства анализа и проектирования (соответствуют Middle CASE)*, поддерживающие наиболее распространенные методологии проектирования, которые используются для создания проектных спецификаций, в частности проектных компонентов интерфейсов системы, архитектуры системы, алгоритмов и структур данных (Vantage Team Builder, Designer/2000, Silverran, PRO-4, CASE-аналитик, ARIS);

– *средства проектирования баз данных (соответствуют Middle CASE)*, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL – Structured Query Language – структурированном языке запросов) для наиболее распространенных СУБД: ER-win, S-Designer/2000, Data Base Designer, ARIS Tooleset;

– *средства разработки приложений (соответствуют Lower CASE)* - средства 4GL – Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000, (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др. и генераторы кода, входящие в состав Vantage Team Builder, PRO-IV и частично – в Silverrun;

– *средства реинжиниринга*, обеспечивающие анализ программных кодов и схем БД и формирование на их основе различных моделей и проектных спецификаций: ER-win, Vantage.Team Builder, Silverran, PRO-4, ARIS Tooleset, Designer/2000 и т.д. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программы на языке C++: Rational Rose, Object Team.

Вспомогательные типы включают:

– средства планирования и управления проектом (Microsoft Project, SE Companion);

– средства конфигурационного управления (PVCS);

– средства тестирования (Quality Works);

– средства документирования (SoDa).

Классификация по степени интегрированности выделяет:

– *локальные CASE-средства* – применяются для анализа системы и разработки автоматизированных рабочих мест, поддерживают 1, 2 типа моделей и методов (CASE-аналитик, Design/IDEF);

– *малые интегрированные CASE-средства*, используются для создания небольших ИС, поддерживают несколько типов моделей и методов (ER-win, BP-win, , Silverran);

– *средние интегрированные CASE-средства* – поддерживают от 4-15 моделей и методов. В этой категории Rational Rose, Designer/2000;

– *крупные интегрированные CASE-средства*, поддерживаются свыше 15 типов моделей и методов (семейство программных продуктов ARIS).

На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

– Vantage Team Builder (Westmount I-CASE);

– Designer/2000;

– Silverrun;

– ERwin+BPwin;

– Design/IDEF;

– S-Designor;

– CASE.Аналитик.

Раздел 2

Инструктивно-методические указания по подготовке к лекционным занятиям

Подготовка к лекционным занятиям заключается в ознакомлении студента с материалом по блокам вопросов, в соответствии с темой предстоящей лекции согласно учебной программы дисциплины. Тема занятия объявляется ведущим преподавателем в конце предшествующей лекции.

Студент составляет краткий конспект по тематическому блоку.

После рассмотрения соответствующей темы на лекции, студент обязан провести углубленное изучение темы, по представленным вопросам для самостоятельного изучения и выполнения.

Лекция 1. Введение в проектирование информационных систем. Основные понятия и определения. (2 часа)

Подготовка по разделу 1 настоящего пособия (п.1.1, 1.2, 1.3).

Литература и информационные источники: [1 – 10]

Вопросы для самоконтроля

1. Сходства и отличия проектирования сложной системы в целом от проектирования ИС.
2. Конструирование и проектирование: сходство и отличия.
3. Важен ли опыт проектировщика в типовом проектировании, и в какой мере?
4. Как может быть связано требование к развитию ИС с проектированием?
5. Может ли локальное проектирование быть заменой реинжинирингу (перепроектированию)?
6. Привести пример, когда «отсутствие лишних функций» является существенным требованием к ИС.

Вопросы для самостоятельного изучения (выполнения)

1. Автоматизация документирования проекта.
2. Проблемы, связанные с проектированием ИС.

Лекция 2. Стандарты проектирования информационных систем (2 часа)

Подготовка по разделу 1 настоящего пособия (п.1.4)

Литература и информационные источники: [8,10 – 17].

Вопросы для самоконтроля

Сравнительный анализ комплексов стандартов: ГОСТ Р ИСО/МЭК 12207, ГОСТ Р ИСО/МЭК 15288-2005, ГОСТ 34.601-90.

Вопросы для самостоятельного изучения (выполнения)

Систематизировать комплекс государственных и международных стандартов, регламентирующих процессы разработки ИС, заполнив таблицу 2.1.

Таблица 2.1. Стандарты по разработке информационных систем

| Обозначение стандарта | Наименование стандарта |
|---|------------------------|
| Российские (стандарты СССР) | |
| ... | |
| Российские, идентичные международным | |
| ... | |

Лекция 3. Модели жизненного цикла информационных систем (2 часа)

Подготовка по разделу 1 настоящего пособия (п.1.4)

Литература и информационные источники: [8,10 – 22].

Вопросы для самоконтроля

1. Жизненный цикл проектирования как локальный жизненный цикл системы в целом.
2. Принципиальные отличия ЖЦ ИС.
3. Методы выбора модели ЖЦ ИС.

Вопросы для самостоятельного изучения (выполнения)

Проанализировать модели жизненного цикла информационных систем, являющиеся практиками компаний – лидеров в области проектирования информационных систем и разработки программного обеспечения, заполнив таблицу 2.2.

Таблица 2.2. Модели жизненного цикла «фирменных» методологий проектирования

| Компания | Модель жизненного цикла (краткое описание) | Инструменты |
|----------|--|-------------|
| ... | | |

Лекция 4,5. Методологии и инструментальные средства структурного подхода (анализа) к проектированию ИС.

Подготовка по разделу 1 настоящего пособия (п.1.7, п.п. 1.8.1, п.1.9)

Литература и информационные источники: [18,23 – 37].

Вопросы для самоконтроля

1. Пояснить смысл и содержание структурного и функционального проектирования.
2. Перечислить стандартизированные методологии структурного и функционального проектирования.

Вопросы для самостоятельного изучения (выполнения)

Систематизировать методологии проектирования информационных систем, заполнив таблицу 2.3.

Таблица 2.3. Методологии проектирования информационных систем

| Обозначение методологии | Описание | Инструменты |
|--|----------|-------------|
| Методологии структурного подхода (анализа) | | |
| ... | | |

Лекция 6. Методологии и инструментальные средства модельно-ориентированного подхода (анализа) к проектированию ИС.

Подготовка по разделу 1 настоящего пособия (п.п.1.8.2, п.1.9)

Литература и информационные источники: [10,21, 38 – 42].

Вопросы для самоконтроля

1. Пояснить смысл и содержание модельно-ориентированного проектирования.
2. Перечислить стандартизированные и фирменные методологии модельно-ориентированного проектирования.

Вопросы для самостоятельного изучения

Продолжить заполнение Таблицы 2.3.

| Обозначение методологии | Описание | Инструменты |
|---|----------|-------------|
| Методологии модельно-ориентированного подхода (анализа) | | |
| ... | | |

Лекция 7. Гибкие методологии (agile-методы) и инструментальные средства проектирования ИС.

Подготовка по разделу 1 настоящего пособия (п.п.1.8.3, п.1.9)

Литература и информационные источники: [43 – 49].

Вопросы для самоконтроля

1. Пояснить смысл и содержание гибких методологий проектирования.
2. Перечислить фирменные гибкие методологии проектирования.

Вопросы для самостоятельного изучения

Продолжить заполнение Таблицы 2.3.

| Обозначение методологии | Описание | Инструменты |
|-----------------------------------|----------|-------------|
| Гибкие методологии (agile-методы) | | |
| ... | | |

Раздел 3

Инструктивно-методические указания к практическим занятиям

Практическое занятие №1

Модели жизненного цикла и методы планирования и управления проектами

Цель:

- изучить модели жизненного цикла информационных систем и стандарты проектирования информационных систем;
- ознакомиться с методами планирования и управления проектами;
- построить сетевую диаграмму проекта.

Время: 2 часа

Краткие теоретические сведения

1. Модели жизненного цикла информационных систем и стандарты проектирования

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде *жизненного цикла* (ЖЦ) ИС. Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла – структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

В настоящее время известны и используются следующие модели жизненного цикла (см. п.1.6.):

- каскадная модель предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе;
- поэтапная модель с промежуточным контролем. Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки;
- спиральная модель. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка. Особое внимание уделяется

начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).

Существует целый ряд стандартов, регламентирующих ЖЦ информационных систем, а в некоторых случаях и процессы разработки.

В настоящее время существует несколько комплексов стандартов, которые регламентируют процессы проектирования и разработки информационных систем. Часть этих комплексов, а также основные (с точки зрения проектирования) нормативные документы, входящие в них, приведены в табл.3.1.

Таблица 3.1. Комплексы нормативных документов на разработку информационных систем

| Обозначение | Наименование |
|--|---|
| <i>Стандарты ISO/IEC (ISO/МЭК) в области разработки и документирования программных средств</i> | |
| ГОСТ Р ИСО/МЭК 12207-02 | Информационная технология. Процессы жизненного цикла программных средств |
| ГОСТ Р ИСО/МЭК 15271-02 | Информационная технология. Руководство по ИСО/МЭК 12207 (процессы жизненного цикла программных средств) |
| ГОСТ Р ИСО/МЭК 9126-93 | Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению |
| ГОСТ Р ИСО/МЭК 12119-94 | Информационная технология. Пакеты программ. Требования к качеству и тестирование |
| <i>Комплекс нормативных документов на автоматизированные системы</i> | |
| ГОСТ 34.003-90 | Автоматизированные системы. Термины и определения |
| ГОСТ 34.201-89 | Виды, комплектность и обозначение документов при создании автоматизированных систем |
| ГОСТ 34.601-90 | Автоматизированные системы. Стадии создания |
| ГОСТ 34.602-89 | Техническое задание на создание автоматизированной системы |
| РД 50-698-50 | Автоматизированные системы. Требования к содержанию документов |
| РД 50-34.126-92 | Рекомендации. Правила проведения работ при создании автоматизированных систем |
| <i>Комплекс стандартов Единой системы программной документации (ЕСПД)</i> | |
| ГОСТ 19.101-77 | Виды программ и программных документов |
| ГОСТ 19.102-77 | Стадии разработки |
| ГОСТ 19.105-78 | Общие требования к программным продуктам |
| ГОСТ 19.201-78 | Техническое задание. Требования к содержанию и оформлению |
| ГОСТ 19.701-90 (ИСО/МЭК 5807-85) | Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения |

Из приведенных в табл.3.1 комплексов стандарты 19-й и 34-й серий являются наиболее устаревшими, но в то же время и наиболее полными.

Международный стандарт ИСО/МЭК 12207-02 содержит минимум ограничений и конкретных рекомендаций, что позволяет взять его за основу при разработке ведомственных нормативных документов или фирменных методик.

2. Процессы планирования и управления проектом в жизненном цикле

Из курса «Управление IT-проектами» известно, что в самом общем виде методология проектного менеджмента определяет и формализует процедуры, методы и инструменты реализации пяти групп управленческих процессов (согласно стандарту PMBOK Guide – Руководству к своду знаний по управлению проектами [50]):

- инициации проекта;
- планирования;
- организации исполнения;
- контроля исполнения;
- завершения проекта.

Инициация проекта – процесс управления проектом, результатом которого является авторизация и санкционирование начала проекта или очередной фазы его жизненного цикла.

Инициация проекта может включать следующие процедуры:

- разработка концепции проекта:
 - анализ проблемы и потребности в проекте;
 - сбор исходных данных;
 - определение целей и задач проекта;
 - рассмотрение альтернативных вариантов проекта.
- рассмотрение и утверждение концепции.
- принятие решения о начале проекта:
 - определение и назначение менеджера проекта;
 - принятие решения об обеспечении ресурсами выполнения первой

фазы проекта.

Планирование проекта – непрерывный процесс, направленный на определение и согласование наилучшего способа действий для достижения поставленных целей проекта с учетом всех факторов его реализации.

Основным результатом этого этапа является **план** проекта. Однако, процесс планирования не завершается разработкой и утверждением первоначального плана проекта. В ходе осуществления проекта могут происходить изменения как внутри проекта, так и во внешнем окружении, которые требуют уточнения планов, а часто значительного перепланирования. Поэтому процессы планирования могут осуществляться на протяжении всего жизненного цикла проекта, начиная с предварительного укрупненного плана в составе концепции проекта, и заканчивая детальным планом работ завершающей фазы проекта.

Планирование предметной области проекта включает следующие задачи и процедуры:

- анализ текущего состояния и уточнение целей и результатов проекта;
- уточнение основных характеристик проекта;
- подтверждение и уточнение критериев успеха и неудач проекта;

- анализ и корректировку ограничений и допущений, принятых на стадии инициации проекта;
- выбор критериев оценки промежуточных и окончательных результатов создания проекта;
- построение структурной декомпозиции предметной области проекта;
- планирование времени проекта.

Планирование времени проекта. Согласованная работа всех участников проекта организуется на основе календарных планов или расписаний работ проекта, основными параметрами которых являются: сроки выполнения, ключевые даты, продолжительности работ и др.

Календарными планами называют проектно-технологические документы, устанавливающие полный перечень работ проекта, их взаимосвязь, последовательность и сроки выполнения, продолжительности, а также исполнителей и ресурсы, необходимые для выполнения работ проекта.

Планирование проекта по временным параметрам заключается в составлении различных календарных планов (расписаний работ), удовлетворяющих всем требованиям и ограничениям проекта и его частей. Календарные планы составляются на весь жизненный цикл проекта и его этапы, для различных уровней управления и участников проекта,

Календарное планирование проекта состоит из следующих этапов:

Этап 1. Составление структурной декомпозиции работ

Структурная декомпозиция работ (СДР) – графическое изображение иерархической структуры всех работ проекта, рис.3.1.

Структурная декомпозиция работ проекта (Work Breakdown Structure - WBS) – разбиение проекта на составные части (элементы, модули, работы и др.), необходимые и достаточные для его эффективного планирования и контроля.

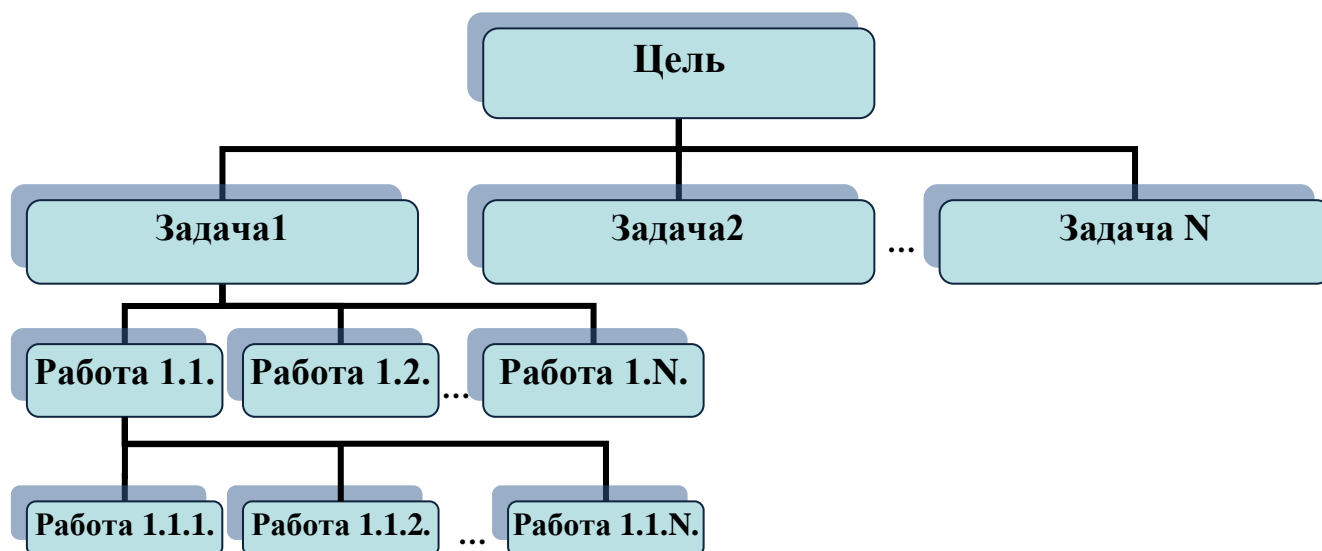


Рис.3.1. Структурная декомпозиция работ (СДР).

СДР является центральным инструментом определения работ, которые должны выполняться в рамках проекта. Описание работ (пакетов работ) должно включать:

- содержание работ;
- предполагаемые результаты;
- концептуальные границы интегрированного планирования и управления;
- последовательные измерения и оценки степени выполнения проекта.

При построении иерархии структуры работ (ИСР) необходимо соблюдать следующие *правила*:

- работы нижнего уровня являются способом достижения работ верхнего уровня;
- у каждой родительской работы может иметься несколько дочерних работ, достижение которых автоматически обеспечивает достижение родительской работы;
- у каждой дочерней работы может быть только одна родительская работа;
- декомпозиция родительской работы на дочерние производится по одному критерию, в качестве которого могут выступать: компоненты результатов и продуктов проекта, этапы жизненного цикла проекта, ресурсы и функциональные виды деятельности, а также элементы организационной структуры;
- на одном уровне дочерние работы, декомпозирующие родительскую должны быть равнозначны. В качестве критерия равнозначности могут выступать: объем и время выполнения работ, пр.;
- при построении иерархической структуры работ на различных уровнях можно и следует применять различные критерии декомпозиции.
- последовательность критериев декомпозиции работ следует выбирать таким образом, чтобы как можно большая часть зависимостей и взаимодействий между работами оказалась на самых нижних уровнях ИСР. На верхних уровнях работы должны быть автономны.
- декомпозиция работ прекращается тогда, когда работы нижнего уровня удовлетворяют следующим условиям:
 - работы ясны и понятны менеджеру и участникам проекта (являются элементарными);
 - понятен конечный результат работы и способы его достижения;
 - временные характеристики и ответственность за выполнение работ могут быть однозначно определены.

Этап 2. Определение списка работ проекта на основе структурной декомпозиции проекта.

Этап 3. Определение последовательности выполнения работ и их взаимосвязей с помощью организационно-технологических моделей. Уточнение временных ограничений.

Этап 4. Определение продолжительности работ.

На данном шаге, необходимо указать продолжительность выполнения каждой работы по проекту. Эта продолжительность может быть рассчитана, исходя из нормативов, может быть указана, исходя из личного опыта.

Часто невозможно однозначно определить продолжительность той или иной работы. В таком случае можно использовать методы PERT [51].

Метод PERT (Program Evaluation and Review Technique) – метод событийного сетевого анализа, используемый для определения длительности проекта при наличии неопределенности в оценке продолжительностей индивидуальных операций.

PERT основан на методе критического пути (см. дальше), длительность операций в котором рассчитывается как взвешенная средняя оптимистического, пессимистического и ожидаемого прогнозов. PERT рассчитывает стандартное отклонение даты завершения от длительности критического пути.

Продолжительность работы рассчитывается в данном случае как средневзвешенная средняя оптимистического, пессимистического и ожидаемого прогнозов.

Например, предстоит работа «поиск практического материала для написания дипломного проекта». Однозначно определить ее продолжительность затруднительно. В таком случае можно указать оптимистический прогноз продолжительности (минимально возможная продолжительность) выполнения данной работы, например – 3 дня. Далее указывается пессимистический прогноз продолжительности (максимально возможная продолжительность), например: 30 дней. Также необходимо указать наиболее вероятную продолжительность данной работы, например: 10 дней, табл.3.2.

Следующим шагом является расчет вероятности каждого из рассчитанных прогнозов, как показано ниже в таблице. Вероятность, как правило, указывается в процентах или в долях. 100% (или 1) - это означает однозначное осуществление прогноза, 0% (или 0) – означает, что прогноз не сбудется ни при каких обстоятельствах.

Таблица 3.2. Показатели продолжительности работы

| | оптимистический | наиболее вероятный | пессимистический |
|------------------------------------|------------------------|---------------------------|-------------------------|
| Прогнозное значение | 3д | 10д | 30д |
| Вероятность осуществления прогноза | 0,2 | 0,6 | 0,2 |
| Взвешенное значение | 0,6д | 6д | 6д |
| Сумма взвешенных значений: | 12,6д | | |

***Замечание:** значения вероятности могут быть рассчитаны известными математическими методами, либо определены экспертным образом.*

В завершении расчета продолжительности выполнения работы методом PERT необходимо перемножить значение каждого прогноза на его вероятность и полученные величины сложить.

Таким образом, получится средневзвешенное значение продолжительности выполнения работы. В рассматриваемом случае, как показано в таблице, расчет был следующим:

$$3д \cdot 0,2 + 10д \cdot 0,6 + 30д \cdot 0,2 = 0,6д + 6д + 6д = 12,6д.$$

Это означает, что примерная продолжительность работы «поиск практического материала для написания дипломного проекта» составляет около 13 дней.

Этап 5 Составление сетевой диаграммы проекта.

Сетевая диаграмма – графическое отображение работ проекта и зависимостей между ними, рис.3.2.

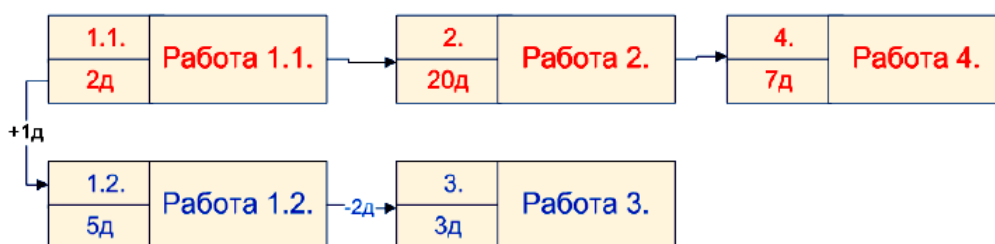


Рис.3.2. Сетевая диаграмма.

Цель методов сетевого планирования – сократить до минимума продолжительность проекта.

Как правило, сетевая диаграмма представляется в виде графа, в котором вершинами являются проектные работы, а взаимосвязь и последовательность работ отображается соединительными линиями, как показано на рисунке.

Работа в сетевой диаграмме отображается в виде прямоугольника, в котором содержится информация о работе: код в сетевой диаграмме работ (например:1.1.), наименование и продолжительность работы.

Стрелками, обозначается последовательность и взаимосвязь работ. Взаимосвязи также могут характеризоваться временными показателями. Например, работы 1.1. и 1.2. связаны соединительной стрелкой со значением «+1д», Это означает, что работа 1.2. должна начаться через день после того как начнется работа 1.1. А на стрелке, соединяющей работы 1.2. и 3. стоит значение «-2д». Это означает, что работа 3 должна начаться за два дня до окончания работы 1.2. Если на стрелке нет дополнительной информации, как например на стрелке, соединяющей работы 1.1. и 2., то это означает, что работа 2 начинается сразу как закончится работа 1.1.

Для оптимизации расписания работ в проекте могут быть использованы различные методы. Одним из них является метод критического пути (МКП).

Критический путь – максимальный по продолжительности полный путь в сети; работы, лежащие на этом пути, также называются критическими.

Критическая работа – работа, увеличение продолжительности которой, влечет увеличение продолжительности всего проекта. На рисунке они отображены красным цветом.

Некритические работы имеют временной резерв. В случае, если этот временной резерв исчерпан в процессе реализации работы, она становится

критической, т.е. продолжительность ее выполнения начинает влиять на продолжительность всего проекта.

Этап 6 Составление диаграммы Ганта

Диаграмма Ганта – горизонтальная линейная диаграмма, на которой работы проекта представляются протяженными во времени отрезками, характеризующимися временными и другими параметрами.

Работы проекта отображаются в виде прямоугольников, однако, в отличие от сетевой диаграммы, в диаграмме Ганта, рис.3.3, длина прямоугольника соответствует продолжительности работы, Стрелки также характеризуют последовательность и взаимосвязь работ. При необходимости, можно дополнять диаграмму информацией о стоимости работ, об их исполнителях.

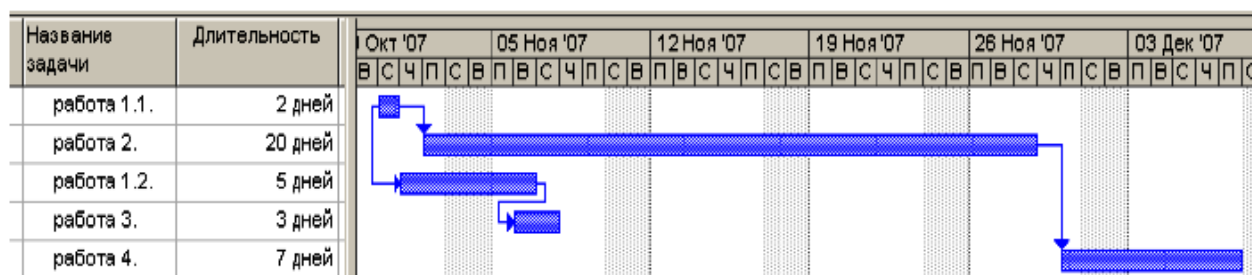


Рис.3.3. Диаграмма Ганта.

Задание на практическое занятие №1

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами Microsoft Visual Studio 2010 Express или Dia Diagram Editor (альтернатива коммерческой MS Office Visio, свободное скачивание <http://dia-installer.de/download/index.html.en>) [52,53].

1. Собрать и изучить предварительную информацию об исследуемой предметной области (варианты заданий представлены в Приложении 1).

2. Сформулировать видение выполнения проекта и границы проекта. Составить отчет об обследовании, содержащий следующие данные:

- краткая информация о предметной области;
- цели проекта;
- подразделения и пользователи системы;

3. Составить структурную декомпозицию работ.

4. Определить список работ проекта на основе СДР проекта, назначить номера работ.

5. Определение последовательности выполнения работ и их взаимосвязей с помощью организационно-технологических моделей. Уточнить временные ограничения (за основу взять последовательность выполнения практических и лабораторных работ по изучаемому курсу).

6. Определить продолжительность работ (за основу взять календарный график практических занятий и лабораторных работ по изучаемому курсу).

7. Результаты выполнения п.4,5,6 свести в табл.3.3.
8. Составить сетевую диаграмму проекта.

Таблица 3.3. Показатели продолжительности работы

| № работы | Название работы | Описание работы | Длительность работы | № предшеств. работы | № следующей работы | Временные уточнения |
|-----------------|------------------------|------------------------|----------------------------|----------------------------|---------------------------|----------------------------|
| | | | | | | |

Задание для самостоятельного выполнения №1

1. Обосновать и составить схему жизненного цикла проектируемой информационной системы (в соответствии с вариантом).
2. Определить комплекс стандартов, регламентирующих создание проектируемой информационной системы.
3. Описать этапы и стадии определенного для информационной системы жизненного цикла (в соответствии с определенным стандартом).

Содержание части отчета №1 по практическому заданию

1. Текстовое описание предметной области.
2. Описание проекта (название, цели, задачи, которые будет решать информационная система).
3. Перечень стандартов, в соответствии с которыми будет осуществляться разработка проекта.
4. Диаграмма жизненного цикла, перечень и описание этапов, стадий (вех), контрольных точек и т.д. (в зависимости от модели жизненного цикла, согласно варианту) (диаграмма представляется в виде схемы, таблицы и т.д.).
5. Схема структурной декомпозиции работ проекта.
6. Сводная таблица списка работ проекта с описанием работ (табл.3.2)
7. Сетевая диаграмма

Контрольные вопросы

1. Перечислить, какие функции выполняют в планировании проекта сетевое, календарное планирование.
2. На основании каких методов осуществляется сетевое и календарное планирование проекта?
3. Объяснить, какую роль играет определение критических операций и критического пути проекта.
4. Какие виды резервов можно определять при планировании проекта?
5. Какими методами можно определить длительность операций проекта?
6. Почему метод PERT наиболее часто используется при определении длительности операций?

Практическое занятие №2

Моделирование процесса движения информации (структурный анализ на основе DFD-диаграмм)

Цель:

- изучить общие положения о моделировании потоков данных и компоненты диаграммы потоков данных DFD;
- построить диаграмму декомпозиции в нотации DFD;

Время: 2 часа

Краткие теоретические сведения

1. Методология DFD

Диаграммы потоков данных (DFD) (см. п.1.9.1.2) являются **основным средством моделирования функциональных требований** к проектируемой системе. С их помощью эти требования представляются в виде иерархии функциональных компонентов (процессов), связанных потоками данных.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям - потребителям информации.

То есть, главная задача DFD – показать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Состав диаграмм потоков данных, табл.3.4 [27]. Основными компонентами диаграмм потоков данных являются:









- процессы (работы);
- внешние сущности;
- потоки данных.
- хранилища (накопители) данных;

Процессы (работы). В DFD работы обозначают функции или процессы системы, которые обрабатывают и изменяют информацию (преобразуют входы в выходы). Процессы изображаются прямоугольниками с закругленными углами, (смысл их совпадает со смыслом работ IDEF0 и IDEF3). Процессы в DFD-нотации имеют входы и выходы (не поддерживают управления и механизмы, как IDEF0). Все стороны работы равнозначны и не имеют конкретного назначения. В каждую работу может входить и выходить по несколько стрелок.

Внешние сущности. Внешняя сущность представляет собой материальный объект, например заказчики, персонал, поставщики, клиенты, склад, изображают входы в систему и/или выходы из системы и указывают на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но

располагаются за рамками диаграммы. Одна и та же внешняя сущность может. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы (см. п.1.9.1.2 рис.1.14, 1.15).

Таблица 3.4. Графические элементы DFD диаграмм

| Компонента | Нотация Йордона – де Марко | Нотация Гейна-Сарсона |
|-------------------------------|---|---|
| Поток данных |  |  |
| Процесс (система, подсистема) |  |  |
| Накопитель данных |  |  |
| Внешняя сущность |  |  |

Потоки данных. Поток данных определяет качественный характер информации, передаваемой от источника к приемнику. Потоки данных на диаграммах DFD изображаются линиями со стрелкой на одном из ее концов или на обоих концах. Стрелка показывает направление информационного потока в системе.

Хранилища данных. Хранилища данных представляют собой собственно данные, к которым осуществляется доступ. Эти данные могут быть созданы или изменены работами. В отличие от потоков данных, описывающих информацию в движении, хранилища данных изображают информацию в покое. Хранилища данных на диаграммах DFD изображаются прямоугольными блоками с двумя полями. В левом поле указывается номер или идентификатор хранилища. На одной диаграмме может присутствовать несколько копий одного и того же хранилища данных.

Диаграммы потоков данных дают четкое представление о том, какие данные используются и какие функции выполняются существующей информационной системой или моделируемым программным обеспечением.

На этапе построения DFD-диаграмм может оказаться, что существующие потоки информации реализованы ненадежно и нуждаются в реорганизации.

2. Этапы построения DFD-диаграммы

Первый шаг построения DFD – **построение основной контекстной диаграммы** типа «звезда», на которой в центре представлен главный моделируемый процесс и все внешние сущности, с которыми он взаимодействует

– приемники и источники информации, посредством которых, пользователи и другие внешние системы, взаимодействуют с системой.

Перед построением контекстной DFD анализируют внешние события (внешние сущности), оказывающие влияние на функционирование системы. Количество потоков на контекстной диаграмме должно быть по возможности небольшим, поскольку каждый из них может быть в дальнейшем разбит на несколько потоков на следующих уровнях диаграммы.

Для проверки контекстной диаграммы можно составить список событий. Список событий – это описания действий внешних сущностей (событий) и соответствующих реакций системы на события. Одному (или более) потоку данных соответствует одно событие: входные потоки интерпретируются как воздействия, а выходные потоки – как реакции системы на входные потоки.

Главный процесс не раскрывает структуры сложной системы (которая имеет большое количество внешних сущностей, многофункциональна, которую можно разбить на подсистемы).

Критериями сложности в данном случае являются: наличие большого числа внешних сущностей, многофункциональность системы, ее распределенный характер. Внешние сущности выделяются по отношению к основному процессу. Для их определения необходимо выделить поставщиков и потребителей основного процесса, т.е. все объекты, которые взаимодействуют с основным процессом.

На этом этапе описание взаимодействия заключается в выборе глагола, дающего представление о том, как внешняя сущность использует основной процесс, или используется им. Например, основной процесс – «учет обращений пользователей», внешняя сущность – «пользователи», описание взаимодействия – «подает заявки и получает ответы». Этот этап является принципиально важным, поскольку именно он определяет границы моделируемой системы.

Для всех внешних сущностей строится таблица событий, описывающая их взаимодействие с основным потоком. Таблица событий включает в себя наименование внешней сущности, событие, его тип (типичный для системы или исключительный, реализующийся при определенных условиях) и реакцию системы.

На *втором шаге* происходит **декомпозиция основного процесса** на набор взаимосвязанных процессов, обменивающихся потоками данных. Сами потоки не конкретизируются, определяется лишь характер взаимодействия.

Каждое событие представляется в виде процесса с соответствующими входными и выходными потоками, накопителями данных, внешними сущностями и ссылки на другие процессы для описания связей между этим процессом и его окружением.

Декомпозиция завершается, когда процесс становится простым, т.е.:

- процесс имеет относительно небольшое количество входных и выходных потоков данных (2-3 потока);
- существует возможности описания преобразования данных процессом в виде последовательного алгоритма;

- возможно выполнение процессом единственной логической функции преобразования входной информации в выходную;
- существует возможность описания логики процесса при помощи спецификации небольшого объема (не более 20 – 30 строк).

Для простых процессов строится миниспецификация – формальное описание алгоритма преобразования входных данных в выходные.

Спецификация процесса должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.

Спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании спецификации принимается аналитиком исходя из вышеперечисленных критериев.

Фактически спецификации представляют собой описания алгоритмов задач, выполняемых процессами. Спецификации содержат номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса, являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные. Известно большое количество разнообразных методов, позволяющих описать тело процесса. Соответствующие этим методам языки могут варьироваться от структурированного естественного языка или псевдокода до визуальных языков проектирования.

После декомпозиции основного процесса для каждого подпроцесса строится аналогичная таблица внутренних событий.

Третий шаг – выделение потоков данных, которыми обмениваются процессы и внешние сущности. Простейший способ их выделения заключается в анализе таблиц событий. События преобразуются в потоки данных от инициатора события к запрашиваемому процессу, а реакции – в обратный поток событий. После построения входных и выходных потоков аналогичным образом строятся внутренние потоки. Для их выделения для каждого из внутренних процессов выделяются поставщики и потребители информации. Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом.

Четвертый шаг – проверка диаграммы на полноту и непротиворечивость. Полнота диаграммы обеспечивается, если в системе нет «повисших» процессов, не используемых в процессе преобразования входных потоков в выходные. Непротиворечивость системы обеспечивается выполнением наборов формальных правил о возможных типах процессов:

- на диаграмме не может быть потока, связывающего две внешние сущности – это взаимодействие удаляется из рассмотрения;
- ни одна сущность не может непосредственно получать или отдавать информацию в хранилище данных – хранилище данных является пассивным элементом, управляемым с помощью интерфейсного процесса;
- два хранилища данных не могут непосредственно обмениваться информацией – эти хранилища должны быть объединены.

3. Правила и рекомендации для построения DFD-диаграммы

При построении DFD-диаграмм целесообразно:

- размещать на каждой диаграмме от 3 до 7 процессов. Верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего один процесс или два;
- не загромождать диаграммы не существенными на данном уровне деталями;
- декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов. Эти две работы должны выполняться одновременно, а не одна после завершения другой;
- выбирать понятные имена процессов и потоков, не использовать аббревиатуры.

При детализации процессов нужно выполнять следующие правила:

- правило балансировки – детализирующая диаграмма в качестве внешних источников или приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеют информационную связь детализируемые подсистема или процесс на родительской диаграмме;
- правило нумерации – при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т. д.

При построении спецификаций необходимо руководствоваться следующим требованиям:

- для каждого процесса нижнего уровня должна существовать одна и только одна спецификация;
- спецификация должна определять способ преобразования входных потоков в выходные;
- нет необходимости (по крайней мере, на стадии формирования требований) определять метод реализации этого преобразования;
- спецификация должна стремиться к ограничению избыточности – не следует переопределять то, что уже было определено на диаграмме;
- набор конструкций для построения спецификации должен быть простым и понятным.

Задание на практическое занятие №2

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio).

Построить DFD-диаграмму главного (основного) процесса предметной области:

1. Провести анализ внешних событий (определить внешние сущности) исследуемой предметной области, оказывающих влияние на функционирование системы.
2. Составить список событий – описаний действий внешних сущностей и соответствующих реакций системы на события.
3. Результаты выполнения п.1 и п.2. занести в табл.3.5.

Таблица 3.5. Пример описания процессов для DFD-диаграммы

| № | Внешняя сущность | Событие (описание взаимодействия) | Тип события | Основной процесс | Реакция системы на события |
|---|------------------|-----------------------------------|-------------|------------------------------|----------------------------|
| 0 | пользователи | подает заявку | типичный | учет обращений пользователей | регистрация заявки |
| | | получает ответы | типичный | | выдает ответ |

4. Выделить потоки данных, которыми обменивается процесс и внешние сущности, при необходимости ввести хранилища данных.
5. Построить DFD-диаграмму главного (основного) процесса.

Задание для самостоятельного выполнения №2

1. Произвести декомпозицию (детализацию) диаграмму детализации основного процесса.
2. Для каждого полученного подпроцесса построить таблицу внутренних событий аналогично табл.3.5.
3. Выделить потоки данных, которыми обмениваются процессы и внешние сущности, для каждого из внутренних процессов выделить поставщиков и потребителей информации, при необходимости ввести хранилища данных.

Содержание части отчета №2 по практическому заданию

1. Краткий анализ внешних и внутренних событий исследуемой предметной области, оказывающих влияние на функционирование системы.
2. Описание основного процесса и подпроцессов (табл.3.5) – описания действий внешних сущностей и внутренних событий, а также соответствующих реакций системы на события с выделенными потоками данных.
3. DFD-диаграмма главного (основного) процесса.
4. DFD-диаграммы декомпозиции основного процесса.
5. Спецификации процессов нижнего уровня.

Контрольные вопросы

1. Какова цель создания диаграммы потоков данных DFD?
2. Какая модель (по отображаемому аспекту) строится с использованием диаграмм потоков данных?

3. Описать компоненты диаграммы потоков данных, их вид и назначение.
4. Что такое процесс? Привести примеры.
5. Что такое внешняя сущность? Привести примеры.
6. Каково назначение потоков данных?
7. Что такое хранилище данных? Привести примеры.
8. Назвать основные правила и рекомендации построения диаграмм потоков данных.

Практическое занятие №3

Функциональное моделирование процессов (методология IDEF0)

Цель:

- изучить общие положения о функциональном моделировании процессов, ориентированном на потоки данных;
- построить диаграмму в нотации IDEF0;

Время: 2 часа

Краткие теоретические сведения

1. Методология IDEF0

IDEF0 (см. п.1.9.1.3) рекомендована для использования Госстандартом РФ и активно применяется в отечественных госструктурах (например, в Государственной налоговой инспекции РФ).

Результатом моделирования процессов является модель, которая относится к одному из трех типов:

- модель AS-IS (как есть) – модель текущей организации процессов;
- модель TO-BE (как будет) – модель идеальной организации процессов;
- модель SHOULD-BE (как должно бы быть) – идеализированная модель, не отражающая реальную организацию процессов.

Модель (AS-IS, TO-BE или SHOULD-BE) может содержать 4 типа диаграмм [21, 23]:

- контекстную диаграмму;
- диаграммы декомпозиции;
- диаграммы дерева узлов;
- презентационные диаграммы (диаграммы только для экспозиции) (for exposition only, FEO).

Контекстная диаграмма (диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т. е. определяются функции, из которых состоит основная.

Далее функции делятся на подфункции и так до достижения требуемого уровня детализации исследуемой системы. Диаграммы, которые описывают каждый такой фрагмент системы, называются **диаграммами декомпозиции**. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных процессов созданным диаграммам. Найденные несоответствия устраняются, после чего приступают к дальнейшей детализации процессов.

Диаграмма дерева узлов показывает иерархическую зависимость функций (работ), но не связи между ними. Их может быть сколько угодно, поскольку дерево можно построить на произвольную глубину и с произвольного узла.

Диаграммы для экспозиции (презентационные) строятся для иллюстрации отдельных фрагментов модели с целью отображения альтернативной точки зрения на происходящие в системе процессы (например, с точки зрения руководства организации).

Функциональный блок (Activity Box), рис.3.4, графически изображается в виде прямоугольника и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы.

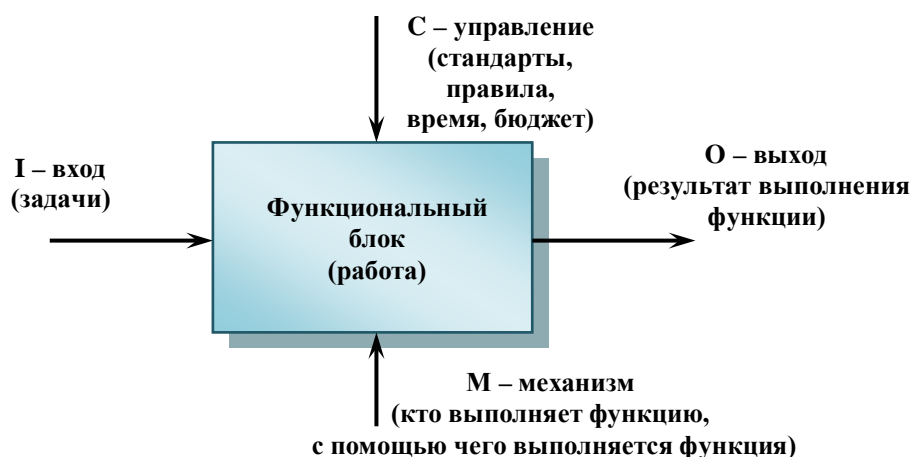


Рис.3.4. Функциональный блок и интерфейсные дуги IDEF0-диаграммы.

По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «производить услуги», а не «производство услуг»).

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

- верхняя сторона – «Управление» (Control);
- левая сторона – «Вход» (Input);
- правая сторона – «Выход» (Output);
- нижняя сторона – «Механизм» (Mechanism).

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.

Интерфейсная дуга (Arrow) или поток. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом «источником» может быть только выходная сторона блока, а «приемником» любая из трех оставшихся.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Различают стрелки четырех видов:

- *I (Input)* – вход, т.е. все, что поступает в процесс или потребляется процессом;
- *C (Control)* – управление или ограничения на выполнение операций процесса;
- *O (Output)* – выход или результат процесса;
- *M (Mechanism)* – механизм, который используется для выполнения процесса.

Каждая интерфейсная дуга должна иметь свое уникальное наименование (Arrow Label). По требованию стандарта [25,26], наименование должно быть оборотом существительного.

- функциональный блок (функция) преобразует входные объекты в выходные;
- управление определяет, когда и как это преобразование может, или должно произойти;
- исполнитель осуществляет это преобразование;

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий описания DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Третьим основным понятием стандарта IDEF0 является **декомпозиция (Decomposition)**. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Диаграмма следующего уровня, которая содержит функциональные блоки, отображающие подфункции декомпозируемого функционального блока, называется *дочерней (Child diagram)* по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме соответственно называется *дочерним блоком – (Child Box)*).

В свою очередь, функциональный блок – предок называется *родительским блоком (Parent Box)* по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит – *родительской диаграммой (Parent Diagram)*.

Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока.

В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок, или исходящие из него фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0.

Число уровней не ограничивается, зато рекомендуется на одной диаграмме использовать не менее 3 и не более 6 блоков.

Четвертым понятием IDEF0 является *глоссарий (Glossary)*.



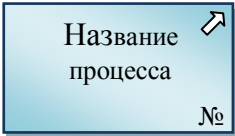

Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом.

Этот набор называется глоссарием и является описанием сущности данного элемента.

В табл.3.6 представлены графические элементы IDEF0 диаграмм.

Таблица 3.6. Графические элементы IDEF0 диаграмм

| Название | Графический символ | Описание |
|-------------------------|---|---|
| Процесс |  | Процесс обозначается прямоугольным блоком. Внутри каждого блока помещается его имя и номер. Имя должно быть активным глаголом, глагольным оборотом или отглагольным существительным. Номер блока размещается в правом нижнем углу. Номера блоков используются для идентификации на диаграмме и в соответствующем тексте. |
| Стрелка |  | Стрелки обозначают входящие и исходящие из процесса объекты (данные). Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок-стрелка. В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль. Стрелки, входящие в левую сторону блока – входы. Стрелки, входящие в блок сверху -управления. Стрелки, покидающие процесс справа – выходы, т.е. данные или материальные объекты, произведенные процессом. Стрелки, подключенные к нижней стороне блока, представляют механизмы. |
| Туннелированная стрелка |  | Туннелированные стрелки означают, что данные, передаваемые с помощью этих стрелок, не рассматриваются на родительской диаграмме и/или на дочерней диаграмме. Стрелка, помещенная в туннель там, где она присоединяется к блоку, означает, что данные, выраженные этой стрелкой, не обязательны на следующем уровне декомпозиции. Стрелка, помещаемая в туннель на свободном конце, означает, что выраженные ею данные |

| Название | Графический символ | Описание |
|-----------------------|---|---|
| | | отсутствуют на родительской диаграмме. Туннелированные стрелки могут быть использованы на диаграммах процессов в нотациях IDEF0, «Процесс», «Процедура». |
| Внешняя ссылка |  Имя внешней ссылки | Элемент обозначает место, сущность или субъект, которые находятся за границами моделируемой системы. Внешние ссылки используются для обозначения источника или приемника стрелки вне модели. На диаграммах внешняя ссылка изображается в виде квадрата, рядом с которым показано наименование Внешней ссылки. Внешние ссылки могут быть использованы на диаграммах процессов в любых нотациях. |
| Междиаграммная ссылка |  | Элемент, обозначающий другую диаграмму. Междиаграммная ссылка служит для обозначения перехода стрелок на диаграмму другого бизнес-процесса без отображения стрелки на вышележащей диаграмме (при использовании иерархических моделей). В качестве междиаграммной ссылки не может выступать диаграмма процесса в нотации EPC. Междиаграммные ссылки могут быть использованы на диаграммах процессов в нотациях IDEF0, «Процесс», «Процедура». |
| Процесс-ссылка |  | Элемент обозначает ссылку на процесс, описанный в другой модели. Наиболее часто повторяющиеся процессы в рамках модели бизнес-процессов могут быть выделены в качестве типовых в отдельную папку в Навигаторе. Диаграмма типового процесса формируется один раз в одном месте Навигатора. Далее на любой диаграмме может быть использован процесс-ссылка на типовой процесс. Параметры типового процесса заполняются непосредственно в Окне свойств типового процесса. Постоянный список субъектов, принимающих участие в выполнении типового процесса, формируется также в Окне свойств типового процесса. Список субъектов, принимающих участие при выполнении типового процесса в рамках вышележащего процесса, формируется в Окне свойств процесса-ссылки на типовой процесс. Процессы-ссылки могут быть использованы на диаграммах процессов в любых нотациях. |
| Сноска |  сноска | Выносной элемент, предназначенный для нанесения комментариев. Элемент может быть использован на диаграммах процессов в любых нотациях. |
| Текст | Текстовое описание | Комментарий без сноски. Элемент может быть использован на диаграммах процессов в любых нотациях. |

2. Этапы построения IDEF0-диаграммы

Первый шаг построения IDEF0 – **построение основной контекстной диаграммы**, то есть с представления всей системы в виде простейшей компоненты (контекстной диаграммы). Данная диаграмма отображает назначение (основную функцию) системы, необходимые входные и выходные данные, управляющую и регламентирующую информацию, а также механизмы.

То есть модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока, с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором «A-0», рис.3.5.

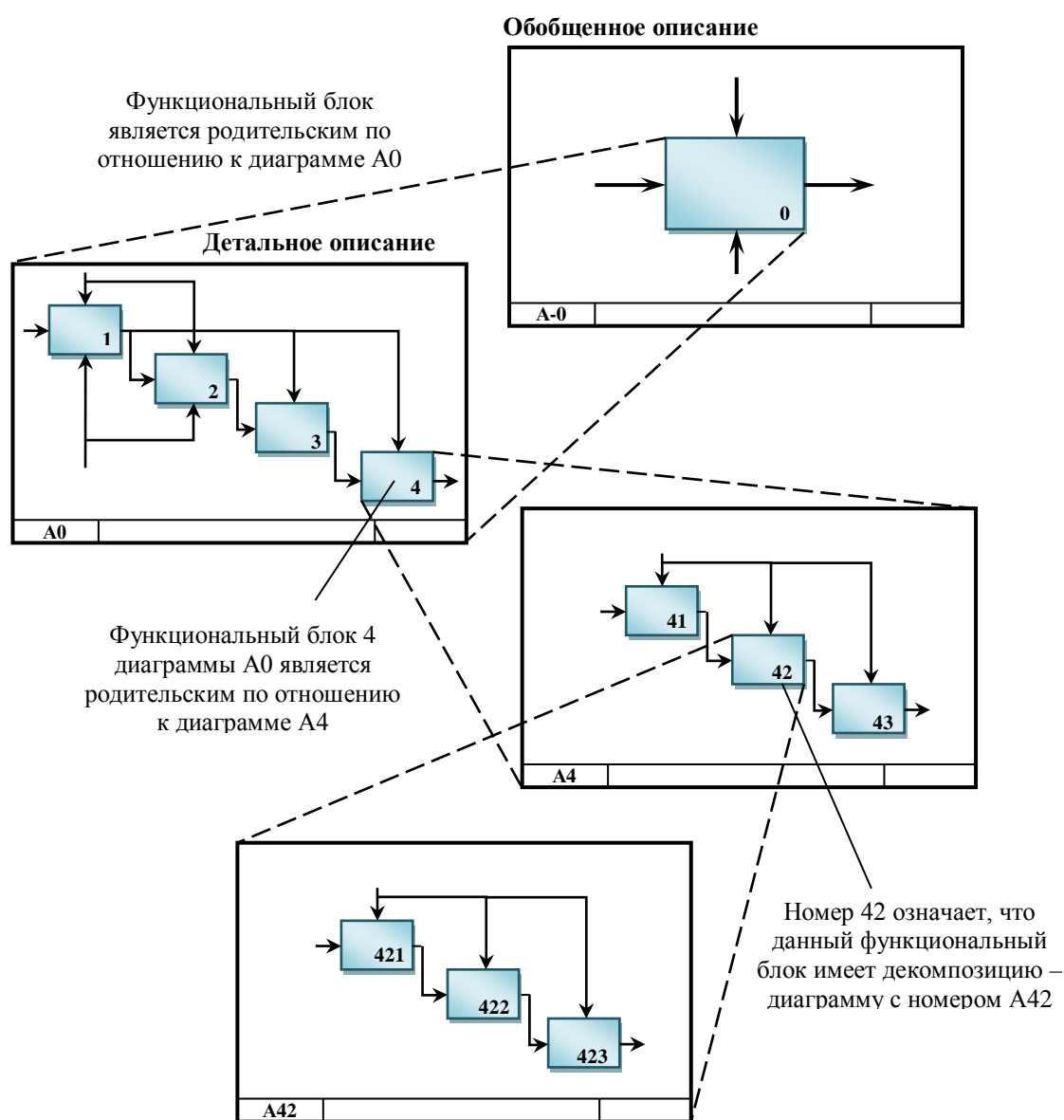


Рис.3.5.Декомпозиция функциональных блоков.

На **втором шаге** контекстная диаграмма детализируется с помощью **диаграммы декомпозиции** первого уровня. На этой диаграмме отображаются

функции системы, которые должны быть реализованы в рамках основной функции. Диаграмма, для которой выполнена декомпозиция, по отношению к детализирующим ее диаграммам называется родительской. Диаграмма декомпозиции по отношению к родительской называется дочерней, рис.3.5.

Как правило, при построении диаграммы декомпозиции исходная функция (декомпозируемая) разбивается на 3–8 подфункций (блоков). При этом блоки на диаграмме декомпозиции рекомендуется располагать слева направо сверху вниз, чтобы лучше была видна последовательность и логика взаимодействия подфункций.

После построения диаграммы декомпозиции первого уровня для указанных на ней функций строятся отдельные диаграммы (диаграммы декомпозиции второго уровня). Затем процесс декомпозиции (построения диаграмм) продолжается до тех пор, пока дальнейшая детализация функций не теряет смысла. Для каждой атомарной функции, описывающей элементарную операцию (т.е. функции, не имеющей диаграмму декомпозиции), составляется подробная спецификация, определяющая ее особенности и алгоритм реализации. В качестве дополнения к спецификации могут использоваться блок-схемы алгоритмов. Таким образом, процесс функционального моделирования заключается в постепенном выстраивании иерархии функций.

Стрелки, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются теми же самыми, что и стрелки, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы. Как следствие этого, границы функции верхнего уровня – это то же самое, что и границы диаграммы декомпозиции.

ICOM-коды (аббревиатура от Input, Control, Output и Mechanism) предназначены для идентификации граничных стрелок. ICOM-код содержит префикс, соответствующий типу стрелки (I, C, O или M), и порядковый номер.

После определения состава функций и взаимосвязей между ними, возникает вопрос о правильной их композиции (объединении) в модули (подсистемы). При этом подразумевается, что каждая отдельная функция должна решать одну, строго определенную задачу. В противном случае необходима дальнейшая декомпозиция или разделение функций.

При объединении функций в подсистемы необходимо стремиться, чтобы внутренняя связность (между функциями внутри модуля) была как можно сильнее, а внешняя (между функциями, входящими в разные модули), как можно слабее. Опираясь на семантику связей методологии IDEF0, вводится классификация связей между функциями (работами). Данная классификация является расширением [21]. **Типы связей** приводятся в порядке уменьшения их значимости (силы связывания).

- *иерархическая связь* (связь «часть» – «целое») имеет место между функцией и подфункциями, из которых она состоит;

- *регламентирующая связь* (управляющая, подчиненная) отражает зависимость одной функции от другой, когда выход одной работы направляется на управление другой. Функцию, из которой выходит управление, следует считать регламентирующей или управляющей, а в которую входит – подчиненной.

Различают прямую связь по управлению, когда управление передается с вышестоящей работы на нижестоящую, и обратную связь по управлению, когда управление передается от нижестоящей к вышестоящей;

- *функциональная связь* (технологическая) имеет место, когда выход одной функции служит входными данными для следующей функции. С точки зрения потока материальных объектов данная связь показывает технологию (последовательность работ) обработки этих объектов. Различают прямую связь по входу, когда выход передается с вышестоящей работы на нижестоящую, и обратную связь по входу, когда выход передается с нижестоящей к вышестоящей;

- *потребительская связь* имеет место, когда выход одной функции служит механизмом для следующей функции. Таким образом, одна функция потребляет ресурсы, вырабатываемые другой;

- *логическая связь* наблюдается между логически однородными функциями. Такие функции, как правило, выполняют одну и ту же работу, но разными (альтернативными) способами или, используя разные исходные данные (материалы);

- *коллегиальная связь* (методическая) имеет место между функциями, алгоритм работы которых определяется одним и тем же управлением. Аналогом такой связи является совместная работа сотрудников одного отдела (коллег), подчиняющихся начальнику, который отдает указания и приказы (управляющие сигналы). Такая связь также возникает, когда алгоритмы работы этих функций определяются одним и тем же методическим обеспечением (СНИП, ГОСТ, официальными нормативными материалами и т.д.), служащим в качестве управления;

- *ресурсная связь* возникает между функциями, использующими для своей работы одни и те же ресурсы. Ресурсно-зависимые функции, как правило, не могут выполняться одновременно.

- *информационная связь* имеет место между функциями, использующими в качестве входных данных одну и ту же информацию;

- *временная связь* возникает между функциями, которые должны выполняться одновременно до или одновременно после другой функции. Эта связь имеет место также между другими сочетаниями управления, входа и механизма, поступающими в одну функцию;

- *случайная связь* возникает, когда конкретная связь между функциями мала или полностью отсутствует.

Из приведенных выше типов связей наиболее сильной является иерархическая связь, которая, по сути, и определяет объединение функций в модули (подсистемы). Несколько слабее являются регламентирующие, функциональные и потребительские связи. Функции с этими связями обычно реализуются в одной подсистеме. Логические, коллегиальные, ресурсные и информационные связи одни из самых слабых. Функции, обладающие ими, как правило, реализуют в разных подсистемах, за исключением логически однородных функций (функций, связанных логической связью). Временная связь

свидетельствует о слабой зависимости функций друг от друга и требует их реализации в отдельных модулях.

Таким образом, при объединении функций в модули наиболее желательными являются первые пять видов связей. Функции, связанные последними пятью связями, лучше реализовывать в отдельных модулях.

Третий шаг построение *диаграммы дерева узлов*. Это обзорная диаграмма, показывающая структуру всей модели. Обычно вершина дерева соответствует контекстному блоку, под вершиной выстраивается вся иерархия блоков модели. Однако не запрещается назначать вершиной произвольный блок, помещая под ним все его дочерние блоки. Из-за высокой итеративности функционального моделирования можно ожидать, что дерево модели будет неоднократно изменяться существенным образом до тех пор, пока не будет получена его стабильная версия. Обзор модели с использованием дерева помогает сконцентрироваться на функциональной декомпозиции модели.

Третий шаг построение *презентационных FEO-диаграмм* для иллюстрации других точек зрения или деталей, выходящих за рамки традиционного синтаксиса IDEF0. Диаграммы FEO допускают нарушение любых правил построения диаграмм IDEF0 в целях выделения важных с точки зрения аналитика частей модели. FEO-диаграмма как правило выглядит как обыкновенная диаграмма IDEF0, удовлетворяя всем ограничениям IDEF0.

Один из способов использования FEO-диаграмм состоит в отделении функционального блока от его окружения посредством создания диаграммы с единственным блоком и всеми относящимися к нему стрелками наподобие контекстной диаграммы.

Кроме того, встречаются следующие виды презентационных диаграмм:

- копия диаграммы IDEF0, которая содержит все функциональные блоки, и стрелки, относящиеся только к одному из функциональных блоков, – это позволяет отразить взаимодействие между этим блоком и другими объектами диаграммы;
- копия диаграммы IDEF0, которая содержит все функциональные блоки, и стрелки, непосредственно относящиеся только к входу и (или) к выходу родительского блока;
- различные точки зрения, как правило, на глубину одного уровня декомпозиции.

3. Правила и рекомендации для построения IDEF0-диаграммы

В IDEF0 существуют соглашения (правила и рекомендации) по созданию диаграмм, которые призваны облегчить чтение и экспертизу модели [21, 23, 55]. Некоторые из этих правил CASE-средства поддерживают автоматически, выполнение других следует обеспечить вручную.

1. Перед построением модели необходимо определиться, какая модель (модели) системы будет построена. Это подразумевает определение ее типа: AS-IS (как есть), TO-BE (как будет) или SHOULD-BE (как должно быть), а также определения позиции, с точки зрения которой строится модель.

«Точку зрения» лучше всего представлять себе как место (позицию) человека или объекта, в которое надо встать, чтобы увидеть систему в действии.

Обычно выбирается одна точка зрения, наиболее полно охватывающая все нюансы работы системы, и при необходимости для некоторых диаграмм декомпозиции строятся диаграммы ФЕО, отображающие альтернативную точку зрения.

2. На контекстной диаграмме отображается один блок, показывающий назначение системы. Для него рекомендуется отображать по 2-4 стрелки, входящие и выходящие с каждой стороны.

3. Количество блоков на диаграммах декомпозиции рекомендуется в пределах 3-6. Если на диаграмме декомпозиции два блока, то она, как правило, не имеет смысла. При наличии большого количества блоков диаграмма становится перенасыщенной и трудно читаемой.

4. Блоки на диаграмме декомпозиции следует располагать слева направо и сверху вниз. Такое расположение позволяет более четко отразить логику и последовательность выполнения работ. Кроме этого маршруты стрелок будут менее запутанными, и иметь минимальное количество пересечений.

5. Отсутствие у функции одновременно стрелок управления и входа не допускается. Это означает, что запуск данной функции не контролируется и может произойти в любой произвольный момент времени либо вообще никогда.

Блок с наличием только управления можно рассматривать как вызов в программе функции (процедуры) без параметров. Если у блока имеется вход, то он эквивалентен вызову в программе функции с параметрами. Таким образом, блок без управления и входа эквивалентен функции, которая в программе ни разу не вызывается на исполнение.

6. У каждого блока должен быть как минимум один выход. Работы без результата не имеют смысла и не должны моделироваться. Исключение составляют работы, отображаемые в модели AS-IS. Их наличие свидетельствует о неэффективности и несовершенстве технологических процессов. В модели TO-BE эти работы должны отсутствовать.

7. При построении диаграмм следует минимизировать число пересечений, петель и поворотов стрелок.

8. Обратные связи и итерации (циклические действия) могут быть изображены с помощью обратных дуг. Обратные связи по входу рисуются «нижней» петлей, обратная связь по управлению – «верхней».

9. Каждый блок и каждая стрелка на диаграммах должны обязательно иметь имя. Допускается использовать ветвление (декомпозицию) или слияние (композицию) стрелок. Это связано с тем, что одни и те же данные или объекты, порожденные одной работой, могут использоваться сразу в нескольких других работах. И наоборот, одинаковые или однородные данные и объекты, порожденные разными работами, могут использоваться в одном месте. При этом допускается задание различным ветвям стрелки уточняющих имен после разветвления (до слияния). Если какая-либо ветвь после ветвления не именована, то считается, что ее имя соответствует имени стрелки, записанному до ветвления.

На диаграмме не допускается рисовать стрелки, когда до и после ветвления они не именованы.

10. При построении диаграмм для лучшей их читаемости может использоваться механизм туннелирования стрелок. Например, чтобы не загромождать лишними деталями диаграммы верхних уровней (родительские), на диаграммах декомпозиции начало дуги помещают в тоннель.

11. Все стрелки, входящие и выходящие из блока, при построении для него диаграммы декомпозиции должны быть отображены на ней. Исключение составляют затуннелированные стрелки. Имена стрелок, перенесенных на диаграмму декомпозиции, должны совпадать с именами, указанными на диаграмме верхнего уровня.

12. Если две стрелки проходят параллельно (начинаются из одной и той же грани одной работы и заканчиваются на одной и той же грани другой работы), то по возможности следует их объединить и называть единым термином.

13. Каждый блок на диаграммах должен иметь свой номер. Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Блок на диаграмме верхнего уровня обозначается 0, блоки на диаграммах второго уровня – цифрами от 1 до 9 (1, 2, ..., 9), блоки на третьем уровне – двумя цифрами, первая из которых указывает на номер детализируемого блока с родительской диаграммы, а вторая номер блока по порядку на текущей диаграмме (11, 12, 25, 63) и т. д. Контекстная диаграмма имеет обозначение «А – 0», диаграмма декомпозиции первого уровня – «А0», диаграммы декомпозиции следующих уровней – состоят из буквы «А», за которой следует номер декомпозируемого блока (например, «А11», «А12», «А25», «А63»).

Задание на практическое занятие №3

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio).

1. Построить контекстную диаграмму (диаграмма А-0 – модель окружения).

2. Составить описание процесса в табличном виде и занести в табл.3.7, таблице дать название «Диаграмма А-0».

Таблица 3.7. Пример описания процессов для контекстной IDEF0-диаграммы

| Шифр | Название процесса | Входные данные | Управляющие данные | Механизм | Вызов | Результат процесса (работы) |
|------|------------------------------|----------------|-----------------------|---------------|-------------------------------|-----------------------------|
| А-0 | Учет обращений пользователей | Подача заявки | Реестр пользователей | Администратор | Проверка данных пользователей | Ответ на заявку |
| | | | Номы времени на ответ | | | |

3. Построить диаграмму декомпозиции первого уровня (диаграмма А0).

4. Составить описание процессов в табличном виде и занести в табл.3.8, таблице дать название «Диаграмма A0».

Таблица 3.8. Пример описания процессов для IDEF0-диаграммы декомпозиции A0

| Шифр | Название процесса | Входные данные | Управляющие данные | Механизм | Вызов | Результат процесса (работы) |
|------|-------------------|----------------|--------------------|----------|-------|-----------------------------|
| A1 | ... | ... | ... | ... | ... | ... |
| A2 | | | | | | |
| ... | ... | ... | ... | ... | ... | ... |

5. Построить диаграмму дерева узлов.

6. Ввести альтернативную точку зрения на один процесс и построить презентационную FEO-диаграмму.

Задание для самостоятельного выполнения №3

1. Построить диаграммы декомпозиции второго и третьего уровней, составить описание процессов в виде таблиц аналогичных табл.3.6. (таблица «Диаграмма A1», ..., «Диаграмма A21» и т.д.)

2. Перестроить диаграмму дерева узлов, учитывая произведенную дальнейшую декомпозицию.

3. Ввести альтернативные точки зрения на два процесса и построить презентационные FEO-диаграмму.

Содержание части отчета №3 по практическому заданию

1. Описание процесса диаграммы A-0 (табл.3.7).
2. Описание процессов диаграммы A0 (табл.3.8).
3. Описание процессов диаграмм декомпозиции следующих уровней в табличном виде.
4. IDEF0-диаграмма декомпозиции, сведенная к диаграмме A0.
5. Диаграмма дерева узлов.
6. Презентационные FEO-диаграммы.

Контрольные вопросы

1. Что отображается на функциональной модели системы?
2. Дать краткую характеристику моделей AS-IS, TO-BE и SHOULD-BE.
3. Перечислить виды диаграмм, используемых в IDEF0.
4. Что показывает контекстная диаграмма?
5. Перечислить элементы графической нотации IDEF0.
6. Назвать назначение ICOM-кодов.
7. Перечислить типы связей между функциями.

8. Назвать основной принцип, определяющий объединение функций в модули.
9. Назвать правила именования стрелок при их ветвлении.
10. Какие стрелки должны обязательно входить и выходить из блока (функции)?
11. Для чего применяется механизм туннелирования стрелок?
12. Какие связи отображаются на диаграмме дерева узлов?
13. Перечислить отличия методологии IDEF0 от DFD.

Практическое занятие №4
Моделирование данных (методология ERD), информационное
моделирование процессов, построение реляционных информационных
структур (методология IDEF1, IDEF1X)

Цель:

- закрепить навыки построения информационной модели в нотации П.Чена;
- освоить методологию построения информационной модели в нотации IDEF1 (IDEF1X).

Время: 2 часа

Краткие теоретические сведения

1. Методологии информационного моделирования ERD, IDEF1 (IDEF1X)

Различают два уровня информационного моделирования: логический и физический.

Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity Relationship Diagram, ERD);
- модель данных, основанная на ключах (Key Based model, KB);
- полная атрибутивная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи «многие-ко-многим» и не включать описание ключей.

Как правило, ERD (см. п.1.9.1.4) используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах – более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи (IDEF1X).

Физическая модель отражает физические свойства проектируемой базы данных (типы данных, размер полей, индексы). Параметры физической

информационной модели зависят от выбранной системы управления базами данных (СУБД).

Базовыми понятиями ERD являются:

Сущность (Entity) – множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и др.), обладающих общими атрибутами или характеристиками. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, ГРАЖДАНИН, а не ПЕТРОВ);

Связь (Relationship) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

Каждая связь может иметь один из следующих типов:

- связь типа один-к-одному (1:1) означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле имеется всего одна сущность, неправильно разделенная на две;

- 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 «мноغو») – дочерней (1:*n*);

- 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. Тип связи много-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности (*m:n*).

Каждая связь может иметь одну из двух модальностей связи:

- модальность «может» означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть, и не связан ни с одним экземпляром (например: каждый пользователь может подать несколько заявок);

- модальность «должен» означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности (например: логин должен принадлежать только одному пользователю).

Связь может иметь разную модальность с разных концов.

Атрибут (Attribute) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств,

ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т.д.). Экземпляр атрибута – это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. На диаграмме «сущность-связь» атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности – называется *первичным ключом (PrimaryKey)*.

При выборе первичного ключа можно внести в сущность дополнительный атрибут и сделать его ключом. Так, для определения первичного ключа часто используют уникальные номера, которые могут автоматически генерироваться системой при добавлении экземпляра сущности в базы данных.

На рис.3.6 представлены графические элементы ERD диаграмм в нотации П.Чена.

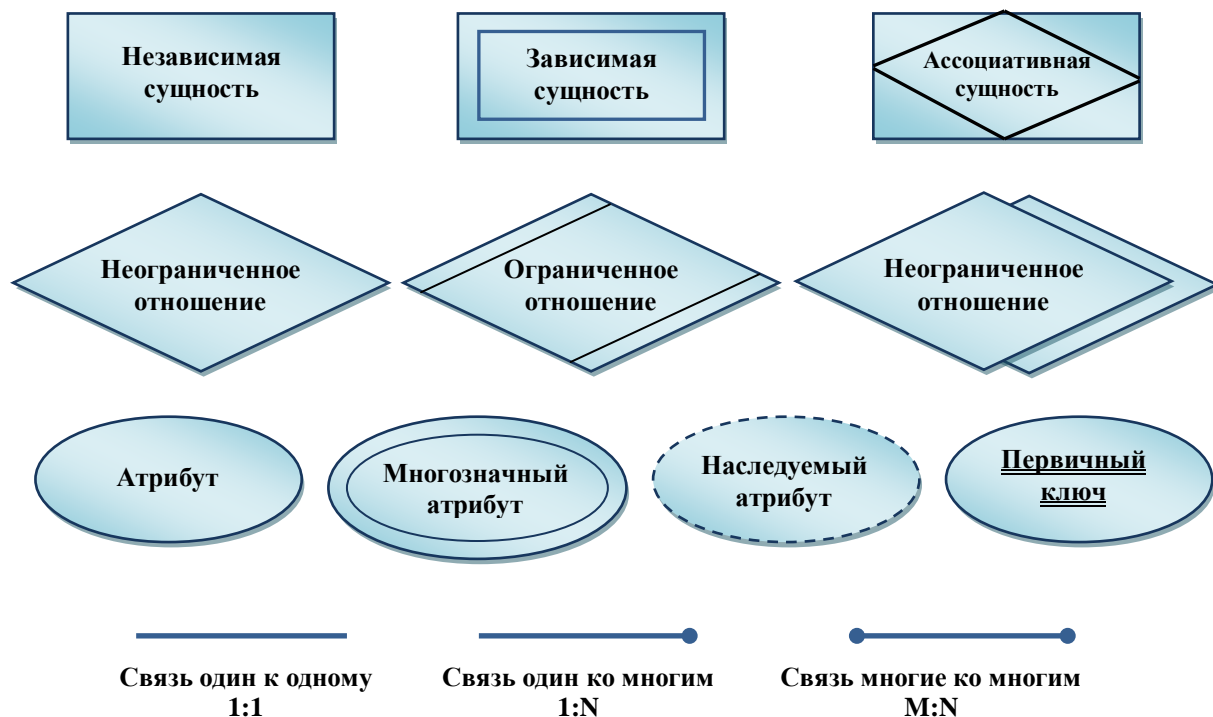


Рис.3.6. Символы ERD , соответствующие сущностям, атрибутам, отношениям и связям в нотации П.Чена.

Методология IDEF1X – язык для семантического моделирования данных, основанный на концепции «сущность-связь» (ERD) (п.1.9.1.6.) Графическая нотация IDEF1 применяется для построения информационной модели, эквивалентной реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEF1 создана его новая версия – метод IDEF1X, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации [31,32,33].

В IDEF1X все сущности делятся на *зависимые* и *независимые* от идентификаторов. Сущность является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Независимая сущность изображается в виде обычного прямоугольника, зависимая – в виде прямоугольника с закругленными углами.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется *идентифицирующей*, в противном случае – *неидентифицирующей*.

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

Пунктирная линия изображает неидентифицирующую связь. Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

Связь может дополнительно определяться с помощью указания *степени или мощности* (количества экземпляров сущности-потомка, которое может порождать каждый экземпляр сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. Мощность связей может принимать следующие значения:

- N – ноль, один или более;
- Z – ноль или один;
- P – один или более;
- цифра – заранее заданное число экземпляров сущности-потомка;

По умолчанию мощность связей принимается равной N.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Сущности могут иметь также *внешние ключи (Foreign Key)*, которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Для обозначения внешнего ключа внутри блока сущности помещают имена атрибутов, после которых следуют буквы FK в скобках.

Идентификация сущностей. Представление о ключах. Сущность описывается в диаграмме IDEF1X графическим объектом в виде прямоугольника. На рис.1.21 приведен пример IDEF1X диаграммы.

Каждый прямоугольник, отображающий собой сущность, разделяется горизонтальной линией на часть, в которой расположены ключевые поля и часть, где расположены неключевые поля. Верхняя часть называется ключевой областью, а нижняя часть областью данных. Ключевая область объекта содержит поле «Уникальный идентификатор», в области данных находятся поля «Атрибут 1», «Атрибут 2» т.д.

Ключевая область содержит первичный ключ для сущности. Первичный ключ – это набор атрибутов, выбранных для идентификации уникальных экземпляров сущности. Атрибуты первичного ключа располагаются над линией в ключевой области. Как следует из названия, неключевой атрибут – это атрибут, который не был выбран ключевым. Неключевые атрибуты располагаются под чертой, в области данных.

Выбор первичного ключа для сущности является очень важным шагом, и требует большого внимания. В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов. Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты). Кандидаты в ключи должны уникально идентифицировать каждую запись сущности. В соответствии с этим, ни одна из частей ключа не может быть NULL, не заполненной или отсутствующей.

Правила устанавливают, что атрибуты и группы атрибутов должны:

- уникальным образом идентифицировать экземпляр сущности;
- не использовать NULL значений;
- не изменяться со временем. Экземпляр идентифицируется при помощи ключа. При изменении ключа, соответственно меняется экземпляр;
- быть как можно более короткими для использования индексирования и получения данных. Если необходимо использовать ключ, являющийся комбинацией ключей из других сущностей, то нужно убедиться в том, что каждая из частей ключа соответствует правилам.

При выборе первичного ключа для сущности, разработчики модели часто используют дополнительный (суррогатный) ключ, т.е. произвольный номер, который уникальным образом определяет запись в сущности. Суррогатный ключ лучше всего подходит на роль первичного ключа потому, что является коротким и быстрее всего идентифицирует экземпляры в объекте. К тому же суррогатные ключи могут автоматически генерироваться системой так, чтобы нумерация была сплошной, т.е. без пропусков.

Потенциальные ключи, которые не выбраны первичными, могут быть использованы в качестве вторичных или альтернативных ключей. С помощью

альтернативных ключей часто отображают различные индексы доступа к данным в конечной реализации реляционной базы.

Если сущности в IDEF1X диаграмме связаны, связь передает ключ (или набор ключевых атрибутов) дочерней сущности. Эти атрибуты называются внешними ключами. Внешние ключи определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь. Передаваемые атрибуты называются мигрирующими.

Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями.

Неидентифицирующие связи отображаются пунктирной линией между объектами. Так как переданные ключи в неидентифицирующей связи не являются составной частью первичного ключа дочерней сущности, то этот вид связи не проявляется ни в одной идентифицирующей зависимости.

Тем не менее, взаимосвязь может отражать зависимость существования, если бизнес правило для взаимосвязи определяет то, что внешний ключ не может принимать значение NULL. Если внешний ключ должен существовать, то это означает, что запись в дочерней сущности может существовать только при наличии ассоциированной с ним родительской записи.

2. Этапы построения ERD-диаграммы

На *первом шаге* производится определение сущностей. Для построения ER-диаграммы «с нуля» производится анализ предметной области и выделяются информационные объекты проектируемой системы, другими словами составляется список (пул) потенциальных сущностей (как правило, выделяются все существительные в текстовом описании предметной области). В случае если имеется DFD-диаграмма, то в этом случае в качестве сущностей можно взять названия интерфейсных дуг.

На *втором шаге* необходимо описать каждый информационный объект набором характеристик (атрибутов), которые представляют важность с точки зрения выполняемых системой функций, то есть из списка потенциальных сущностей выделяются сущности, а остальное преобразуется в атрибуты сущностей.

На *третьем шаге* устанавливаются отношения и связи между сущностями по описанию предметной области на естественном языке. Определяются виды отношений и типы связей.

На *четвертом шаге* из списка атрибутов выделить атрибуты, способные однозначно идентифицировать экземпляры сущности, то есть определить первичные ключи. Корректируются

На *пятом шаге* строится ER-диаграмма, п.1.9.1.4. рис.1.18.

3. Правила и рекомендации для построения ERD-диаграммы

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от

всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- иметь уникальное имя; к одному и тому же имени должна всегда применяться одна и та же интерпретация; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- иметь один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;
- иметь один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности *первичный ключ (PrimaryKey)*.

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

4. Этапы построения IDEF1 (IDEF1X)-диаграммы

На *первом шаге*. Из списка сущностей выделяются зависимые и независимые сущности, определяются типы и мощность связей между сущностями.

На *втором шаге* проверяется правильность выбранных первичных ключей и установленных отношений при создании модели данных верхнего уровня (ER-диаграммы). Создается логическая модель данных, основанная на ключах.

На *третьем шаге* осуществляется нормализация полной атрибутивной модели, то есть процесс проверки и реорганизации сущностей и атрибутов с целью устранения избыточности и дублирования информации. Для получения IDEF1X информационной модели необходимо привести ее к третьей нормальной форме.

Задание на практическое занятие №4

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio).

1. Определить список (пул) информационных объектов (словарь данных) для проектируемой системы, составить таблицу потенциальных сущностей, табл.3.9.

Таблица 3.9. Список потенциальных сущностей

| № | Сущность | Описание |
|---|------------|-----------------------------|
| 1 | Сущность 1 | Краткое описание сущности 1 |
| 2 | Сущность 2 | Краткое описание сущности 2 |
| | | |
| m | Сущность m | Краткое описание сущности m |

2. Разделить список на сущности и их атрибуты, преобразовать таблицу 3.7 в соответствующую ей табл. 3.10.

3. Составить описание предметной области на естественном языке, пользуясь следующей схемой построения фраз:

<Каждый экземпляр Сущности 1> <модальность связи> <наименование связи>
 <тип связи> <экземпляр Сущности 2>
 описание представить в виде табл.3.11.

Таблица 3.10. Список сущностей и их атрибутов

| № | Сущность | Атрибут |
|-----|------------|-----------|
| 1. | Сущность 1 | Атрибут 1 |
| | | Атрибут 2 |
| | | ... |
| | | Атрибут N |
| ... | ... | |
| m | Сущность m | Атрибут N |
| | | ... |
| | | Атрибут K |

Таблица 3.11. Описание предметной области

| № | Описание предметной области на естественном языке |
|---|--|
| | каждый пользователь (сущность 1) <может>< подать> <одну или несколько> заявок (сущность 2) |
| | ... |

4. Определить имена отношений, типы связей между сущностями, задать мощности связей между сущностями, результат представить в виде табл.3.12.

Таблица 3.12. Матрица отношений между сущностями

| | Сущность 1 | заявка | Сущность 3 | | Сущность m |
|--------------|-----------------------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| пользователь | | подает вид связи (один ко многим 1:N) | | | имя отношения 2 (вид связи) |
| Сущность 2 | имя отношения 3 (вид связи) | | | | |
| Сущность 3 | | | | | |
| ... | | | имя отношения 4 (вид связи) | | |
| Сущность m | | | | имя отношения k (вид связи) | |

5. Определить ключевые атрибуты для каждой сущности (или ввести необходимые атрибуты, которые станут первичными ключами). Скорректировать табл.3.10 и представить в виде табл.3.13.

6. Построить информационную модель уровня «сущность-связь» – ER-диаграмму в нотации П.Чена.

Таблица 3.13. Список сущностей, атрибутов, ключевых атрибутов

| № | Сущность | Атрибут |
|-----|------------|--------------------|
| 1. | Сущность 1 | ключевой атрибут 1 |
| | | ... |
| | | Атрибут N |
| ... | ... | |
| m | Сущность m | ключевой атрибут m |
| | | ... |
| | | Атрибут K |

Задание для самостоятельного выполнения №4

Основываясь на модели данных верхнего уровня (ERD) построить IDEF1X-диаграмму, для этого:

1. Определить зависимые (обычный прямоугольник) и независимые (прямоугольник с закругленными углами) сущности (прямоугольники разделены линией на две зоны: верхняя зона – зона атрибутов первичного ключа и нижняя зона – область неключевых атрибутов).

2. Определить идентифицирующие (сплошная линия с точкой на конце у сущности-потомка) и неидентифицирующие (пунктирная линия с точкой на конце у сущности-потомка) связи между сущностями.

3. Определить мощности связей (проставить индексы: N, P, Z или цифра).

4. Построить логическую модель данных, основанную на ключах (Key Based, KB), для этого необходимо проверить правильность первичного ключа, выбранного при построении модели данных верхнего уровня ER-диаграммы. То есть, должны соблюдаться следующие требования:

- первичный ключ должен быть подобран таким образом, чтобы по значениям атрибутов, в него включенных, можно было точно идентифицировать экземпляр сущности;

- никакой из атрибутов первичного ключа не должен иметь нулевое значение;

- значения атрибутов первичного ключа не должны меняться. Если значение изменилось, значит, это уже другой экземпляр сущности;

- можно внести в сущность дополнительный атрибут и сделать его ключом.

Первичный ключ, выбранный при создании логической модели, может быть неудачным для осуществления эффективного доступа к базам данных и должен быть изменен при проектировании физической модели.

Потенциальный ключ, не ставший первичным, называется альтернативным ключом (AlternateKey). При создании альтернативного ключа на диаграмме рядом с атрибутом появляются символы (AK).

При проведении связи между двумя сущностями в дочерней сущности автоматически образуются внешние ключи (ForeignKey). Атрибуты, входящие в первичный ключ родительской или общей сущности, наследуются в качестве атрибутов сущностью-потомком или категориальной сущностью соответственно.

Эти атрибуты и называются внешними ключами обозначаются символами (FK) после своего имени.

При установлении идентифицирующей связи дочерняя сущность автоматически превращается в зависимую. Атрибуты первичного ключа родительской сущности автоматически мигрируют в зону атрибутов первичного ключа дочерней сущности как внешние ключи (ForeignKey).

5. Дополнить сущности неключевыми атрибутами, тем самым получить полную атрибутивную модель FA (FullerAttributer)

6. Провести нормализацию полной атрибутивной модели к третьей нормальной форме.

Содержание части отчета №4 по практическому заданию

1. Список (пул) информационных объектов (словарь данных), табл.3.9.
2. Список сущностей и их атрибутов, табл.3.10.
3. Описание предметной области на естественном языке, табл.3.11.
4. Матрица отношений между сущностями, табл.3.12.
5. Список сущностей и их ключевых и неключевых атрибутов, табл.3.13.
6. ER-диаграмма в нотации П.Чена.
7. Логическая модель данных, основанная на ключах.
8. Полная атрибутивная модель.
9. IDEF1X-диаграмма (полная атрибутивная модель в третьей нормальной форме).

Контрольные вопросы

1. Какова цель информационного моделирования?
2. Понятие сущности, атрибута, связи.
3. Понятие первичного и внешнего ключа.
4. Изображение ключей на диаграммах IDEF1X.
5. Понятие зависимой и независимой сущности.
6. Изображение сущностей на диаграммах IDEF1X.
7. Понятие идентифицирующей и неидентифицирующей связи.
8. Изображение связей на диаграммах IDEF1X.
9. Понятие мощности связи. Обозначение мощностей связей на диаграммах IDEF1X.

Практическое занятие №5

Описания логики взаимодействия информационных потоков (методология IDEF3)

Цель:

- изучить общие положения о функциональном моделировании процессов, ориентированным на потоки данных;
- построить диаграмму логики взаимодействия информационных потоков в нотации IDEF3;

Время: 2 часа

Краткие теоретические сведения

1. Методология IDEF3

IDEF3(п.1.9.1.7) – способ описания процессов, основной целью которого является обеспечение структурированного метода, используя который эксперт в предметной области может описать положение вещей как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу.

Технология IDEF3 хорошо приспособлена для сбора данных, требующихся для проведения структурного анализа системы. В отличие от большинства технологий моделирования бизнес-процессов, IDEF3 не имеет жестких синтаксических или семантических ограничений, делающих неудобным описание неполных или нецелостных систем. Кроме того, автор модели (системный аналитик) избавлен от необходимости смешивать свои собственные предположения о функционировании системы с экспертными утверждениями в целях заполнения пробелов в описании предметной области.

Любая IDEF3-диаграмма может содержать [34,35]:

- работы;
- связи;
- перекрестки (соединения);
- объекты ссылок.

Работа (Unit of Work, activity) изображается прямоугольником с прямыми углами, рис.3.7, и имеет имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы. Все стороны работы равнозначны. В каждую работу может входить и выходить ровно по одной стрелке.

Связи предназначены для выделения существенных взаимоотношений между действиями. Все связи в IDEF3 являются однонаправленными, и, хотя стрелка может начинаться или заканчиваться на любой стороне блока,

обозначающего действие, диаграммы IDEF3 обычно организовываются слева направо таким образом, что стрелки начинаются на правой и заканчиваются на левой стороне блоков. В табл.3.14 приведены три возможных типа связей.



Рис.3.7. Изображение и нумерация Работы (Unit of Work, activity) в диаграмме IDEF3.

Таблица 3.14. Типы связей в модели IDEF3

| Изображение | Название | Назначение |
|-------------|---|---|
| → | Временное предшествование (Temporal precedence) | Исходное действие должно завершиться прежде, чем конечное действие сможет начаться |
| ⇒ | Объектный поток (Object flow) | Выход исходного действия является входом конечного действия. Из этого, в частности, следует, что исходное действие должно завершиться прежде, чем конечное действие сможет начаться |
| — — ► | Нечеткое отношение (Relationship) | Вид взаимодействия между исходным и конечным действиями задается аналитиком отдельно для каждого случая использования такого отношения |

Связь типа «Временное предшествование». Отражает, что исходное действие должно полностью завершиться, прежде чем начнется выполнение конечного действия. Связь должна быть поименована таким образом, чтобы человеку, просматривающему модель, была понятна причина ее появления. Во многих случаях завершение одного действия инициирует начало выполнения другого.

Связь типа «Объектный поток». Одной из наиболее часто встречающихся причин использования этого типа связи состоит в том, что некоторый объект, являющийся результатом выполнения исходного действия, необходим для выполнения конечного действия. Такая связь отличается от связи временного предшествования двойным концом обозначающей ее стрелки. Наименования потоковых связей должны четко идентифицировать объект, который передается с их помощью. Временная семантика объектных связей аналогична связям предшествования. Это означает, что порождающее объектную связь исходное действие должно завершиться, прежде чем конечное действие начнет выполняться.

Связь типа «Нечеткое отношение». Используется для выделения отношений между действиями, которые невозможно описать с использованием

предшественных или объектных связей. Значение каждой такой связи должно быть определено, поскольку связи данного типа сами по себе не предполагают никаких ограничений. Одно из применений нечетких отношений – отображение взаимоотношений между параллельно выполняющимися действиями. Название стрелки может быть использовано для описания природы отношения, более подробное объяснение может быть приведено в виде отдельной ссылки. Наиболее часто нечеткие отношения используются для описания специальных случаев связей предшествования, например для описания альтернативных вариантов временного предшествования.

Перекрестки (соединения). Завершение одного действия может инициировать начало выполнения сразу нескольких других действий, или, наоборот, определенное действие может требовать завершения нескольких других действий для начала своего выполнения. Различают перекрестки для слияния (сворачивающие соединения) (Fan-in Junction) и разветвления (разворачивающие соединения) (Fan-out Junction) стрелкою Соединения разбивают или соединяют внутренние потоки и используются для описания ветвления процесса.

Сворачивающие соединения объединяют потоки. Завершение одного или нескольких действий вызывает начало выполнения только одного другого действия.

Разворачивающие соединения используются для разбиения потока. Завершение одного действия вызывает начало выполнения нескольких других.

Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму необходимо указать тип перекрестка.

Если действия, инициируемые разворачивающими соединениями не должны начинать выполняться одновременно, то имеет место *асинхронное соединение*

Если же время начала или окончания параллельно выполняемых действий, инициируемых разворачивающимся соединением, выполняются одновременно, то это *синхронные соединения*.

Синхронное соединение обозначается двумя вертикальными линиями внутри обозначающего его прямоугольника в отличие от одной вертикальной линии в асинхронном соединении.

Классификация возможных типов перекрестков приведена в табл.3.15.






Парность соединений. Все соединения на диаграммах должны быть парными, из чего следует, что любое разворачивающее соединение имеет парное себе сворачивающее. Однако типы соединений вовсе не обязательно должны совпадать.

Синхронное разворачивающее соединение не обязательно должно иметь парное себе сворачивающее соединение. Действительно, начинающиеся одновременно действия вовсе не обязаны оканчиваться одновременно, как это видно из примера с состязаниями. Также возможны ситуации синхронного окончания асинхронно начавшихся действий

Соединения могут комбинироваться для создания более сложных правил ветвления. Однако, комбинации соединений следует использовать с

осторожностью, поскольку перегруженные ветвлением диаграммы могут оказаться сложными для восприятия.

Таблица 3.15. Классификация возможных типов перекрестков

| Обозначение | Наименование | Смысл в случае слияния стрелок (Fan-in Junction) | Смысл в случае разветвления стрелок (Fan-out Junction) |
|--|--------------------|--|---|
|  | Asynchronous AND | Все предшествующие процессы должны быть завершены | Все следующие процессы должны быть запущены |
|  | Synchronous AND | Все предшествующие процессы завершены одновременно | Все следующие процессы запускаются одновременно |
|  | Asynchronous OR | Один или несколько предшествующих процессов должны быть завершены | Один или несколько следующих процессов должны быть запущены |
|  | Synchronous OR | Один или несколько предшествующих процессов завершаются одновременно | Один или несколько следующих процессов запускаются одновременно |
|  | XOR (Exclusive OR) | Только один предшествующий процесс завершен | Только один следующий процесс запускается |

Указатели – специальные символы, которые ссылаются на другие разделы описания процесса. Они выносятся на диаграмму для привлечения внимания читателя к каким-либо важным аспектам модели. В табл.3.16 представлены типы указателей модели IDEF3.

Таблица 3.16. Типы указателей модели IDEF3

| Тип указателя | Назначение |
|---|---|
| ОБЪЕКТ (OBJECT) | Для описания того, что в действии принимает участие какой-либо заслуживающий отдельного внимания объект. |
| ССЫЛКА (GOTO) | Для реализации цикличности выполнения действий. Указатель ССЫЛКА может относиться и к соединению. |
| ЕДИНИЦА ДЕЙСТВИЯ (Unit of Behavior – UOB) | Для помещения на диаграмму дополнительного экземпляра уже существующего действия без заикливания. |
| ЗАМЕТКА (NOTE) | Для документирования любой важной информации общего характера, относящейся к изображенному на диаграммах (в этом смысле ССЫЛКА служит альтернативой методу помещения текстовых заметок непосредственно на диаграммах. |
| УТОЧНЕНИЕ (Elaboration – ELAB) | Для уточнения или более подробного описания изображенного на диаграмме. Обычно используются для описания логики ветвления у соединений |

Указатель изображается на диаграмме в виде прямоугольника, рис.3.8, похожего на изображение действия. Имя указателя обычно включает его тип (например, OBJECT, UOB и т.п.) и идентификатор (имя).

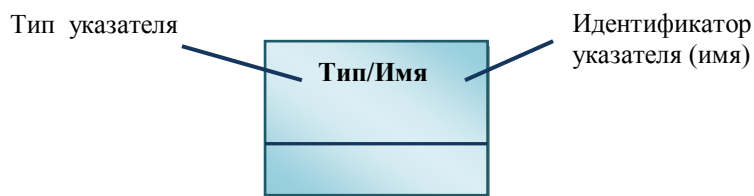


Рис.3.8. Изображение и нумерация Указателя в диаграмме IDEF3.

Действия в IDEF3 могут быть **декомпозированы**, или разложены на составляющие, для более детального анализа. Декомпозировать действие можно несколько раз. Это позволяет документировать альтернативные потоки процесса в одной модели.

Для корректной идентификации действий в модели с множественными декомпозициями схема нумерации действий расширяется и наряду с номерами действия и его родителя включает в себя порядковый номер декомпозиции. Например, в номере действия 1.2.5: 1 – номер родительского действия, 2 – номер декомпозиции, 5 – номер действия.

Номера UOB дочерних диаграмм имеют сквозную нумерацию, т.е., если родительский UOB имеет номер "1", то блоки UOB на его декомпозиции будут соответственно иметь номера "1.1", "1.2" и т.д. Применение принципа декомпозиции в IDEF3 позволяет структурировано описывать процессы с любым требуемым уровнем детализации.

2. Этапы построения IDEF3-диаграммы

На **первом шаге** производится определение сценария, границ моделирования и точки зрения. На этом этапе должны быть задокументированы сценарии и рамки модели для того, чтобы можно было понять цели декомпозиции. Кроме того, если точка зрения моделирования отличается от точки зрения заказчика, она должна быть особенно тщательно задокументирована. Необходимо приготовить список вопросов для проведения интервью.

На **втором шаге** необходимо определить работы и объекты. Обычно уже имеется описание предметной по которому составляется текстовое описание сценария. В дополнение к этому может существовать документация, описывающая интересующие процессы. Из всей этой информации необходимо составить список кандидатов на работы (отглагольные существительные, обозначающие процесс, одиночные или в составе фразы) и кандидатов на объекты (существительные, обозначающие результат выполнения работы), которые необходимы для перечисленных в списке работ.

В некоторых случаях целесообразно создать графическую модель для представления предметной области. Графическая модель может быть также создана после сеанса сбора информации для того, чтобы детали форматирования диаграммы не смущали участников.

На **третьем шаге** осуществляется выстраивание последовательности и согласование. Если диаграмма создается после проведения интервью, то принимаются некоторые решения, относящиеся к иерархии диаграмм, например,

сколько деталей включать в одну диаграмму. Если последовательность и согласование диаграмм неочевидны, может быть проведена еще одна экспертиза для детализации и уточнения информации.

На **четвертом шаге** определяются работы, перекрестки и документирование объектов. IDEF3 позволяет внести информацию в модель различными способами. Например, логика взаимодействия может быть отображена графически в виде комбинации перекрестков. Та же информация может быть отображена в виде объекта ссылки типа ELAB (Elaboration).

Важно учитывать, что модели могут быть реорганизованы, например, для их представления в более презентабельном виде. Выбор формата для презентации часто имеет важное значение для организации модели, поскольку комбинация перекрестков занимает значительное место на диаграмме и использование иерархии перекрестков затрудняет расположение работ на диаграмме.

3. Правила и рекомендации для построения IDEF3-диаграммы

При построении модели в нотации IDEF3 существуют следующие соглашения (правила и рекомендации).

1. На вершине дерева декомпозиции диаграмм должна находиться, либо контекстная диаграмма в нотации IDEF0, с указанием цели моделирования и точки зрения, либо IDEF0 или DFD диаграмма (в случае если IDEF3-диаграммы дополняют модель в нотации и IDEF0 или DFD).

2. Диаграммы должны быть декомпозированы до уровня, на котором присутствуют операции обработки конкретных документов (или совокупности документов).

3. Каждая операция, не имеющая декомпозиции, помечается небольшой диагональной чертой, расположенной в левом верхнем углу прямоугольника, изображающего эту операцию.

4. В случаях сложных диаграмм рекомендуется использовать различные цвета или «уровни» для прямоугольников и стрелок, позволяющие показывать или распечатывать только часть схемы и добиваться её большей наглядности.

5. Диаграммы должны содержать не менее трех и не более 8 операций.

6. Каждая операция имеет свой уникальный номер и имя.

7. Следует обеспечить максимальное расстояние между прямоугольниками и поворотами стрелок, а также между прямоугольниками и пересечениями стрелок для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные стрелки.

8. Дочерние диаграммы (описания и сценарии) должны иметь один вход. Один выход должна иметь дочерняя диаграмма-описание.

9. Рекомендуется стрелки, обозначающие связи направлять либо слева направо, либо сверху вниз.

10. При соединении большого числа прямоугольников необходимо избегать необязательных пересечений стрелок. Следует минимизировать число петель и поворотов каждой стрелки.

11. Связь через потоки объектов должна иметь имя, которое является уникальным.

12. Старшая связь и связи-отношения могут иметь имя, которые также должны быть уникальными. Уникальным именем должны обладать объекты ссылок.

13. При наличии стрелок со сложной топологией целесообразно повторить имя для удобства ее идентификации.

14. Стрелки должны сливаться и разветвляться через перекрестки.

15. Каждому перекрестку присваивается уникальный номер.

16. В ссылках на операции обработки документов должны быть указания на обрабатываемые документы.

Задание на практическое занятие №5

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio)

Исходные данные: контекстная диаграмма (A-0) – модель окружения и диаграмма 1-го уровня (A0), построенные с помощью методологии IDEF0.

1. Используя результаты выполнения предыдущих практических занятий определить список действий и объектов, составляющих моделируемый процесс. Результаты выполнения занести в таблицу 3.17.

Таблица 3.17. Список действий и объектов, составляющих моделируемый процесс

| № действия | Название действия | Объекты принимающие участие в действии (при необходимости) |
|------------|-------------------|--|
| | | |

2. Для каждого действия установить предшествующие действия и определить наличие связи между ними: достаточное, необходимое или необходимое и достаточное. Результаты выполнения занести в таблицу 3.18 (порядок заполнения столбцов указан во второй строке табл.3.18).

Таблица 3.18. Список действий с указанием предшествующих и последующих событий с указанием типа связи

| № предшествующего действия или номера предшествующих действий | тип связи | № действия | тип связи | № последующего действия или номера последующих действий |
|---|---------------------------|------------|---------------------------|---|
| 2 | 3 | 1 | 5 | 4 |
| Действие 1 | Временное предшествование | Действие 2 | Временное предшествование | Действие 3 |
| ... | ... | ... | ... | ... |

3. Установить для каждого действия список, действия из которого должны быть все завершены до начала рассматриваемого действия (соединение «И» (&)).

4. Установить для каждого действия список, действия из которого должны быть завершены до начала данного действия, причем завершение каждого действия списка вызывает начало рассматриваемого действия (эксклюзивное «ИЛИ» (X)).

5. Установить отношение между началом и окончанием связанных соединением действий.

6. Результаты выполнения п. 3,4,5 занести в таблицу 3.19 (порядок заполнения столбцов указан во второй строке табл.3.19).

Таблица 3.19. Список действий с указанием предшествующих и последующих событий с указанием установленных отношений

| № предшествующего действия или номера предшествующих действий | Вид казуального отношения | № действия | Вид казуального отношения | № последующего действия или номера последующих действий |
|---|---------------------------|------------|---------------------------|---|
| 2 | 3 | 1 | 5 | 4 |
| - | - | Действие 1 | & | Действие 2 |
| | | | | ... |
| | | | | Действие n |
| Действие 2 | О | Действие p | О | Действие k |
| ... | | | | |
| Действие n | | | | |
| ... | ... | ... | ... | ... |

Задание для самостоятельного выполнения №5

На основании контекстной диаграммы (А-0) – модели окружения и диаграммы 1-го уровня (А0), построенные с помощью методологии IDEF0, декомпозировать функциональные блоки модели окружения на 1-2 уровня вглубь до потоков, связи с внешними системами и хранилищами с помощью методологии IDEF3 (диаграммы 2-го уровня должны содержать не менее 4-х функциональных блоков, диаграммы 3-го уровня должны содержать не менее 2-х функциональных блоков).

Ввести фиктивные действия, если для разворачивающих действий отсутствуют парные сворачивающие действия.

Содержание части отчета №5 по практическому заданию

1. Список действий, составляющих моделируемый процесс с установленными типами связей, табл.3.17.

2. Список действий с указанием предшествующих и последующих событий с указанием типа связи, табл.3.18.

3. Список действий с указанием предшествующих и последующих событий с указанием установленных отношений, табл.3.19.

4. IDEF3 диаграммы 2-го и последующих уровней.

Контрольные вопросы

1. Что представляет собой модель в нотации IDEF3?
2. Перечислить основные элементы нотации IDEF3.
3. Перечислить типы стрелок в диаграммах IDEF3.
4. Что называется перекрестком?
5. Назвать типы перекрестков.
6. Нумерация объектов в IDEF3
7. Виды взаимодействия параллельных процессов. Примеры.
8. Объектная связь. Примеры.
9. Связь временного предшествования. Примеры.
10. Типы ссылок.
11. Порядок построения IDEF3 диаграмм.
12. Сходство и различие диаграмм IDEF3 и IDEF0 моделей.

Практическое занятие №6

Моделирование, анализ и реорганизация бизнес-процессов (методология BPMN)

Цель:

- изучить общие положения о функциональном моделировании процессов, ориентированном на потоки данных;
- построить диаграмму бизнес-процессов в нотации BPMN.

Время: 2 часа

Краткие теоретические сведения

1. Методология моделирования, анализа и реорганизации бизнес-процессов (BPMN)

Моделирование бизнес-процессов (см. п.1.9.1.8) используется для донесения широкого спектра информации до различных категорий пользователей. Диаграммы бизнес-процессов (BPMN) позволяют описывать сквозные бизнес-процессы, но в то же время помогают читателям быстро понимать процесс и легко ориентироваться в его логике.

Диаграмма, описанная в нотации BPMN [36], представляет собой алгоритм (сценарий) выполнения процесса, а также отображение того, как процесс взаимодействует с другими процессами с точки зрения обмена сообщениями (информацией, документами и другими объектами деятельности). Алгоритм выполнения процесса представляется на диаграмме с помощью графических элементов. Это помогает пользователям быстро понимать логику процесса. Выделяют четыре основные категории элементов:

- объекты потока управления: события, действия и логические операторы;
- соединяющие объекты: поток управления, поток сообщений и ассоциации;
- роли: пулы и дорожки;
- артефакты: данные, группы и текстовые аннотации.

Элементы этих четырёх категорий позволяют строить простейшие диаграммы бизнес-процессов. Для повышения выразительности модели спецификация разрешает создавать новые типы объектов потока управления и артефактов.

В сквозной BPMN-модели можно выделить три типа подмоделей:

- частные (внутренние) бизнес-процессы;
- абстрактные (открытые) бизнес-процессы;
- процессы взаимодействия (глобальные).

Частные бизнес-процессы описывают внутренние процессы и представляют бизнес-процессы в общепринятом понимании (business processes или workflows). При использовании ролей частный бизнес-процесс помещается в

отдельный пул. Поэтому поток управления находится внутри одного пула и не может пересекать его границ. Поток сообщений, напротив, пересекает границы пулов для отображения взаимодействия между разными частными бизнес-процессами.

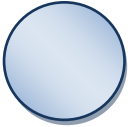

Абстрактные (открытые) бизнес-процессы служат для отображения взаимодействия между двумя частным бизнес-процессами (то есть между двумя участниками взаимодействия) В открытом бизнес-процессе показываются только те действия, которые участвуют в коммуникации с другими процессами. Все другие, «внутренние», действия частного бизнес-процесса не показываются в абстрактном процессе. Таким образом, абстрактный процесс показывает окружающим последовательность событий, с помощью которой можно взаимодействовать с данным бизнес-процессом. Абстрактные процессы помещаются в пулы и могут моделироваться как отдельно, так и внутри большей диаграммы для отображения потока сообщений между действиями абстрактного процесса с другими элементами. Если абстрактный процесс и соответствующий частный процесс находятся в одной диаграмме, то действия, отображённые в обоих процессах, могут быть связаны ассоциациями.

Процессы взаимодействия (глобальные) отображают взаимодействия между двумя и более сущностями. Эти взаимодействия определяются последовательностью действий, обрабатывающих сообщения между участниками. Процессы взаимодействия могут помещаться в пул. Эти процессы могут моделироваться как отдельно, так и внутри большей диаграммы для отображе

Любой процесс, описанный в нотации BPMN, представляет собой последовательное или параллельное выполнение различных действий (операций) с указанием определённых бизнес-правил.

Таблицы 3.18, 3.19 содержат перечень основных графических элементов моделирования, изображенных при помощи графических нотаций.

Таблица 3.18. Основные графические элементы моделирования

| Элемент | Описание | Нотация |
|----------------------------|--|---|
| Событие (Event) | Событие – это то, что происходит в течение бизнес-процесса и оказывает влияние на его ход. Чаще всего событие имеет причину (триггер) или воздействие (результат). Изображается в виде круга со свободным центром, предназначенным для дифференцировки внутренними маркерами различных триггеров или их результатов. Согласно влиянию Событий на ход бизнес-процесса, выделяют три типа: Стартовое событие (Start), Промежуточное событие (Intermediate) и Конечное событие (End). |  |
| Действие (Activity) | Действие – общий термин, обозначающий работу, выполняемую исполнителем. Действия могут быть либо элементарными, либо неэлементарными (составными). Выделяют следующие виды действий, являющихся частью модели Процесса: Процесс (Process). Подпроцесс (Sub-Process) и Задача (Task). Задача и Подпроцесс изображаются в виде прямоугольника с закругленными углами. Процесс |  |









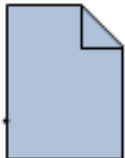


| Элемент | Описание | Нотация |
|---|--|---|
| | либо не имеют границ, либо находятся внутри Пула. | |
| Шлюз (Gateway) | Шлюзы используются для контроля расхождений и схождении потока операций. Таким образом, данный термин подразумевает ветвление, раздвоение, слияние и соединение маршрутов. Внутренние маркеры указывают тип контроля развития бизнес-процесса. |  |
| Обмен сообщениями (Conversation) | Описание действия, характеризующего обмен информацией между участниками (пулами) взаимодействия. |  |

Таблица 3.19. Основные графические элементы моделирования диаграммы бизнес-процесса (BPD)

| Элемент | Описание | Нотация |
|--|---|---|
| Поток операций (Sequence Flow) | Поток операций служит для отображения того порядка, в котором организованы действия Процесса. |  |
| Поток сообщений (Message Flow) | Поток сообщений служит для отображения обмена сообщениями между двумя участниками, готовыми эти сообщения отсылать и принимать. На диаграмме BPMN два отдельно взятых Пула представляют собой двух участников процесса (бизнес-объекты или бизнес-роли). |  |
| Ассоциация (Association) | Ассоциация служит для установления связи между информацией и элементами потока. Текстовые объекты, а также графические объекты, не относящиеся к элементам потока, могут соотноситься с Элементами потока. |  |
| Ссылка на обмен сообщениями (Conversation Link) | Указывает на обмен сообщениями между участниками взаимодействия. |  |
| Пул (Pool) | В BPMN Пул представляет собой Участника Процесса. Пул также может выступать в качестве Зоны ответственности или графического контейнера, отвечающего за разделение определенного набора действий, относящихся к другим Пулам, что обычно встречается в ситуациях типа «бизнес для бизнеса» (B2B). |  |
| Дорожка (Lane) | Дорожка обеспечивает разделения внутреннего пространства Пула как по вертикали, так и по горизонтали. Служит для упорядочивания и категоризации действий. |  |
| Объект данных (Data Object) | Объект данных относится кArteфактам, т.к. не оказывает непосредственного влияния ни на Поток операций, ни на Поток сообщений. Однако Объект данных предоставляет информацию о том, какие действия необходимо выполнить и/или каков результат этих действий. |  |

| Элемент | Описание | Нотация |
|--|--|---|
| Группа (блок, содержащий группу объектов, предназначенных для документирования) (Group) | Группировка объектов, не оказывающих влияния на Поток операций. Такого рода группировка может использоваться в целях составления документации или анализа. Группы также могут использоваться для идентификации операции, распределенной внутри Пула. |  |
| Текстовая аннотация (связана с Ассоциацией) (Text Annotation) | Текстовая аннотация служит в качестве механизма, позволяющего разработчику модели бизнес-процесса вводить дополнительную информацию для тех, кто работает с BPMN диаграммами. |  |

При выполнении процесса могут происходить различные события, оказывающие влияние на ход процесса: старт процесса, его завершение, смена статуса документа, получение сообщения и многое другое. *События* – необязательные элементы, поэтому на диаграмме процесса в нотации BPMN они могут не отображаться.

Все события классифицируются по следующим признакам.

По времени наступления:

– *стартовое событие* – инициирует начало процесса (диаграммы). Из стартового события поток управления может только исходить, а поток сообщений – как входить, так и исходить. На диаграмме процесса, как правило, отображается только одно стартовое событие, но оно может отсутствовать, или их может быть несколько при отображении процесса с пулами, дорожками или развернутыми подпроцессами. Контур события отображается одинарной тонкой линией;

– *конечное событие* – является результатом выполнения процесса. В конечное событие поток управления может только входить, а поток сообщений – как входить, так и исходить. В конечное событие может только входить поток (стрелка). На диаграмме конечное событие, как и стартовое, может быть одно, несколько (даже при отсутствии пулов и дорожек) или ни одного. Контур события отображается одинарной жирной линией;

– *промежуточное событие* – все остальные события, возникающие в ходе выполнения процесса. В промежуточное событие обязательно должен входить и выходить один поток. Исключение составляет граничные (Boundary) события, возникающие и обрабатываемые непосредственно либо в самом начале действия либо в его конце. Такие события отображаются на границе (контуре) действия и у них может быть только либо входящий либо исходящий поток. Контур события отображается двойной тонкой линией.

По возможности прерывания выполнения действия (подпроцесса):

– *непрерывающее событие* – стартовое или промежуточное событие, возникающее в ходе выполнения действия, но инициирующее связанный с событием исходящий поток только после завершения действия. Контур события отображается штриховой линией;

– *прерывающее событие* – событие, возникающее до или после стандартного выполнения действия или требующее его немедленного прекращения в исключительных ситуациях. Например, при отсутствии всей необходимой информации или возникновении ошибки в ходе ее обработки, необходимости выполнения дополнительных действий и т.д. Контур события отображается сплошной линией;

По типу результата действия:

– *событие-инициатор обработки* – стартовое или промежуточное событие, возникшее в результате выполнения действия и требующее его последующей обработки. Отображается незакрашенной иконкой;

– *событие-результат обработки* – промежуточное или конечное событие, возникшее в результате выполнения действия и являющееся итоговым результатом стандартного или нестандартного выполнения процесса. Отображается закрашенной иконкой;

По причине возникновения (триггеру)

- неопределенное (None);
- сообщение (Message);
- таймер (Timer);
- ошибка (Error);
- прерывание, эскалация (Escalation);
- отмена (Cancel);
- компенсация (Compensation);
- условие (Conditional);
- связь (Link);
- сигнал (Signal);
- завершение (Terminate)
- множественное (Multiple);
- параллельно-множественное (Parallel Multiple);

Действие. Процесс, отображаемый в виде диаграммы, представляет собой упорядоченный набор действий, выполняемых с целью получения конкретного результата. Временная последовательность выполнения процессов задается расположением процессов на диаграмме слева-направо (сверху-вниз на вертикальной диаграмме процесса BPMN), а также направлением стрелок у соединяющих элементов.

Различают три основных вида действий и их разновидности:

- ***задача (Task)*** – элементарное (неделимое, атомарное) действие:
 - *сервисная (Service)*. Задача предназначена для оказания услуги, которая может являться как веб-сервисом, так и автоматизированным приложением;
 - *отправка сообщения (Send)*. Задача считается выполненной, если сообщение послано хотя бы один раз;
 - *получение сообщения (Receive)*. Задача считается выполненной, если сообщение получено хотя бы один раз;

- *пользовательская (User)*. Характерная задача, выполняемая исполнителем при содействии других людей или программного обеспечения;
- *ручное исполнение (Manual)*. Характерная задача, выполняемая исполнителем без каких-либо средств автоматизации;
- *бизнес-правило (Business-Rule)*. Задача, технология выполнения которой зависит от текущих обстоятельств и выбирается на основе заданного бизнес-правила;
- *сценарий (Script)*. Задача, порядок выполнения операций которой описан на языке, распознаваемом исполнителем. Обычно используется для задач, выполняемых автоматическими средствами;

– **подпроцесс (Sub-Process)** – составное действие, включающее в себя другие действия, шлюзы, события и потоки операций. Части подпроцесса могут непосредственно отображены на диаграмме внутри символа действия или вынесены на отдельную диаграмму декомпозиции. Во втором случае на родительской диаграмме в центре нижнего края действия (подпроцесса) отображается символ + . Кроме стандартных подпроцессов, имеется еще две специфические его разновидности:

- *событийный подпроцесс (Event Sub-Process)*. Запускается каждый раз, когда происходит одно из стартовых событий. На диаграмме событийный подпроцесс не связан с другими действиями потоками операций. Контур подпроцесса отображается точками;
- *транзакция (Transaction)*. Действие, состоящее из составных операций, удачное завершение (получение конкретного положительного результата) которого возможно при удачном завершении всех его составляющих. В случае возникновения проблем при выполнении подпроцесса (невозможности выполнения одной из операций или высокой вероятности ее некорректного выполнения) результаты предыдущих операций отменяются (событие отмена) или компенсируются (событие компенсация). Контур подпроцесса отображается двойной сплошной линией;

– **вызов (Call)**. Позволяет включать в состав диаграммы повторно используемые задачи и подпроцессы. На диаграмме выделяется жирным контуром.

Дополнительные особенности реализации или выполнения действия могут быть указаны с помощью маркеров, отображаемых у нижнего края символа:

– *цикл (Loop)*. Действие выполняется в цикле с пред- (while) или пост- (repeat-until) условием;

– *многоэкземпляльность (Multi-Instance)*. Параллельное или последовательное выполнение нескольких экземпляров однотипных действий. При последовательном выполнении действие можно рассматривать как цикл с параметром (for);

– *компенсация (Compensation)*. Действие выполняется взамен стандартного при невозможности его удачного завершения;

– *настраиваемый подпроцесс (Ad-Hoc)*. Указывается только для подпроцессов. Конкретный состав и последовательность входящих в него действий определяется исполнителем в процессе его выполнения.

В общем случае для действия может быть указано несколько маркеров.

Шлюз. Шлюз предназначен для указания специфики пропуска потока операций по альтернативным или параллельным ветвям. Шлюз может не иметь входящих или исходящих потоков, но должен иметь, как минимум, два и более либо входящих либо исходящих потока. Тип шлюза задается маркером, указываемым внутри его символа:

– *эксклюзивный (Exclusive, XOR – исключаящее ИЛИ)*. Предназначен для разделения потока операций на несколько альтернативных маршрутов, т.е. в ходе выполнения процесса может быть активирован только один из предложенных маршрутов. Условия пропуска по исходящему маршруту задается рядом с соответствующей линией в виде логического выражения;

– *неэксклюзивный (Inclusive, OR – логическое ИЛИ)*. Предназначен для разделения потока операций на несколько маршрутов, каждый из которых активируется при условии истинности связанного с ним логического выражения. Таким образом, при выполнении процесса может быть выбрано сразу несколько маршрутов, в т.ч. и ни одного в случае ложности всех выражений;

– *комплексный (Complex)*. Аналогичен неэксклюзивному шлюзу. Отличие заключается в том, что с ним связано одно выражение, которое определяет, какие из потоков операций будут активированы;

– *параллельный (Parallel, AND – логическое И)*. Предназначен для слияния/ветвления одновременно (параллельно) выполняемых потоков операций;

– *эксклюзивный, основанный на событиях (Exclusive Event-Based)*. Предназначен для разделения потока операций на несколько альтернативных маршрутов. Единственный маршрут, по которому будет продолжен процесс, выбирается не на основе логического выражения, а в зависимости от произошедших событий, которые указываются по соответствующему маршруту;

– *эксклюзивный, основанный на событиях, запускающий процесс (Exclusive Event-Based Gateway to start a Process)*. Аналогичен предыдущему, но используется в качестве начального символа процесса (подпроцесса). Не имеет входящих потоков;

– *параллельный, основанный на событиях, запускающий процесс (Parallel Event-Based Gateway to start a Process)*. Аналогичен предыдущему, но возможна активация сразу нескольких маршрутов в случае срабатывания событий, с которыми они связаны. Возможно асинхронное выполнение маршрутов (связанных потоков операций и действий). Т.е. после активации и начала выполнения одного из маршрутов, другие маршруты тоже могут быть активированы и выполнены, пока не наступил момент завершения процесса (подпроцесса). Не имеет входящих потоков.

Объект данных. С помощью дополнительных маркеров на диаграмме может быть показана специфика использования и содержания данных:

- *входные данные (Data Inputs)*. Исходные ТМЦ или информация для выполнения действий. Отображается у верхнего края символа;
- *выходные данные (Data Outputs)*. Результат действия. Отображается у верхнего края символа;
- *набор данных (Data Collection)*. Коллекция или массив однотипных данных. Отображается у нижнего края символа.

Связь между объектом данных и действиями отображается с помощью ассоциации.

Потоки операций. В дополнение к стандартному изображению потока операций, на диаграмме могут быть указаны специфические потоки:

- *условный поток операций (Conditional Sequence Flow)*. Используется при ветвлении потоков. Обычно отображается исходящим из действия, чтобы не отображать на диаграмме шлюз. Условия активации потока задается рядом в виде логического выражения;

- *поток операций по умолчанию (Default Sequence Flow)*. Используется при ветвлении потоков. Может исходить из действия или шлюза. Не связан ни с каким логическим выражением. Поток по умолчанию активируется, если активация других потоков в соответствии с их логическими выражениями или событиями невозможна.

Чтение процесса всегда начинается со **Стартового события** (зеленого круга).

Стартовое событие указывает на то, в какой точке берет начало тот или иной процесс. В контексте потока операций Стартовое событие является начальной точкой в процессе, это означает, что никакой входящий поток операций не может быть соединен со стартовым событием.

Стартовое событие в нотации BPMN изображается в виде круга со свободным центром.

Далее от *Стартового события* выполнение процесса идет по линиям (*Поток операций*) до **Конечного события** (красный круг), их может быть несколько.

Конечное событие указывает на то, в какой точке завершается тот или иной процесс. В контексте *Потока операций* *Конечное событие* завершает ход *Процесса*, это означает, что никакой *Исходящий поток* операций не может быть соединен с *Конечным событием*.

Конечное событие представляет собой круг, выполненный одиночной, жирной линией. Толщина линии должна быть жирной настолько, чтобы без труда можно было отличить Конечное событие от Стартового.

Вся логика работы (ход) процесса выражается во всевозможных элементах, расположенных между Стартовым и Конечным событием. Основным элементом, отражающим деятельность, выполняемую внутри процесса, являются **Действия**. *Действия* – это точки выполнения работ в ходе *Процесса*. Они относятся к выполняемым элементам *Процесса* BPMN.

Действие может быть как **элементарным**, так и **неэлементарным** (составным).

Элементарное Действие выражается в выполнении одной единственной *Задачи*. Графически *Задача* изображается в виде прямоугольника с закругленными углами (самой распространённой *Задачей* является типичная для технологического процесса задача, где человек участвует в качестве исполнителя. Такие *Задачи* называются *Пользовательскими*).

Для контроля расхождений и схождений потока операций в рамках процесса используются *Шлюзы (Условия)*. Термин шлюз подразумевает пропускное устройство, которое либо позволяет осуществлять переход через шлюз, либо нет.

Графический элемент Шлюза представляет собой небольшой ромб, используемый во многих нотациях схем бизнес-процессов для изображения ветвления и знакомый большинству инструментов моделирования. Фактически *Шлюз* – есть совокупность входов и выходов.

2. Этапы построения BPMN -диаграммы

На *первом шаге* создается упрощенная модель процесса. Для этого требуется построить (или использовать уже имеющуюся) диаграмму процесса, декомпозированную на задачи, которая размещается в основном пуле.

На *втором шаге* производится распределение задач между действующими лицами (есть определяются роли) и указывается, в какой последовательности эти задачи выполняются. Для этого добавляются дополнительные элементы – *пулы* и *дорожки*.

На *третьем шаге* проводится соединение элементов потоками операций и включаются элементы потока управления, соединения и бизнес-элементы, то есть необходимо связать задачи потоками операций и показать логику их взаимодействия при помощи *шлюзов*.

На *четвертом шаге* на схеме производится размещение:

- документов (с точки зрения управления процессами, документ – это информация на любом информационном носителе – бумажный документ, электронное письмо, доклад, презентация и т.д.);
- программ и баз данных;
- инструментов и материалов;
- показателей эффективности в процессе.

3. Правила и рекомендации для построения BPMN -диаграмм

Процесс моделирования процессов с помощью BPMN *подчиняется классическим принципам моделирования: декомпозиции и иерархического упорядочивания*. Декомпозиция, с отображением на отдельных диаграммах, выполняется для участников (пулов) и отдельных подпроцессов, подобно работам на диаграммах IDEF0 или предопределенным процессам на блок-схемах.

Кроме того, следует придерживаться следующих правил и рекомендации при построения BPMN-диаграмм.

1. Несмотря на тот факт, что события – необязательные элементы на диаграммах, рекомендуется отображать начальные и конечные события. У одного процесса (пула, дорожки, развернутого подпроцесса) должно быть только одно начальное событие, но может быть несколько конечных событий.

2. На диаграмме не должны присутствовать элементы без единой связи.

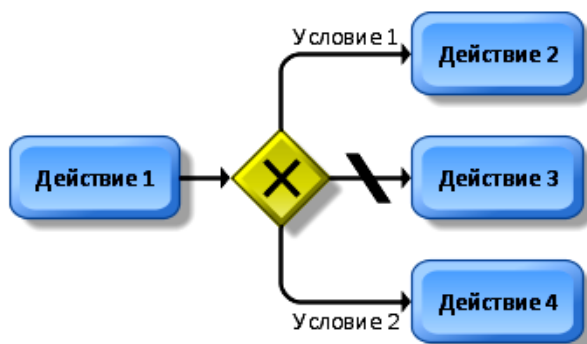
3. Допускается (в отличие от ЕРС-диаграмм) последовательное следование нескольких событий или процессов в подряд.

4. Каждый шлюз слияния должен обладать минимум двумя входящими связями, шлюз ветвления – минимум двумя исходящими.

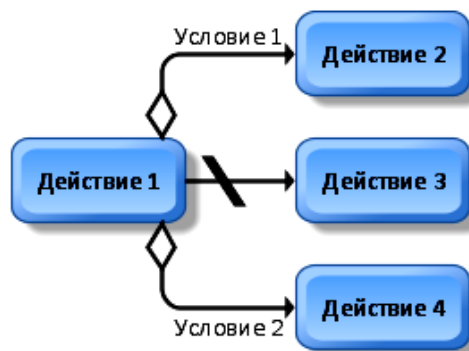
5. Ветвление на альтернативные потоки по логическим выражениям («исключающее ИЛИ» или логическое «ИЛИ») можно отобразить через соответствующий шлюз (эксклюзивный, неэксклюзивный или комплексный) или с использованием специфических потоков операций, рис.3.9.

6. Ветвление на альтернативные потоки в зависимости от произошедших событий можно отобразить через эксклюзивный шлюз, основанный на событиях, или с использованием граничных событий, рис.3.10.

7. Шлюз, разветвляющий ветки, и шлюз, объединяющий эти ветки, должны совпадать. Допускается также ситуация, когда шлюз ветвления «И», шлюз объединения – «ИЛИ», рис.3.11.

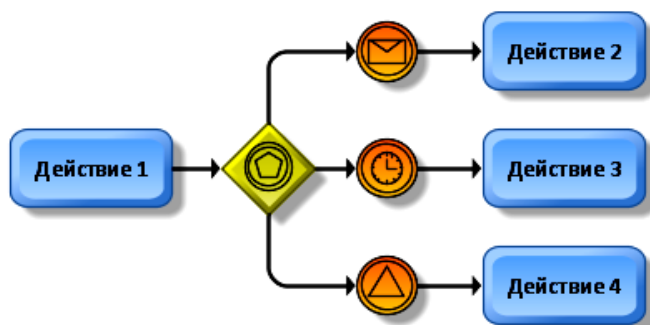


а) ветвление с использованием шлюза

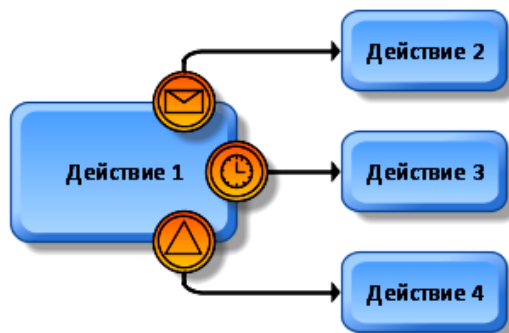


б) ветвление с использованием потоков

Рис.3.9. Примеры ветвления на альтернативные потоки по логическим выражениям.



а) ветвление с использованием шлюза



б) ветвление с использованием граничных событий

Рис.3.10. Примеры ветвления на альтернативные потоки в зависимости от произошедших событий.

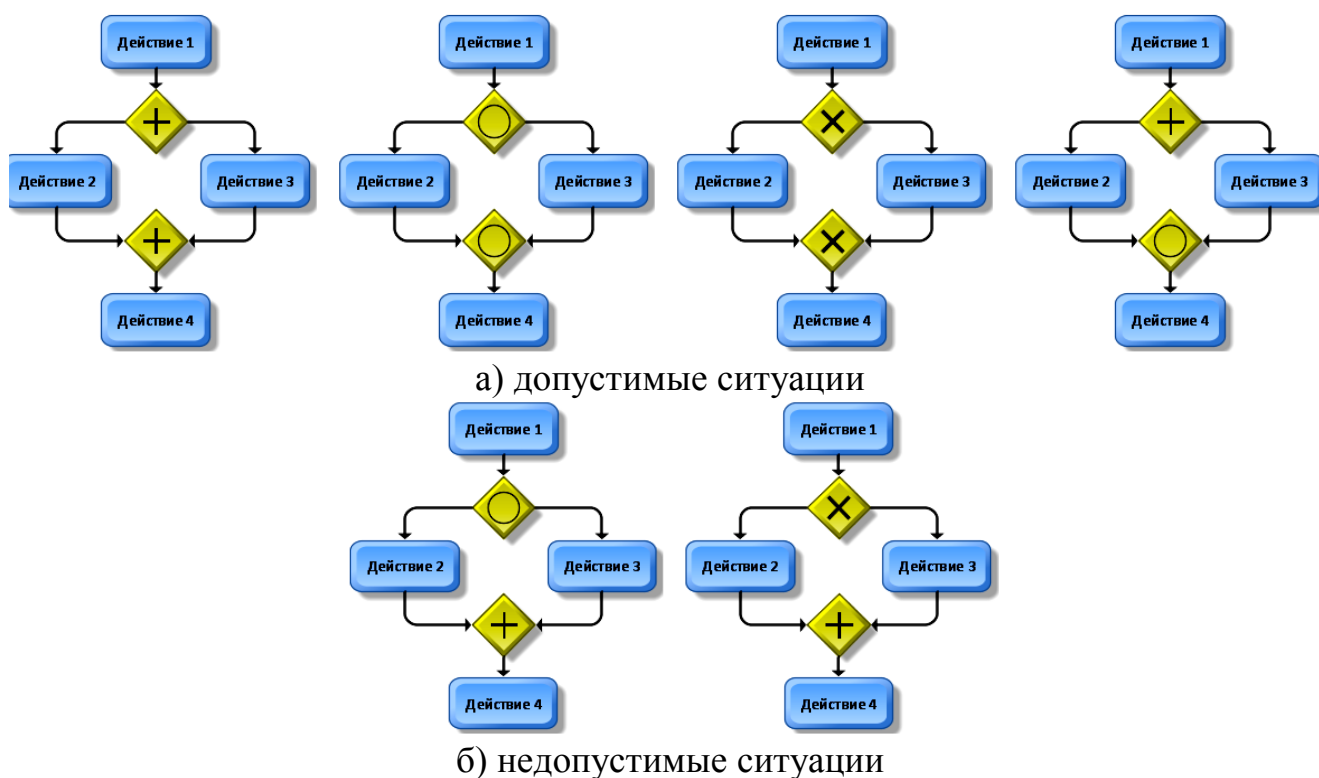


Рис.3.11. Примеры допустимого и недопустимого использования шлюзов.

8. Количество пересечений линий следует минимизировать. При этом считается, что пересекающиеся линии не имеют логической связи друг с другом. Другими словами, потоки в местах пересечений не меняют своего направления.

Задание на практическое занятие №6

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio).

1. Построить простую модель процесса.
2. Произвести разделение задач между участниками процесса.
3. Определить объекты данных (документы, программы и баз данных, инструменты и материалы) и показатели эффективности необходимых или получающихся в ходе выполнения задачи, по п.1, 2 и 3 составить табл.3.20.

Таблица 3.20. Список задач, действующих лиц, объектов данных и показателей эффективности

| № задачи | Название задачи | № и список действий составляющих решение задачи | Участник, осуществляющий решение задачи | Объекты данных | Показатели эффективности |
|----------|-----------------|---|---|----------------|--------------------------|
| 1 | Задача 1 | Действие 1.1 | | | |
| | | | | | |
| | | Действие n.1 | | | |
| ... | ... | ... | ... | | |

4. Построить упрощенную модель бизнес процесса.

Задание для самостоятельного выполнения №6

Построить усложненную модель бизнес-процесса (BPMN-диаграмму):

1. Соединить элементы потока управления потоками операций и включить элементы потока управления, соединения и бизнес-элементы, то есть необходимо связать задачи потоками операций и показать логику их взаимодействия при помощи *иллюзов*.
2. Произвести размещение:
 - документов (с точки зрения управления процессами, документ – это информация на любом информационном носителе – бумажный документ, электронное письмо, доклад, презентация и т.д.);
 - программ и баз данных;
 - инструментов и материалов;
 - показателей эффективности в процессе.

Содержание части отчета №6 по практическому заданию

1. Простая модель процесса.
2. Список задач, действующих лиц, объектов данных и показателей эффективности, табл.3.20.
3. Усложненную модель бизнес-процесса (BPMN-диаграмма).

Контрольные вопросы

1. Что такое BPMN? Перечислить особенности данной нотации.
2. Перечислить основные категории элементов BPMN-модели.
3. Перечислить объекты потока управления.
4. Дать определение процесса, подпроцесса, задачи.
5. В чем отличие между задачей и событием?
6. Дать определение бизнес-элемента.
7. Для чего в модели используются ресурсы?
8. Чем отличаются ресурсы и бизнес-элементы?

Практическое занятие №7

Модельно-ориентированное проектирование систем

Цель:

- изучить общие положения модельно-ориентированного проектирования;
- построить и описать модель процесса.

Время: 2 часа

Краткие теоретические сведения

1. Модельно-ориентированное проектирование (МОП)

Как уже отмечалось (см. п.1.9.2) модельно-ориентированное проектирование (МОП) – это математический и визуальный метод решения задач, связанных с проектированием систем управления, обработки сигналов и связи и т.д. ОП является методологией, применяемой при разработке встроенного программного обеспечения.

МОП определяет общую структуру взаимодействия в процессе проектирования, эффективно реализуя V-образный цикл разработки.

2. Этапы МОП

На *первом шаге* осуществляется построение модели объекта. Построение модели может быть эмпирическим и теоретическим. При эмпирическом построении модели используются такие методы, как идентификация системы. При идентификации системы собираются и обрабатываются исходные данные, полученные от реальной системы, и некоторый алгоритм используется для определения математической модели объекта. Перед построением системы управления модель может быть использована для анализа и построения различных симуляторов. При теоретическом моделировании строятся блок-схемы модели, которые реализуют известные дифференциально-алгебраические уравнения, описывающие динамику объекта. К этому типу относится физическое моделирование, где модель создается с помощью соединяющихся блоков, представляющих собой физические элементы, из которых фактически состоит модель.

На *втором шаге* производится анализ и построение системы управления. Математическая модель, сконструированная на первом шаге, используется для определения динамических характеристик модели объекта. На основе этих характеристик строится система управления.

На *третьем шаге* осуществляется оффлайн-моделирование и моделирование в реальном времени. Время отклика динамической системы на входные данные, изменяющиеся во времени, исследуется с помощью симуляции модели в виде простой линейной стационарной системы или нелинейной

системы. Симуляция позволяет немедленно найти характеристики модели, требования, накладываемые на неё, и ошибки построения до начала проектирования. Моделирование в реальном времени может быть осуществлено с помощью автоматической генерации кода системы управления, построенной на втором шаге. Этот регулятор может быть запущен на специальном компьютере, управляющем работой объекта в реальном времени. Если прототип объекта отсутствует или тестирование на прототипе опасно или дорого, код прототипа может автоматически генерироваться из модели объекта и запускаться на специальном компьютере, работающем в реальном времени и соединенном с целевым процессором с меняющимся кодом управления. Таким образом, система управления может быть протестирована в реальном времени на модели объекта.

На **четвертом шаге** производится реализация регулятора. В идеале это делается с помощью автоматической генерации кода из системы управления, полученной вторым шагом. Маловероятно, что система управления будет работать в реальной системе так же хорошо, как это было при моделировании, поэтому итерационный процесс отладки осуществляется на основе анализа результатов на фактическом объекте и обновления модели регулятора. Инструменты МОП позволяют выполнить все эти итерационные шаги в единой визуальной среде.

Задание на практическое занятие №7

Замечание: все построения, диаграмм, схем, блок-схем проводить средствами MS Office Visio или Dia (альтернатива коммерческой MS Office Visio).

1. Провести выбор процесса для моделирования (по согласованию с ведущим преподавателем).

2. Провести формулировку задачи. Формулировка задачи является частью постановки задачи. В этом разделе описываются:

- цель, назначение, организационно-техническая сущность задачи и обоснование целесообразности ее решения автоматизированным способом;
 - перечень и характеристика управляемых объектов;
 - описание назначения выходной информации;
 - периодичность решения и ограничения по срокам выдачи информации;
 - требования к организации сбора и передачи в обработку входной информации, к порядку ее контроля и корректировки, срокам ее поступления;
 - требования к составу и содержанию информационной базы;
 - условия, при которых прекращается решение задачи;
 - связи данной задачи (комплекса задач) с другими задачами (комплексами задач);
 - должности лиц и / или наименования подразделений, определяющие условия и временные характеристики конкретного решения задачи;
 - распределение функций между персоналом и техническими средствами при различных ситуациях в решении задачи.
3. Определить состав исходных данных для моделирования, дать описание.

4. Определить состав выходных данных. Должно быть отражено, что и в каком виде должно быть получено в результате решения задачи, перечень и описание выходных сообщений. Описание выходных документов представляется в виде, наилучшим образом удовлетворяющей требованиям получателей.

5. Дать математическое описание процесса. Математический аппарат, используемый для решения задачи: если задача не имеет математической формулировки ее решения, используется описание логики последовательных действий в виде выполняемых функций обработки информации по задаче. Математическая или логическая модель решения задачи описывается с достаточной степенью детализации, чтобы в дальнейшем по модели можно было составить алгоритм и программу решения задачи. Для задач, имеющих экономико-математическое описание, выбирается метод решения, который должен обеспечить: необходимую точность расчетов; эффективное решение задачи; программную поддержку (методо-ориентированные ППП, стандартные подпрограммы, встроенные функции и т.п.). Операндами математических формул являются идентификаторы реквизитов входной и выходной информации.

Задание для самостоятельного выполнения №7

Представить алгоритм решения задачи. Алгоритм отражает последовательность и логику выполнения операций обработки информации, способа формирования результатов решения с указанием последовательности счета, расчетных и / или логических формул. Алгоритм представить в виде блок-схемы.

При наличии функционально выделенных подсистем представить отдельные блок-схемы и их описания. Все элементы блок-схем должны сопровождаться необходимыми пояснениями.

В блок-схеме алгоритма указывается, какая информация (документы, файлы и т.д.) используется на каждом этапе решения задачи и какая информация при этом получается.

Содержание части отчета №7 по практическому заданию

1. Детальное описание задачи моделирования.
2. Структура входных и выходных данных.
3. Математическая модель выбранного процесса.
4. Результирующая модель процесса в виде блок-схемы.

Контрольные вопросы

1. В чем суть модельно-ориентированного проектирования (МОП).
2. Перечислить основные этапы модельно-ориентированного проектирования.

Раздел 4

Инструктивно-методические указания по выполнению лабораторных работ

4.1. Лабораторный практикум

Лабораторная работа №1

Инструментальные средства (CASE-средства) планирования и управления проектами

Цель:

- изучить автоматизированные средства планирования и управления проектами;
- осуществить выбор и применение инструментального средства для планирования и управления проектом.

Время: 4 часа

Краткие теоретические сведения

1. Инструментальные средства планирования и управления проектами

В настоящее время разработаны специализированные программы, обеспечивающие поддержку процессов управления проектами.

Программное обеспечение для управления проектами (Project Management Software) – класс компьютерных программ, разработанных специально для автоматизации процессов управления проектами.

В особую группу обычно выделяют системы календарного планирования и контроля проектов.

Среди проприетарных (несвободных) инструментальных средств управления проектами на российском рынке представлены пакеты, сильно различающиеся своими функциональными возможностями и ценой. Этот рынок можно условно подразделить на 2 основные группы – недорогие пакеты (до 1000 долларов), ориентированные на начинающих или непрофессиональных менеджеров, и более дорогие профессиональные пакеты (до 15000 долларов).

К недорогим средствам можно отнести американские пакеты Microsoft Project, Time Line, CA-SuperProject, SureTrak. Разработчики этих программ особое внимание уделяют легкости использования и обучения.

Из профессиональных пакетов представлены: российский пакет Spider Project и американские Artemis Schedule Publisher, Primavera Project Planner, Open Plan, Artemis Project View. Project Expert. Эти пакеты более ориентированы на широту функциональных возможностей управления.

2. Кроссплатформенное программное обеспечение для планирования проектов OpenProj (свободное ПО)

OpenProj – бесплатный аналог Microsoft Project, предназначенный для планирования проектов (<http://sourceforge.net/projects/openproj/> или <http://openproj.ru.uptodown.com/> – свободное скачивание).

Программа включает все необходимые функции:

- диаграмма Ганта, рис.4.1;
- PERT-диаграмма (сетевой график), рис.4.2;
- распределение ресурсов, рис.4.3;
- отчёты;
- поддерживает импорт/экспорт документов Microsoft Project, рис.2.5.

Программа существует в двух вариациях: платная, для совместного пользования и бесплатная, а именно OpenProj где доступно использование продукта только на своём персональном компьютере.

Основные окна программы:

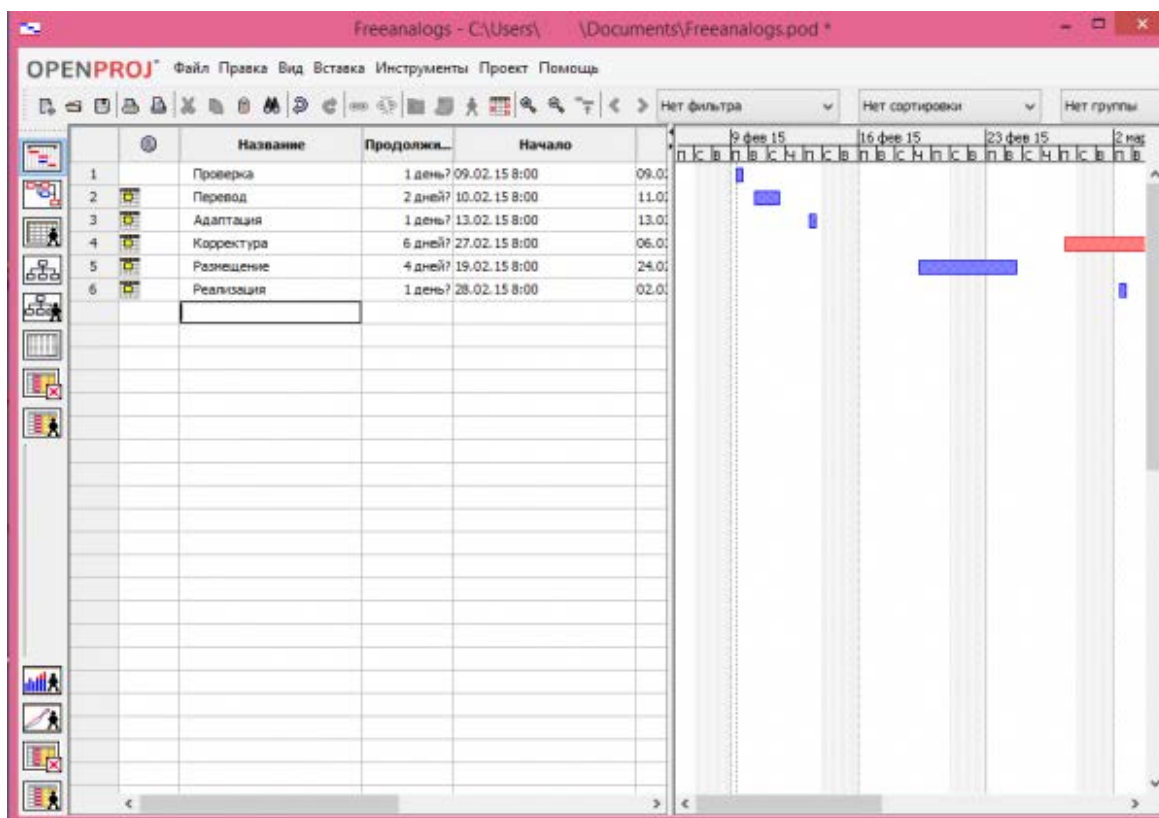


Рис.4.1. Диаграмма Ганта.

Задание на лабораторную работу №1

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №1 осуществить реализацию этапа планирования проекта информационной системы в рамках заданных в варианте предметной области, модели жизненного цикла и методологии проектирования, используя CASE-средство OpenProj.

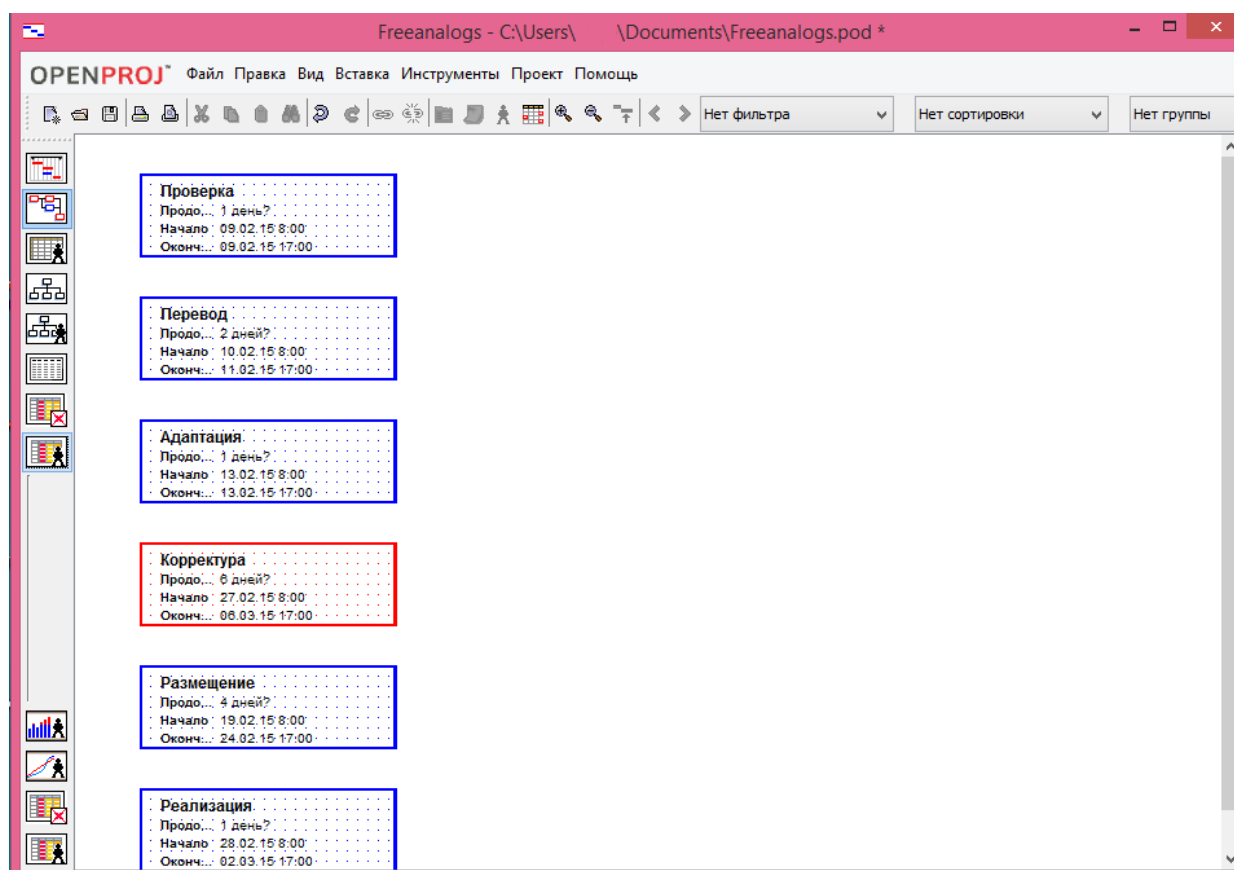


Рис.4.2. Сетевой график.

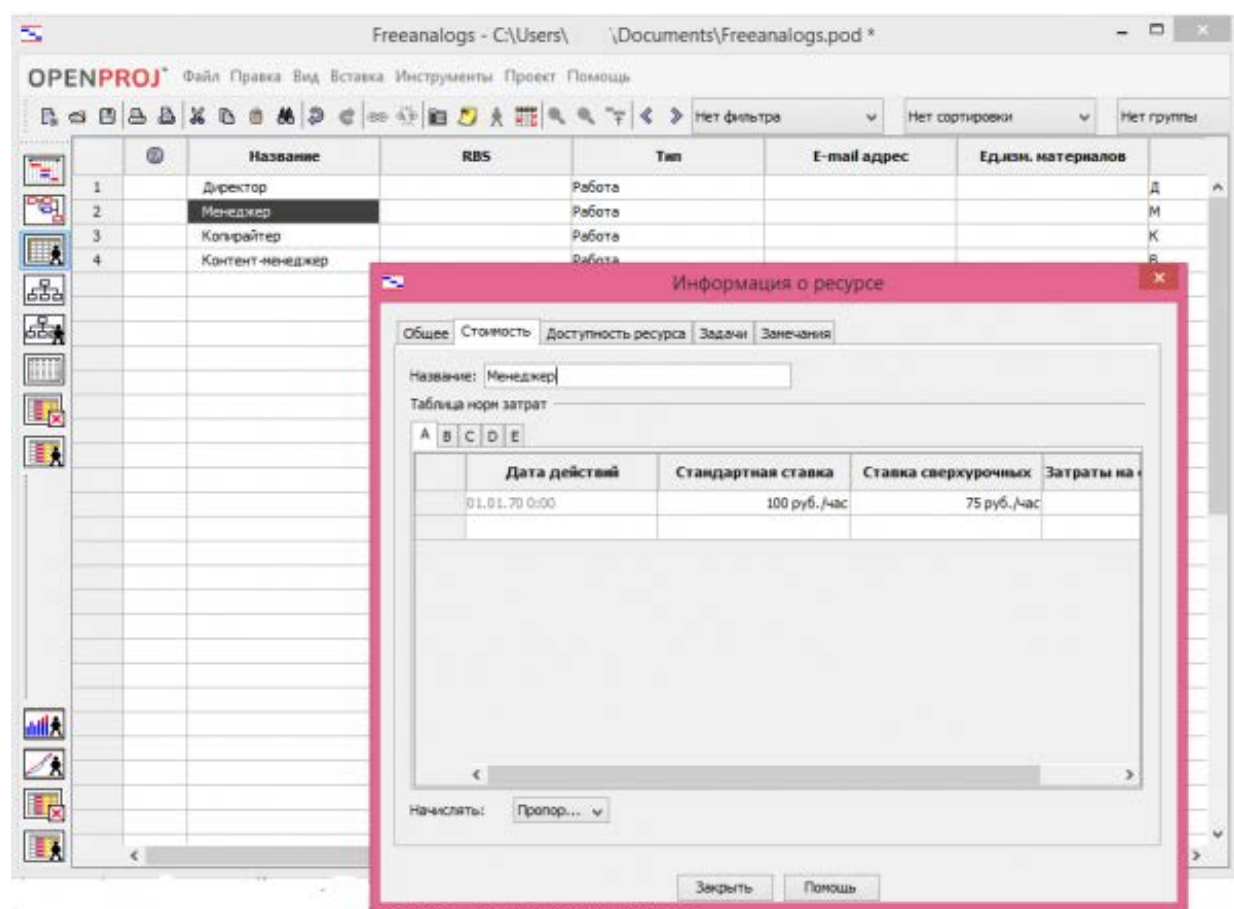


Рис.4.3. Ресурсы.

Порядок выполнения лабораторной работы №1

1. Провести планирование проекта:
 - определить цели проекта;
 - осуществить постановку задачи на проектирование ИС;
 - построить диаграмму жизненного цикла проекта (в соответствии с вариантом).
2. Определить перечень стандартов, в соответствии с которым будет проводиться проектирование информационной системы.
3. Провести анализ проприетарного и свободного программного обеспечения (инструментальных средств) (включая, входящее в состав программных сред, обеспечивающих поддержку полного жизненного цикла), используемого для планирования и управления проектами.
4. Обосновать и осуществить выбор инструментального средства, которое будет использоваться для выполнения задания.
5. Изучить интерфейс и основной функционал выбранного OpenProj [54].
6. С помощью OpenProj разработать график, включающий ресурсы и сроки (этапы) проведения работ. Ресурсы включают персонал, технические средства, ПО и финансирование.
7. С помощью OpenProj сгенерировать все возможные отчеты.

Содержание части отчета №1 по лабораторной работе

1. Сравнительный анализ автоматизированных средств, предназначенных для планирования и управления проектами. Результаты оформить в виде табл.4.1.

Таблица 4.1. Сравнительная характеристика CASE-средств планирования и управления проектами.

| Название Функционал | название CASE-средства1 | название CASE-средства2 | ... | название CASE-средства N |
|---|----------------------------|----------------------------|-----|-----------------------------|
| функция (свойство, решаемая задача)* 1 | — (нет) | +(да) | ... | +(да) |
| ... | ... | ... | ... | ... |
| функция (свойство, решаемая задача)* N | +(да) | +(да) | ... | +(да) |

*функции для решения задач планирования и управления проектами.

2. Краткое описание функционала CASE-средства, используемого в работе.
3. Диаграмма Ганта.
4. PERT-диаграмма (сетевой график).
5. Распределение ресурсов.
6. Сформированные отчеты.
7. Выводы.

Лабораторная работа №2

Инструментальные средства (CASE-средства) поддержки методологий функционального моделирования потоков данных (методология DFD)

Цель:

- изучить автоматизированные средства моделирования потоков данных и потоков работ;
- осуществить выбор и применение инструментального средства для функционального моделирования потоков данных (диаграммы DFD).

Время: 4 часа

Краткие теоретические сведения

1. Инструментальные средства поддержки методологии DFD

Для решения задачи функционального моделирования на базе структурного анализа традиционно применяются два типа моделей: SADT-диаграммы и DFD-диаграммы потоков данных. В случае наличия в моделируемой системе программной/программируемой части (т. е. практически всегда) предпочтение, как правило, отдается DFD по следующим соображениям:

- DFD с самого начала создавались как средство проектирования программных систем (тогда как SADT – как средство проектирования систем вообще) и имеют более богатый набор элементов, адекватно отражающих их специфику (например, хранилища данных являются прообразами файлов или баз данных).

- наличие мини-спецификаций DFD-процессов нижнего уровня позволяет преодолеть логическую незавершенность SADT (а именно обрыв модели на некотором достаточно низком уровне, когда дальнейшая ее детализация становится бессмысленной) и построить полную функциональную спецификацию разрабатываемой системы.

- существуют (и поддерживаются рядом CASE-пакетов) алгоритмы автоматического преобразования иерархии DFD в структурные карты, демонстрирующие межмодульные и внутримодульные связи, а также иерархию модулей, что в совокупности с мини-спецификациями является завершенным заданием для программиста.

- в части автоматизированной поддержки моделей приблизительно 85-90% существующих CASE-пакетов поддерживают DFD и лишь 2-3% – SADT.

В табл.4.2 приведен перечень проприетарных (несвободных) инструментальных средств на российском рынке, поддерживающих DFD, и основные составляющие функциональных моделей.

Таблица 4.2. Пакеты, поддерживающие DFD

| Название | Нотация DFD | Мини-спецификации | Поведение |
|---------------------------|---------------------|-------------------|------------------------|
| Ramus | Йордан | | упр. потоки и процессы |
| CASE Аналитик | Гейн-Карсон | структ. язык | упр. потоки и процессы |
| CASE/4/0 | Йордан (расшир.) | - | Йорд-Мелпор (с STD) |
| Designer/2000 | Гейн-Карсон | - | - |
| EasyCASE | Гейн-Карсон. Йодан | структ. язык | Йорд-Мелпор (с STD) |
| I-CASE Yourdon | Йордан | 3GL | STD |
| Prokit *WORKBENCH | Гейн-Карсон | - | - |
| S-Designor | Гейн-Карсон, Йордан | - | - |
| SILVERRUN | произвольная | - | упр. потоки и процессы |
| Visible Analyst Workbench | Гейн-Карсон, Йордан | - | - |

2. Кроссплатформенная система моделирования и анализа бизнес-процессов Ramus Educational (условно свободное ПО)

Ramus Educational – это бесплатный аналог Ramus [56]. Ramus Education может быть использован для создания диаграмм в формате IDEF0 и DFD, рис.4.4,4.5. Ramus Education использует формат файлов полностью совместимый с форматом файла коммерческой версии Ramus. Как и Ramus, Ramus Educational поддерживает импорт/экспорт файлов в формат IDL, таким образом, реализуя частичную совместимость с подобными программами (например, с СА Erwin Process Modeler). Лицензия Ramus Educational запрещает его использование в коммерческих целях.

Ramus Educational доступен только в локальном варианте, и ограничен по функциональности. Для свободного скачивания с официального сайта доступен Ramus Educational (http://ramussoftware.com/index.php?option=com_docman&task=cat_view&gid=15&Itemid=10).

Перечень основных ограничений по сравнению с коммерческой локальной версией:

- ограничен перечень доступных атрибутов классификаторов;
- отсутствует функциональность для работы с матричными проекциями классификаторов;
- отсутствует редактор отчётов;
- отсутствует навигатор по модели.

Тем не менее, Ramus Educational поддерживает единый формат файлов с локальной версией Ramus. Таким образом, файл созданный в Ramus Educational можно редактировать в локальной версии Ramus и наоборот (за исключением атрибутов поддерживаемых только в локальной версии Ramus).

Также имеется возможность импорта/экспорта файлов в формат IDL BPWin. Таким образом, обеспечивается частичная совместимость с СА ERwin Process Modeler (в части графических моделей IDEF0).

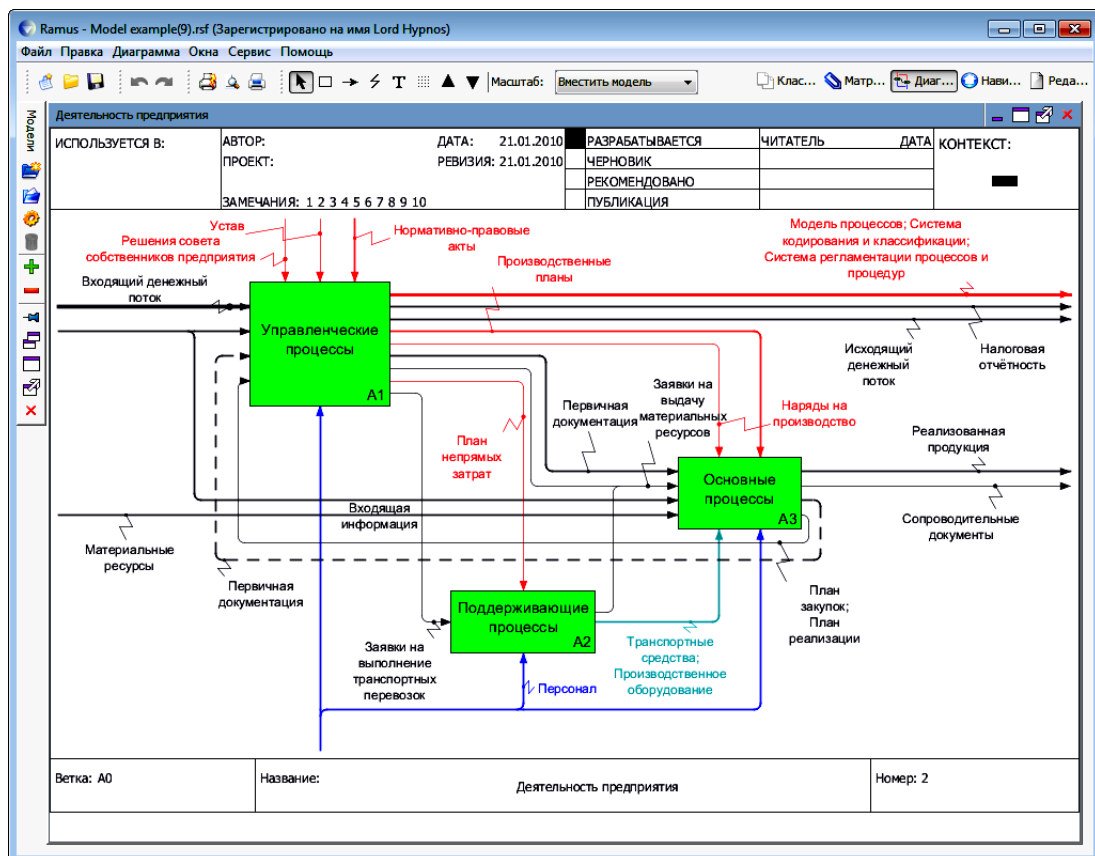


Рис.4.4. Область построения диаграмм Ramus Educational.

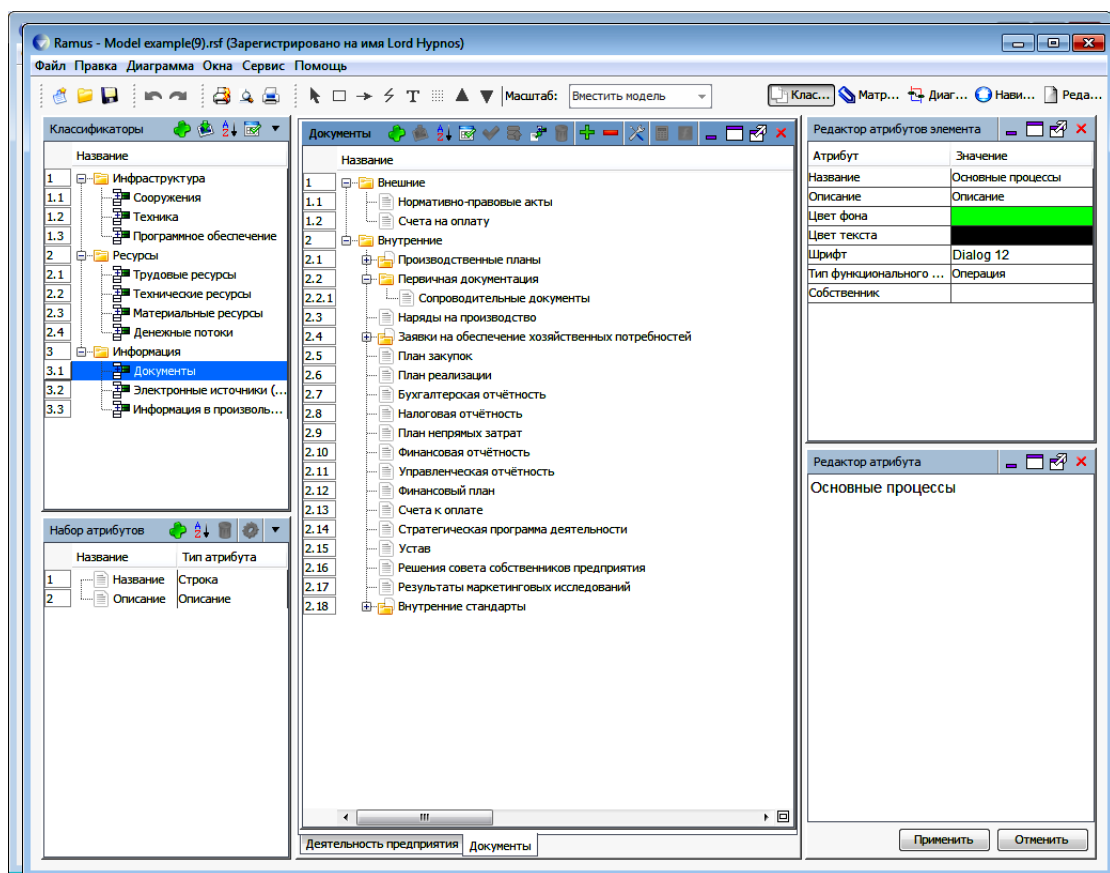


Рис.4.5. Область документирования процессов в Ramus Educational.

Задание на лабораторную работу №2

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №2 выполнить построение DFD-диаграммы при помощи CASE-средства Ramus Educational.

Порядок выполнения лабораторной работы №2

1. Исследовать доступный функционал кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational [57].
2. Осуществить построение DFD-диаграммы в кроссплатформенной системе моделирования и анализа бизнес-процессов Ramus Educational.
3. Сгенерировать все доступные отчеты средствами Ramus Educational (сформировать таблицы операций и таблицы документов).

Содержание части отчета №2 по лабораторной работе

1. Краткое описание основного функционала кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational.
2. DFD-диаграмма, созданная в Ramus Educational.
3. Все возможные доступные отчеты по DFD-диаграмме, сформированные средствами кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational.
4. Выводы.

Лабораторная работа №3

Инструментальные средства (CASE-средства) поддержки методологий функционального моделирования процессов (методология IDEF0)

Цель:

- изучить автоматизированные средства построения и анализа моделей предметной области;
- осуществить выбор и применение инструментального средства функционального моделирования процессов (IDEF0 диаграммы).

Время: 4 часа

Краткие теоретические сведения

1. Инструментальные средства поддержки методологии функционального моделирования процессов (IDEF0)

Нотация IDEF0, предназначенная для построения и анализа моделей предметной области, требует автоматизированных средств анализа (Upper CASE) и поддерживается многими инструментальными средствами

Существует множество CASE средств, поддерживающих функциональное моделирование в стандарте IDEF0. В России получили распространение следующие проприетарные системы:

- семейство продуктов CA ERwin ® Data Modeler, ранее AllFusion Process Modeler/AllFusion ERwin Data Modeler (ранее BPWin/ERWin) (CA Technologies, США);
- Design/IDEF (MetaSoftware, США, распространитель – Метатехнология, Москва);
- система IDEF0/EMTool (компания Ориентсофт, г.Минск);
- ARIS (компания Software AG, Германия – результат поглощения компании IDS Scheer автора методологии Августа-Вильгельма Шеера).

Лидирующую позицию в области моделирования данных настоящее время занимает семейство продуктов CA ERwin ® Data Modeler [58], которое предоставляет действенный способ визуализации данных из множественных источников внутри организации, увеличивая эффективность за счет многократного использования и стандартов и одновременно повышая качество сохраняемой информации, обеспечивая целостное представление стратегических информационных активов.

2. CA ERwin Data Modeler Community Edition (условно свободное ПО)

CA ERwin Data Modeler Community Edition – это бесплатное базовое средство моделирования, включающее в себя подмножество функций

флагманского продукта CA ERwin Data Modeler Standard Edition. Оно идеально подходит для начала работы с ведущим в отрасли инструментом моделирования данных. Это решение предоставляет множество базовых функций моделирования данных с ограничением до 25 объектов моделей, в том числе для проектирования и создания баз данных, сравнения моделей, определения стандартов и др., рис.4.6.

Решение CA ERwin Data Modeler Community Edition помогает управлять сложной инфраструктурой данных с помощью следующих основных возможностей:

- визуализация сложных структур данных. Модели данных создаются автоматически, что дает простое графическое представление для визуализации сложных структур баз данных. Ограничение до 25 объектов;
- создание проектов баз данных. Создавайте проекты баз данных непосредственно на основе визуальных моделей, что позволяет повысить эффективность и уменьшить число ошибок. Ограничение до 25 объектов;
- определение стандартов. Повторно используемые стандарты, такие как шаблоны моделей, домены, стандарты именования, улучшают качество и эффективность;
- сравнение моделей и баз данных. Функция Complete Compare осуществляет сравнение моделей, скриптов и баз данных, выводя все различия на экран (в версии Community Edition данные доступны только для чтения);
- отчетность и публикация. Простой, интуитивно понятный интерфейс Report Designer позволяет создавать отчеты в виде текстовых и HTML-файлов, которые могут содержать диаграммы и метаданные.

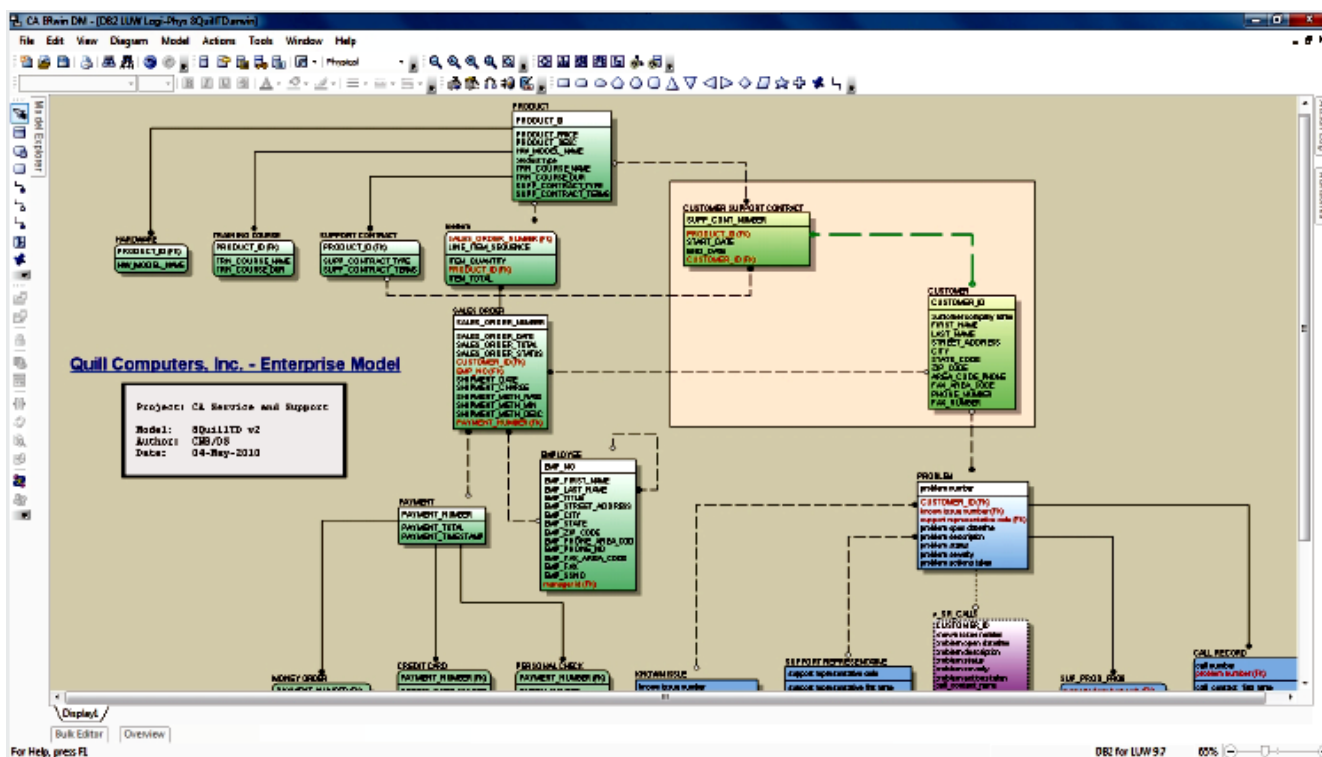


Рис.4.6. Интерфейс CA ERwin Data Modeler Community Edition.

Для свободного скачивания с официального сайта доступен CA ERwin Data Modeler Community Edition (<http://erwin.com/products/data-modeler/community-edition>).

Задание на лабораторную работу №3

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №3 выполнить построение IDEF0-диаграммы при помощи CASE-средств: Ramus Educational и CA ERwin Data Modeler Community Edition.

Порядок выполнения лабораторной работы №3

1. Осуществить построение IDEF0-диаграммы в кроссплатформенной системе моделирования и анализа бизнес-процессов Ramus Educational.
2. Сгенерировать все доступные отчеты средствами Ramus Educational.
3. Исследовать доступные инструменты функционального моделирования процессов системы CA ERwin Data Modeler Community Edition [55,59].
4. Осуществить построение IDEF0-диаграммы в CA ERwin Data Modeler Community Edition.
5. Сгенерировать все доступные отчеты средствами CA ERwin Data Modeler Community Edition.

Содержание части отчета №3 по лабораторной работе

1. Краткое описание доступных инструментов функционального моделирования процессов Ramus Educational.
2. IDEF0-диаграмма, созданная в кроссплатформенной системе моделирования и анализа бизнес-процессов Ramus Educational.
3. Все возможные доступные отчеты по IDEF0-диаграмме, сгенерированные средствами кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational.
4. Краткое описание доступных инструментов функционального моделирования процессов CA ERwin Data Modeler Community Edition.
5. IDEF0-диаграмма, созданная в CA ERwin Data Modeler Community Edition.
6. Все возможные доступные отчеты по IDEF0-диаграмме, сгенерированные средствами CA ERwin Data Modeler Community Edition.
7. Сравнительная характеристика и CA ERwin Data Modeler Community Edition.
8. Выводы.

Лабораторная работа №4

Инструментальные средства (CASE-средства) поддержки методологий моделирования данных и информационного моделирования процессов (методологии ERD, IDEF1, IDEF1X)

Цель:

- изучить автоматизированные средства построения и анализа моделей предметной области;
- осуществить выбор и применение инструментального средства информационного моделирования процессов построения (IDEF1X диаграмм).

Время: 4 часа

Краткие теоретические сведения

Основные CASE-средства способные осуществлять информационное моделирование и построение IDEF1X диаграмм уже были упомянуты и/или рассмотрены в предыдущих лабораторных работах:

- IDEF/Design (компания-производитель – Meta Software Corp) - поддерживает методологии описания и моделирования системных функций (IDEF0/SADT), структур и потоков данных в системе (IDEF1, IDEF1x, ER-диаграммы) и поведения системы (IDEF/CPN – Colored Petri Network);
- ARIS (компания Software AG, Германия – результат поглощения компании IDS Scheer автора методологии Августа-Вильгельма Шеера).
- семейство продуктов CA ERwin® Data Modeler, ранее AllFusion Process Modeler/AllFusion ERwin Data Modeler (ранее BPWin/ERWin) (CA Technologies, США);

Несомненным лидером среди проприетарных CASE-средств поддержки методологий информационного моделирования являются ранее рассмотренное ведущее в отрасли решение для моделирования данных – система CA ERwin Data Modeler Standard Edition.

Бесплатное базовое средство моделирования CA ERwin Data Modeler Community Edition включает в себя подмножество функций флагманского продукта CA ERwin Data Modeler Standard Edition.

Задание на лабораторную работу №4

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №3 выполнить построение IDEF1X-диаграммы при помощи CASE-средства CA ERwin Data Modeler Community Edition.

Порядок выполнения лабораторной работы №4

1. Исследовать функционал моделирования данных системы CA ERwin Data Modeler Community Edition
2. Осуществить информационное моделирование: построить – информационные модели: основанную на ключах и полную атрибутивную модель (IDEF1X-диаграммы) в системе моделирования данных CA ERwin Data Modeler Community Edition [55,59].
3. Сгенерировать все доступные отчеты средствами системы моделирования данных CA ERwin Data Modeler Community Edition.

Содержание части отчета №4 по лабораторной работе

1. Краткое описание функционала моделирования данных системы CA ERwin Data Modeler Community Edition.
2. IDEF1X-диаграмма, созданная в CA ERwin Data Modeler Community Edition.
3. Все возможные доступные отчеты по IDEF1X-диаграммам, сгенерированные средствами CA ERwin Data Modeler Community Edition.
4. Выводы.

Лабораторная работа №5

Инструментальные средства (CASE-средства) поддержки методологий описания логики взаимодействия информационных потоков (методология IDEF3)

Цель:

- изучить автоматизированные средства построения и анализа моделей предметной области;
- осуществить выбор и применение инструментального средства описания логики взаимодействия информационных потоков (IDEF3 диаграммы).

Время: 4 часа

Краткие теоретические сведения

Наиболее популярные системы, поддерживающие функциональное моделирование в стандарте IDEF3 и используемые в России:

- семейство продуктов CA ERwin ® Data Modeler, ранее AllFusion Process Modeler/AllFusion ERwin Data Modeler (ранее BPWin/ERWin) (CA Technologies, США);
- Design/IDEF (MetaSoftware, США, распространитель – Метатехнология, Москва);
- ARIS (компания Software AG, Германия – результат поглощения компании IDS Scheer автора методологии Августа-Вильгельма Шеера).

Задание на лабораторную работу №5

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №5 выполнить построение IDEF3-диаграммы при помощи CASE-средств: Ramus Educational и CA ERwin Data Modeler Community Edition.

Порядок выполнения лабораторной работы №5

1. Исследовать доступный функционал построения IDEF3-диаграмм кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational [58].
2. Осуществить построение IDEF3-диаграммы в кроссплатформенной системе моделирования и анализа бизнес-процессов Ramus Educational.
3. Сгенерировать все доступные отчеты средствами Ramus Educational.
4. Исследовать доступный функционал построения IDEF3-диаграмм системы моделирования данных CA ERwin Data Modeler Community Edition [55,59].

5. Осуществить построение IDEF3-диаграммы в CA ERwin Data Modeler Community Edition.

6. Сгенерировать все доступные отчеты средствами CA ERwin Data Modeler Community Edition.

Содержание части отчета №5 по лабораторной работе

1. Краткое описание функционала описания логики взаимодействия информационных потоков системы Ramus Educational CA ERwin Data Modeler Community Edition.

2. IDEF3-диаграмма, созданная в кроссплатформенной системе моделирования и анализа бизнес-процессов Ramus Educational.

3. Все возможные доступные отчеты по IDEF3-диаграмме, сгенерированные средствами кроссплатформенной системы моделирования и анализа бизнес-процессов Ramus Educational.

4. IDEF3-диаграмма созданная в CA ERwin Data Modeler Community Edition.

5. Все возможные доступные отчеты по IDEF3-диаграмме, сгенерированные средствами CA ERwin Data Modeler Community Edition.

6. Сравнительная характеристика и CA ERwin Data Modeler Community Edition.

7. Выводы.

Лабораторная работа №6

Инструментальные средства (CASE-средства) поддержки методологии BPMN

Цель:

- изучить автоматизированные средства построения и анализа моделей предметной области;
- осуществить выбор и применение инструментального средства моделирования бизнес-процессов (BPMN-диаграммы).

Время: 4 часа

Краткие теоретические сведения

1. Инструментальные средства поддержки моделирования бизнес-процессов BPMN

Современные инструментальные средства поддержки моделирования бизнес-процессов, как правило, связывают построенные модели процессов с операционной деятельностью компании и предоставляют механизмы контроля и мониторинга процессов.

Из наиболее популярных зарубежных программных продуктов выделяются:

- ARIS Business Performance Edition (IDS Scheer AG);
- CA ERwin Data Modeler, ранее AllFusion Process Modeler/AllFusion ERwin Data Modeler (ранее BPWin/ERWin) (CA);
- Hyperion Performance Scorecard (Oracle);
- IBM WebSphere Business Modeler (IBM);
- SAP Strategic Enterprise Management (SAP).

Среди российских разработок можно выделить:

- Business Studio (Современные технологии управления);
- Бизнес-инженер (Битек);
- Инталев: Корпоративный навигатор (Инталев);
- ОРГ-Мастер Про (Бизнес Инжиниринг Групп).

Российские разработки в первую очередь предназначены для описания/проектирования деятельности компании. Они, как правило, предоставляют возможность описания практически любой предметной области. Зарубежные же производители больше ориентированы на исполнение. В большинстве случаев их продукты являются одним или несколькими модулями в линейке программного обеспечения, предоставляемого производителем.

Наиболее мощной из представленных выше систем и самой дорогой является инструментальная система ARIS (разработчик IDS Scheer) [60], которая представляет собой интегрированное семейство программных продуктов, предназначенных для структурированного описания, анализа и последующего совершенствования бизнес-процессов предприятия, а также подготовки организаций к внедрению сложных информационных систем. Программные

продукты ARIS используются на всех этапах цикла работ по созданию и развитию бизнеса.

Все многообразие программных продуктов ARIS можно разделить на четыре платформы, одна из которых поддерживает разработку стратегии организации, а три остальных соответствуют основным этапам жизненного цикла системы управления (разработка, внедрение и контроллинг).

Программные продукты ARIS Platform группируются в четыре специализированных модуля:

- ARIS Strategy Platform позволяет создавать системы сбалансированных показателей и оптимизировать бизнес-процессы в соответствии с ними.

- ARIS Design Platform позволяет выявлять организационные, структурные и технологические недостатки и определять возможности оптимизации.

- ARIS Implementation Platform позволяет реализовывать бизнес-процессы в ИТ-среде. Модули, относящиеся к платформе:

- ARIS Controlling Platform используется для поиска возможностей совершенствования путем оценки и визуализации выполненных процессов, импортированных из ИТ-систем.

В совокупности четыре специализированных модуля образуют единую интегрированную систему, направленную на поддержание полного цикла управления бизнес-процессами.

Высокая стоимость и сложность освоения программы, являющаяся следствием высокой функциональности продукта основная причина невысокой распространенности системы.

2. Инструмент для моделирования бизнес-процессов ARIS Express (свободное ПО)

В 2009 году корпорация IDS Scheer выпустила ARIS Express – бесплатную упрощенную версию программы для моделирования бизнес-процессов (свободное скачивание <http://www.ariscommunity.com/aris-express>).

Бесплатная версия программы поддерживает только базовые типы диаграмм, не имеет многопользовательской поддержки, не использует базу данных, не содержит инструментов для формирования отчетов и средств анализа модели. И самое главное: ARIS Express не поддерживает связи между создаваемыми объектами в отличие от полноценной платной версии, то есть отсутствует контроль целостности и непротиворечивости модели. Это означает, что при редактировании одной модели программа не будет вносить соответствующие изменения в другую модель, а также не будет проверять существуют ли должности, указываемые в качестве ответственных в процессе и т.д.

ARIS Express поддерживает следующие типы моделей, рис.4.7:

- организационная диаграмма (Organizational chart);
- бизнес-процесс (Business process);

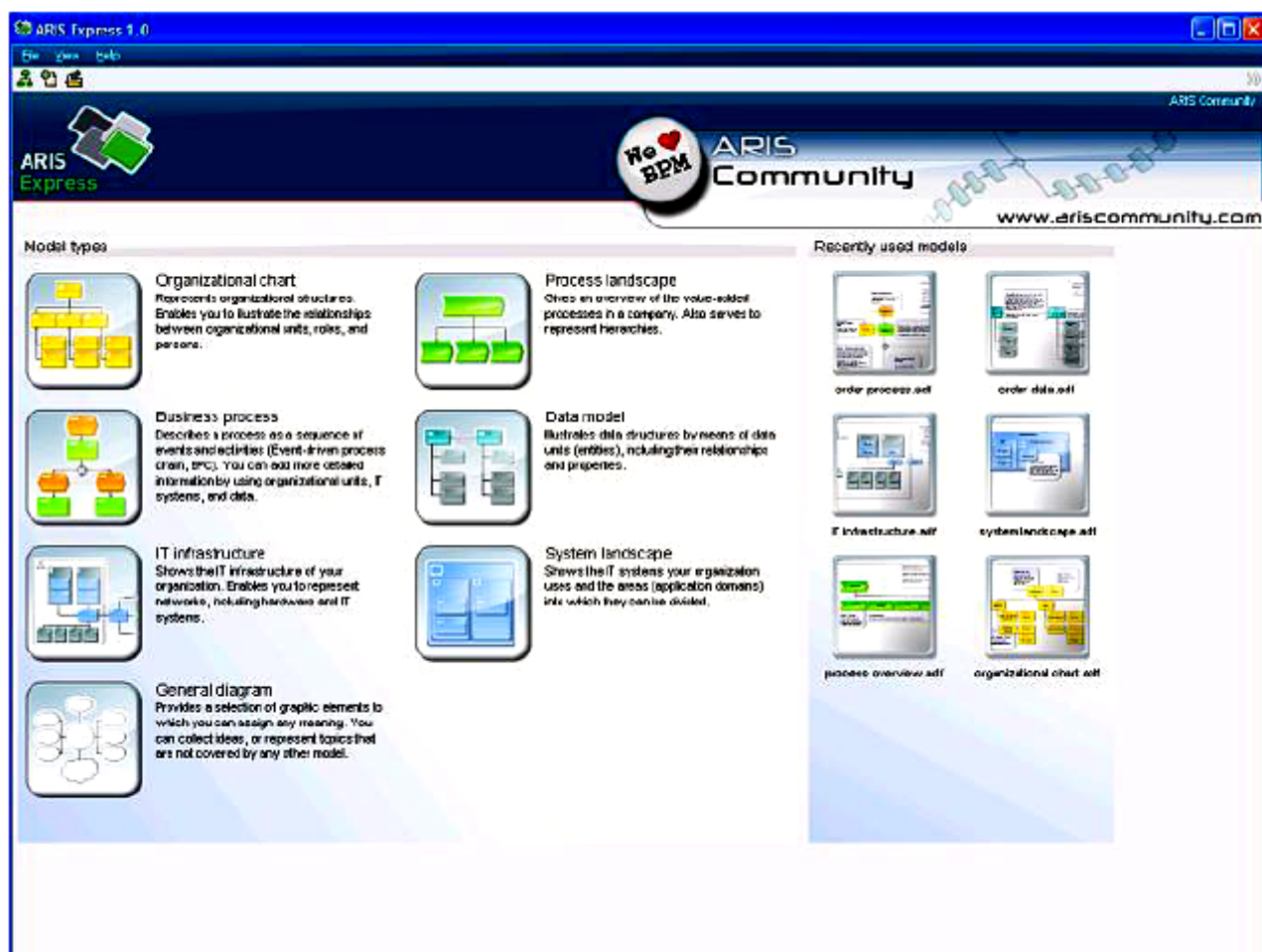


Рис.4.7. Интерфейс ARIS Express.

- ИТ-инфраструктура (IT infrastructure);
- карта процессов (Process landscape);
- модель данных (Data model);
- карта систем (System landscape);
- доска (Whiteboard);
- BPMN диаграмма версии 2.0 (BPMN diagram);
- общие диаграммы (General diagram).

Первый модуль предназначен для построения организационной структуры, рис.4.8.

Включает технологию Smart Design, которая позволяет очень быстро формировать модель в таблице и мгновенно синхронизировать её с графическим отображением в редакторе.

Для моделирования процессов можно использовать стандартный модуль для бизнес-процессов, который позволяет рисовать процессы в нотации eEPC или же использовать редактор диаграмм BPMN. Набор элементов минимальный, но всё необходимое присутствует, рис.4.9. Полученные диаграммы системой не обрабатываются и не «исполняются» как в BPM-системах, поэтому выбор нотации ни на что, по сути, не влияет.

Как и в случае с модулем «Организационная диаграмма» можно строить процесс вручную или при помощи Smart Design. Для дополнительного удобства

разработчики предусмотрели готовые фрагменты типовых диаграмм, которые можно перенести мышкой в редактор. При желании пользователь может заготовить и сохранить свои фрагменты диаграмм для дальнейшего использования при моделировании.

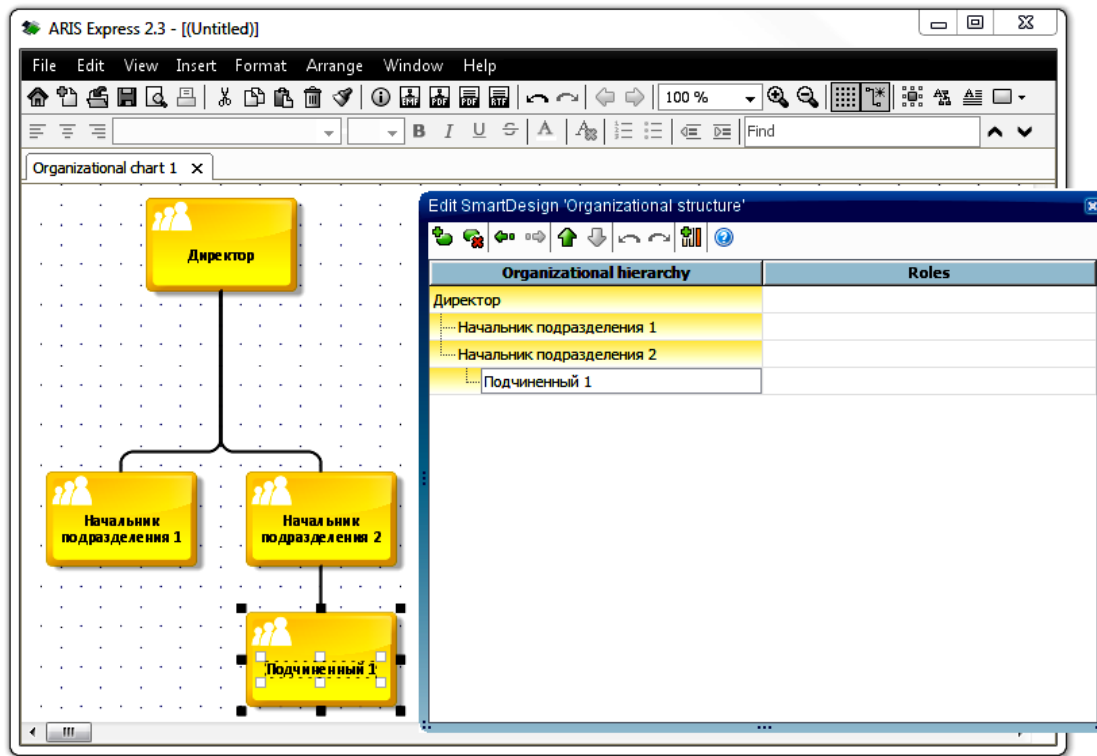


Рис.4.8. Оргструктура в ARIS Express.

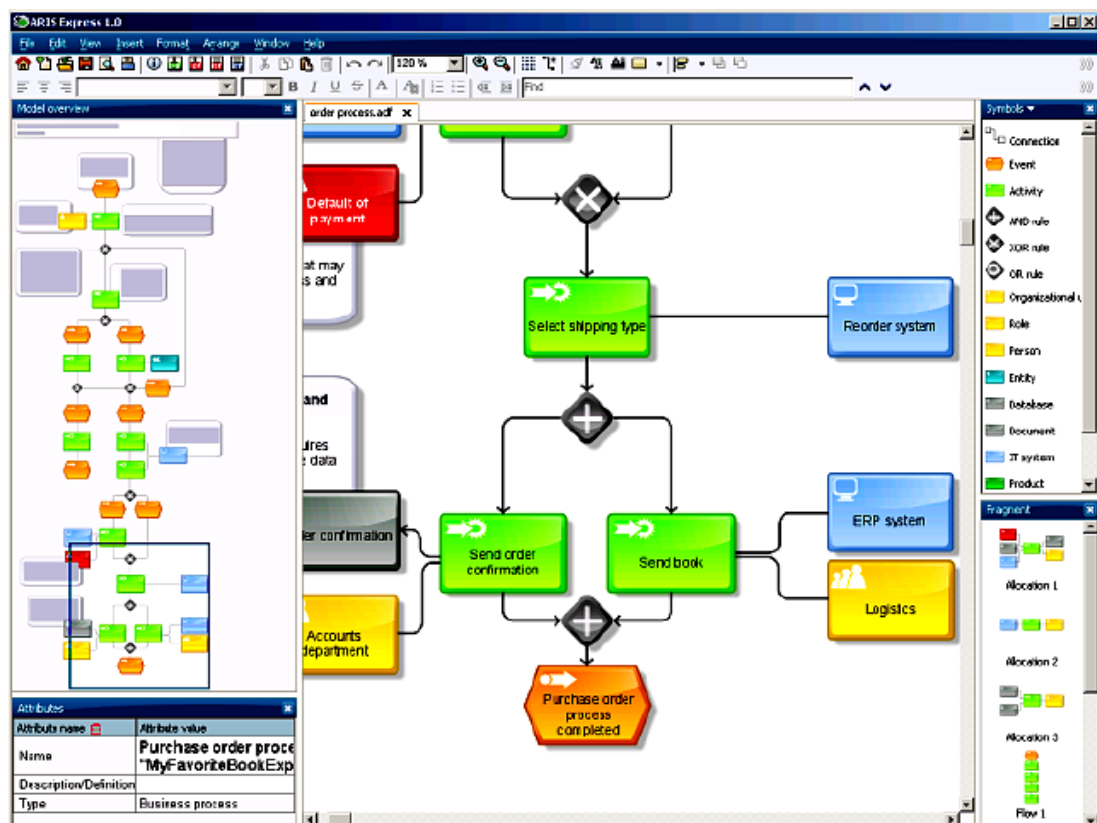


Рис.4.9. Бизнес-процессы в ARIS Express.

Любую диаграмму можно экспортировать в PDF или RTF-формат, а также сохранить как картинку или распечатать. Более того, модели, сохраненные в формат «adf» могут быть перенесены в полноценную версию ARIS. Сохраненные модели можно защитить при помощи пароля.

Остальные модули программы являются вариациями различных диаграмм. При помощи них бизнес-аналитик может спроектировать инфраструктуру компании или модель данных.

Задание на лабораторную работу №6

В соответствии с вариантом предметной области и на основании результатов выполнения задания к практическому занятию №6 выполнить построение BPMN-диаграммы при помощи CASE-средств: ARIS Express.

Порядок выполнения лабораторной работы №6

1. Исследовать доступный функционал системы моделирования и анализа бизнес-процессов ARIS Express [61].
2. Осуществить построение BPMN-диаграммы в системе моделирования и анализа бизнес-процессов ARIS Express.
3. Осуществить построение организационной диаграммы, диаграммы бизнес-процесса, модели данных, IT-инфраструктуры.
4. Сгенерировать все доступные отчеты средствами ARIS Express.

Содержание части отчета №6 по лабораторной работе

1. Краткое описание функционала системы моделирования и анализа бизнес-процессов ARIS Express.
2. Организационная диаграмма, диаграммы бизнес-процесса, модель данных, диаграмма IT-инфраструктуры.
3. BPMN-диаграмма, созданная в системе моделирования и анализа бизнес-процессов ARIS Express.
4. Все возможные доступные отчеты, сгенерированные средствами системы моделирования и анализа бизнес-процессов ARIS Express.
5. Выводы.

Лабораторная работа №7

Инструментальные средства поддержки проектирования модельно-ориентированных систем (AnyLogic)

Цель:

- изучить инструментальные средства построения и анализа моделей предметной области;
- осуществить выбор и применение инструментального средства имитационного моделирования бизнес-процессов.

Время: 4 часа

Краткие теоретические сведения

1. Инструментальные средства поддержки модели-ориентированного проектирования

В современной теории имитационного моделирования существуют четыре основных подхода [62]:

- моделирование динамических систем (системы имитационного моделирования: MATLAB Simulink, VinSim и др.);
- дискретно-событийное моделирование (GPSS, Arena, eMPlant, AutoMod, PROMODEL, Enterprise Dynamics, FlexSim и др.);
- системная динамика (СИМ: VenSim, PowerSim, iThink, и др.);
- агентное моделирование (системы имитационного моделирования AnyLogic, Swarm, Repast и др.).

В каждом из этих направлений развиваются свои инструментальные средства, свои системы имитационного моделирования и языки.

Системная динамика (СД) и Динамические системы – традиционные устоявшиеся подходы, Агентное моделирование – относительно новый. СД и Динамические системы оперируют в основном с непрерывными во времени процессами, а Дискретно-событийное моделирование и Агентное – в основном с дискретными.

В качестве базовых концепций формализации и структуризации в современных системах имитационного моделирования, наиболее часто применяемых при решении бизнес-задач, используются следующие два подхода:

- процессно-транзактно-ориентированные системы моделирования, основанные на описании процессов (process description). На современном рынке информационных технологий они представляют дискретно-событийный подход имитационного моделирования и являются наиболее представительным классом систем такого рода. Это системы: GPSS, Arena, Extend, AutoMod, ProModel, Witness, Taylor, eM-Plant, QUEST, SIMFACTORY II.5, SIMPLE++ и др.;

- агентное моделирование, при котором модели используются для исследования децентрализованных систем, динамика и функционирование

которых определяется не глобальными правилами и законами, а наоборот, эти правила и законы являются результатом индивидуальной активности членов группы. Представитель российского рынка систем этого класса является пакет AnyLogic.

На рис.4.10 представлены среды, ориентированные на разные подходы в имитационном моделировании, из которого видно, что рынок очень неравномерный, дискретные системы наиболее представительны.

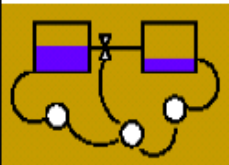
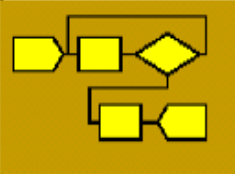

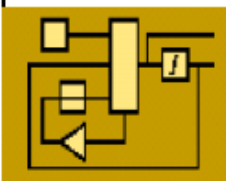
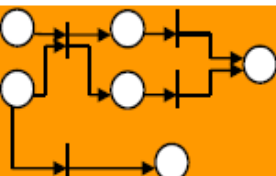
| Системная динамика | Дискретные системы | Агентное моделирование | Динамические системы | Сети |
|---|--|---|--|---|
|  |  |  |  |  |
| Vensim, iThink, Powersim, AnyLogic | GPSS, Simula, Arena, AutoMod, AnyLogic, Extend, ProModel, QUEST, SIMFACTORY II.5, SIMPLE++, eM-Plant, Taylor ED, WITNESS | AnyLogic | MATLAB | ARIS |

Рис.4.10. Базовые концепции и инструментальные решения имитационного моделирования.

2. Инструмент для для разработки и исследования имитационных моделей (свободное ПО)

Пакет AnyLogic – отечественный профессиональный инструмент нового поколения, который предназначен для разработки и исследования имитационных моделей. Разработчик продукта – компания «Экс Джей Текнолоджис» (XJ Technologies), г. Санкт-Петербург.

AnyLogic™ – инструмент имитационного моделирования новейшего поколения. Он основан на результатах, полученных в теории моделирования и в информационных технологиях за последнее десятилетие.

Уникальность AnyLogic™ состоит в его способности эффективно решать задачи моделирования любого масштаба и уровня абстракции, в том числе для разнородных систем в их взаимосвязи.

AnyLogic имеет развитый базовый язык дискретного и смешанного дискретно/непрерывного моделирования, на основе которого построены решения

для конкретных областей: библиотека Enterprise Library, а также Material Flow Library (потoki материалов) и Healthcare Library (работа медицинских учреждений)

Реализация стандартных объектов открыта для пользователя, их функциональность может быть как угодно расширена, вплоть до создания собственных библиотек.

Возможности AnyLogic представлены на рис.4.10.

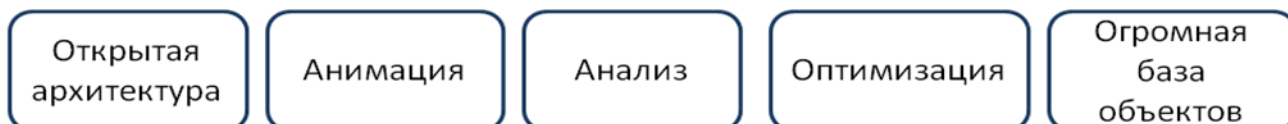


Рис.4.10. Возможности AnyLogic.

Базовые инструменты для разработки модели в среде AnyLogic, Окно редактора которого представлены на рис.4.11, имеет следующие элементы:

- панели инструментов;
- панель Проекты (дерево всех объектов проекта), рис.4.12;
- панель Палитра;
- панель Ошибки;
- окно свойств, рис.4.13.
- окно графического редактора, рис.4.14.

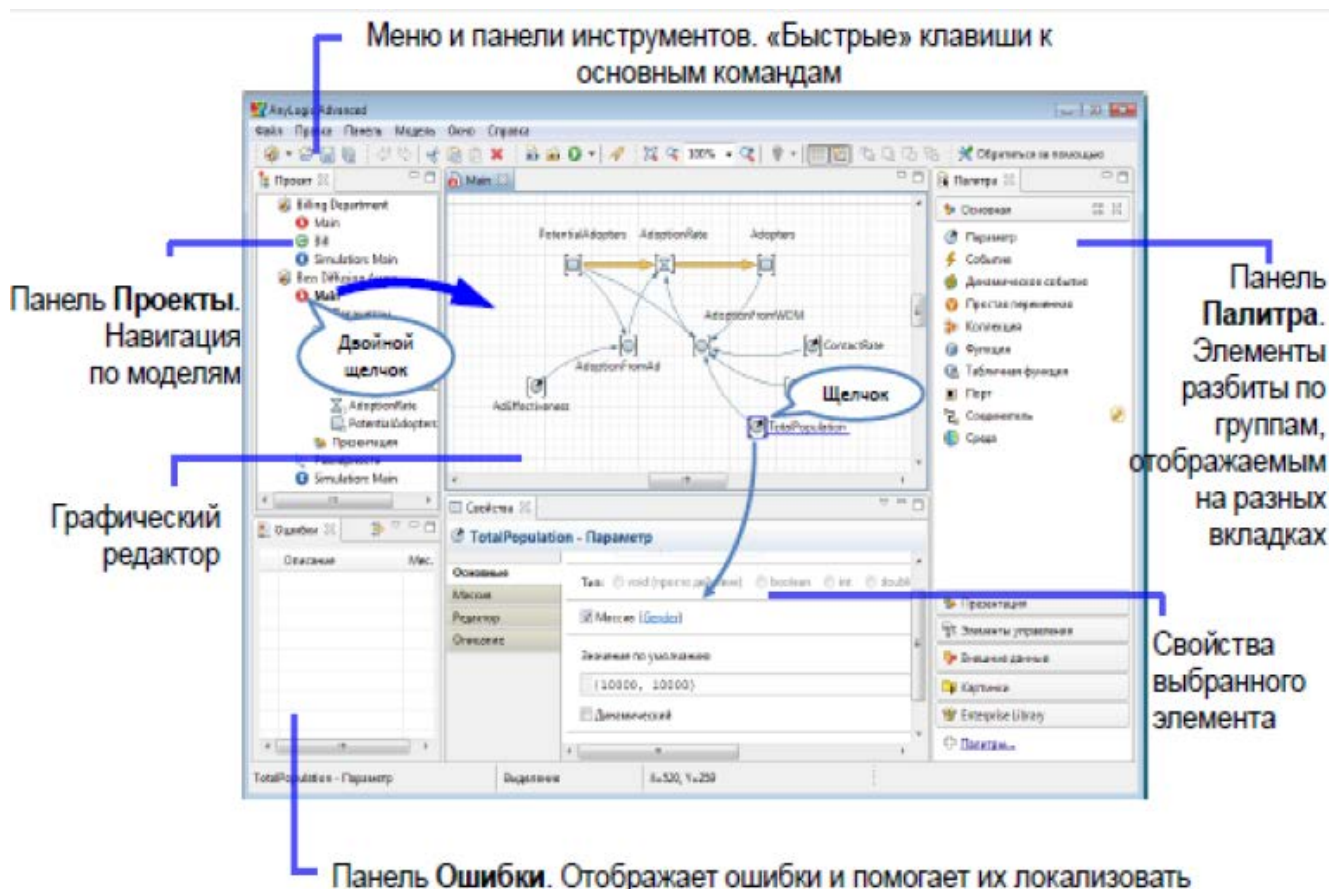


Рис.4.11.Графический редактор модели AnyLogic.

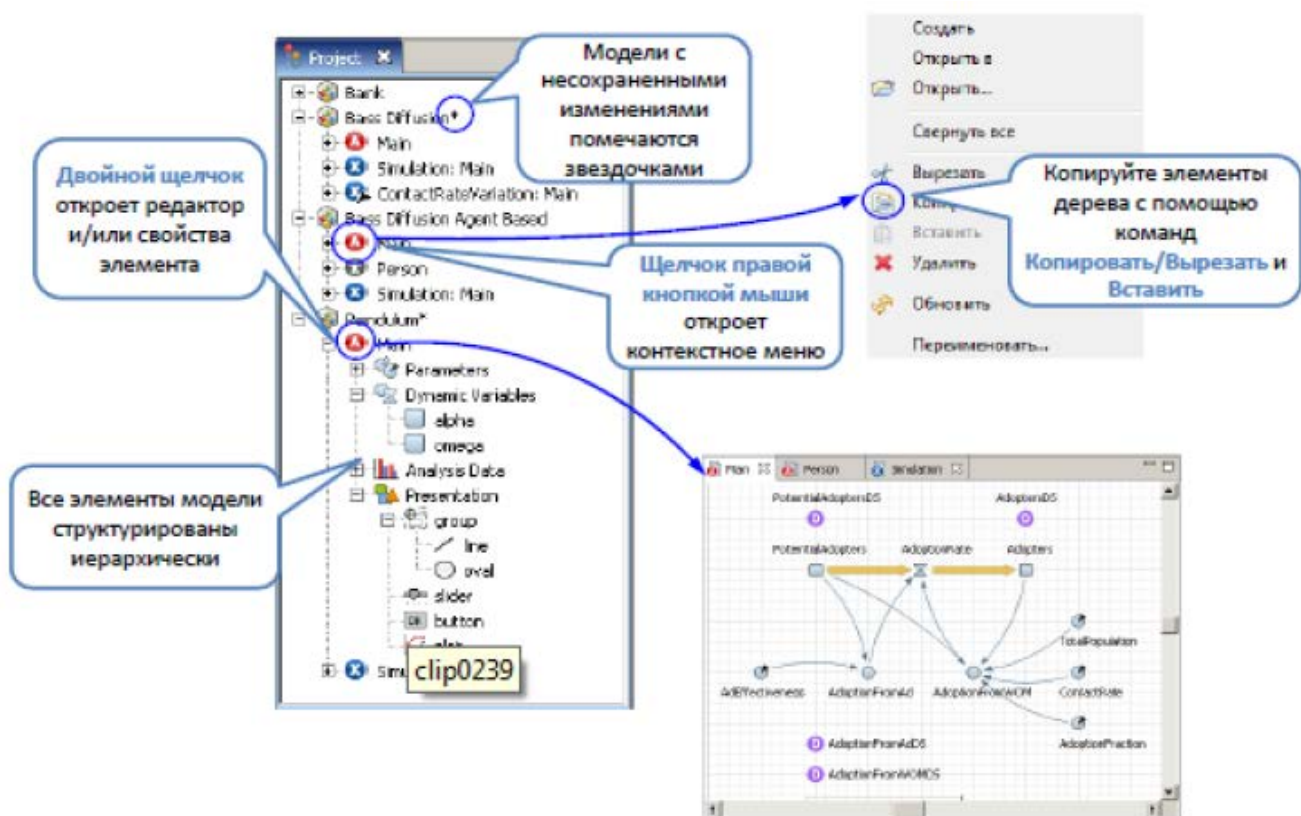


Рис.4.12. Панель Проекты.

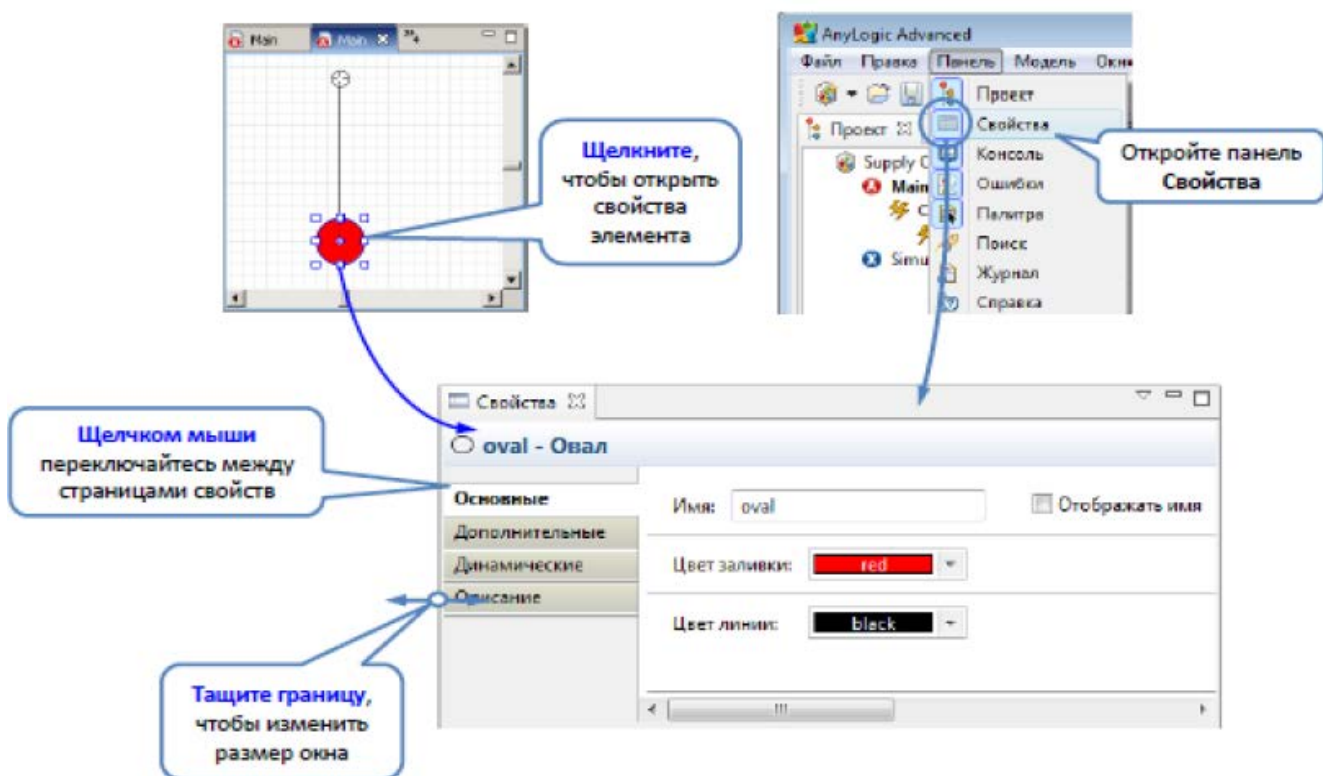


Рис.4.13. Окно свойств.

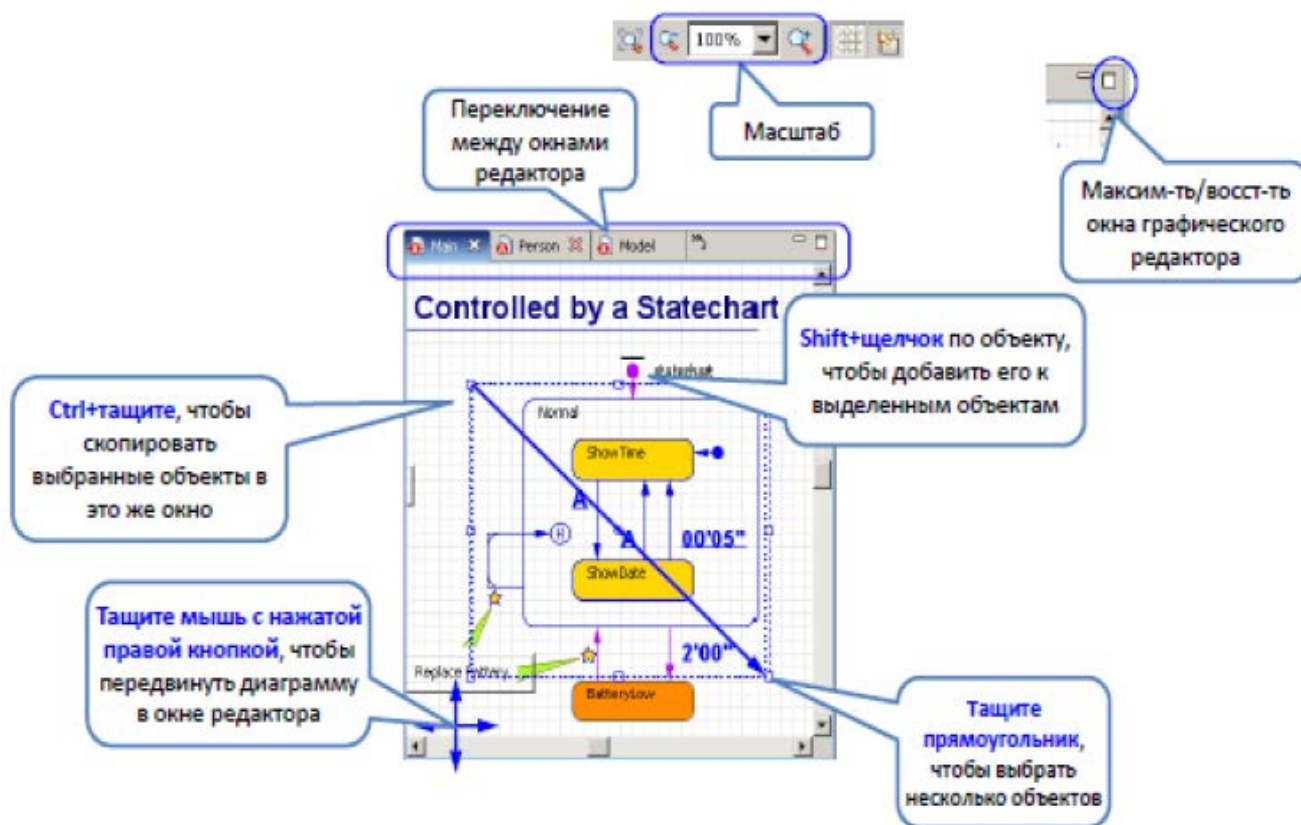


Рис.4.14. Окно графического редактора

AnyLogic доступен бесплатно для обучения студентов и самообразования, в виде версии Personal Learning Edition (свободное скачивание <http://www.anylogic.ru/downloads>).

Задание на лабораторную работу №7

1. Исследовать возможности инструмент имитационного моделирования AnyLogic [64].

2. Используя результаты выполнения практического задания №7 осуществить имитационное моделирование выбранного для исследования процесса средствами AnyLogic.

Порядок выполнения лабораторной работы №7

1. Построить структурную диаграмму процесса. При построении модели нужно задать ее структуру (т.е. описать, из каких частей состоит модель системы).

2. Настроить параметры модели, то есть задать характеристики объектов.

3. Задать поведение отдельных объектов системы, то есть построить диаграмму состояний (или стейтchart) – модифицированные графы переходов конечного автомата. Стейтchart позволяет графически задать пространство состояний алгоритма поведения объекта, а также события, которые являются причинами срабатывания переходов из одних состояний в другие, и действия, происходящие при смене состояний.

4. Осуществить запуск модели.
 5. Задать данные модели, изменяя свойства созданных объектов.
 6. Запустить модель и проанализировать работу. Как изменились характеристики процесса? Сравните результаты с предыдущим прогоном.
 7. Включить сбор статистики.
- * Создать анимацию модели (дополнительное задание).

Содержание части отчета №7 по лабораторной работе

1. Имитационная модель системы, сформированную с использованием системы AnyLogic.
2. Результаты моделирования.
3. Выводы.

4.2. Требования к содержанию и оформлению отчетов

Отчеты по лабораторным работам и практическим заданиям оформляются согласно правилам оформления принятыми на кафедре, ГОСТам и ЕСКД.

Основные правила по оформлению отчетной документации:

Параметры страницы: А4 (21×29,7), ориентация – книжная (допускается использовать альбомную ориентацию страницы для выполнения схем и таблиц).

Поля: левое – 2.5, верхнее – 1.5, нижнее – 1.5, правое – 1.

Нумерация страницы – внизу, справа. Нумерация ведется с титульного листа, номер на титульном листе не ставиться.

Шрифт Times New Roman, кегль 14, интервал – одинарный.

Заголовки разделов: абзацный отступ – 0, выравнивание по центру, шрифт – жирный, нумерация – арабскими цифрами, точка в конце названия раздела не ставиться.

Заголовки подразделов (допускается три уровня, например 1.1., 1.1.1.): абзацный отступ – 1.25÷1.5, выравнивание по ширине, шрифт – жирный, точка в конце названия подраздела не ставиться.

Основной текст: абзацный отступ – 1.25÷1.5, выравнивание по ширине, шрифт – обычный.

Нумерация рисунков и таблиц – сквозная внутри раздела (например, в разделе 1 – рис. 1.1., рис.1.1.2 и т.д., или табл.1.1., табл.1.2. и т.д.).

Рисунки помещаются после упоминания их в тексте и имеют подпись, размещаемую под рисунком без абзацного отступа и имеющую выравнивание по центру и точку на конце названия (например Рис.1.1. Название.).

Таблицы размещаются после ссылки на них в тексте. Название приводится над таблицей, без абзацного отступа с выравниванием по центру, без точки на конце названия (например (Таблица 2.2. Название).

Допускается выносить рисунки и таблицы в Приложения. В этом случае ссылка должна содержать номер приложения (например: рис.1.1. Приложение 1 или табл.А1 Приложение А).

Основная часть должна содержать ссылки на используемую литературу или информационные источники, список которых приводится после раздела Выводы и перед Приложениями. Ссылка заключается в квадратные скобки (например – [1], [5,7], [3-6].

Приложения нумеруются арабскими цифрами (Приложение 1, Приложение 2) или обозначаются русскими заглавными буквами в порядке их следования (Приложение А, Приложение Б). Слово Приложение...выравнивается по правому краю и имеет жирный шрифт. Название приложение располагается на следующей строке, без абзацного отступа, выравнивание по центру, шрифт – жирный.

По завершению изучения курса у студента должен быть сформировать набор отчетов (Приложение №2), сведенных в единый документ и имеющий единый титульный лист (Приложение №3), на котором отражаются результаты прохождения этапов изучения дисциплины.

Каждый раздел этого документа является отчетом по выполнению соответствующего практического задания и лабораторной работы (обязательные разделы и правила выполнения отчетов представлены в Приложении 2).

Сформированный документ, с отметками о выполнении всех практических заданий и лабораторных работ обязателен для представления и является подтверждением о допуске к итоговому контролю.

4.3. Организация защиты и критерии оценивания выполнения практических заданий и лабораторных работ

К защите представляется отчет, включающий в себя результаты выполнения практического задания и лабораторной работы, выполненные согласно правилам, описанным в п.4.2 и единый титульный лист, на котором отмечаются результаты выполнения заданий.

К отчетам прилагается электронный носитель, содержащий папки с файлами диаграмм и отчетов, созданных в ходе выполнения практических заданий и лабораторных работ.

На проверку теоретической подготовки, проводимой по контрольным вопросам, отводиться 5-6 минут.

Степень усвоения теоретического материала оценивается по следующим критериям:

- ***оценка «отлично» выставляется, если:***

- последовательно, четко, связно, обоснованно и безошибочно с использованием принятой терминологии изложен учебный материал, выделены главные положения, ответ подтвержден конкретными примерами, фактами;
- самостоятельно и аргументировано сделан анализ, обобщение, выводы, установлены межпредметные (на основе ранее приобретенных знаний) и внутрипредметные связи, творчески применены полученные знания в незнакомой ситуации;
- самостоятельно и рационально используются справочные материалы, учебники, дополнительная литература, первоисточники; применяется система условных обозначений при ведении записей, сопровождающих ответ; используются для доказательства выводы из наблюдений и опытов, ответ подтверждается конкретными примерами;
- допускает не более одного недочета, который легко исправляется по требованию преподавателя.

- ***оценка «хорошо» ставится, если:***

- дан полный и правильный ответ на основе изученных теорий; допущены незначительные ошибки и недочеты при воспроизведении изученного материала, определения понятий, неточности при использовании научных терминов или в выводах и обобщениях из наблюдений и опытов; материал излагает в определенной логической последовательности;

- самостоятельно выделены главные положения в изученном материале; на основании фактов и примеров проведено обобщение, сделаны выводы, установлены внутриспредметные связи.
- допущены одна негрубая ошибка или не более двух недочетов, которые исправлены самостоятельно при требовании или при небольшой помощи преподавателя; в основном усвоил учебный материал.
- **оценка «удовлетворительно» ставится, если:**
 - усвоено основное содержание учебного материала, но имеются пробелы в усвоении материала, не препятствующие дальнейшему изучению; материал излагает несистематизированно, фрагментарно, не всегда последовательно;
 - показана недостаточная сформированность отдельных знаний и умений; выводы и обобщения аргументируются слабо, в них допускаются ошибки;
 - допущены ошибки и неточности в использовании научной терминологии, даются недостаточно четкие определения понятий; в качестве доказательства не используются выводы и обобщения из наблюдений, фактов, опытов или допущены ошибки при их изложении;
 - обнаруживается недостаточное понимание отдельных положений при воспроизведении текста учебника (записей, первоисточников) или неполные ответы на вопросы преподавателя, с допущением одной – двух грубых ошибок.
- **оценка «неудовлетворительно» ставится, если:**
 - не усвоено и не раскрыто основное содержание материала; не сделаны выводы и обобщения;
 - не показано знание и понимание значительной или основной части изученного материала в пределах поставленных вопросов или показаны слабо сформированные и неполные знания и неумение применять их к решению конкретных вопросов и задач по образцу;
 - при ответе (на один вопрос) допускается более двух грубых ошибок, которые не могут быть исправлены даже при помощи преподавателя;
 - не даются ответы ни на один из поставленных вопросов.

Оценка выполнения практических заданий и лабораторных работ проводится по следующим критериям

- **оценка «отлично» ставится, если студент:**
 - творчески планирует выполнение работы;
 - самостоятельно и полностью использует знания программного материала;
 - правильно и аккуратно выполняет задание;
 - умеет пользоваться литературой и различными информационными источниками;
 - выполнил работу без ошибок и недочетов или допустил не более одного недочета
- **оценка «хорошо» ставится, если студент:**
 - правильно планирует выполнение работы;

- самостоятельно использует знания программного материала;
- в основном правильно и аккуратно выполняет задание;
- умеет пользоваться литературой и различными информационными источниками;
- выполнил работу полностью, но допустил в ней: не более одной негрубой ошибки и одного недочета или не более двух недочетов.
- ***оценка «удовлетворительно» ставится, если студент:***
 - допускает ошибки при планировании выполнения работы;
 - не может самостоятельно использовать значительную часть знаний программного материала;
 - допускает ошибки и неаккуратно выполняет задание;
 - затрудняется самостоятельно использовать литературу и информационные источники;
 - правильно выполнил не менее половины работы или допустил:
 - не более двух грубых ошибок или не более одной грубой и одной негрубой ошибки и одного недочета;
 - не более двух- трех негрубых ошибок или одной негрубой ошибки и трех недочетов;
 - при отсутствии ошибок, но при наличии четырех-пяти недочетов.
- ***оценка «неудовлетворительно» ставится, если студент:***
 - не может правильно спланировать выполнение работы;
 - не может использовать знания программного материала;
 - допускает грубые ошибки и неаккуратно выполняет задание;
 - не может самостоятельно использовать литературу и информационные источники;
 - допустил число ошибок недочетов, превышающее норму, при которой может быть выставлена оценка «3»;
 - если правильно выполнил менее половины работы;
 - не приступил к выполнению работы;
 - правильно выполнил не более 10% всех заданий.

Раздел 5
Перечень вопросов
для подготовки к экзамену по дисциплине «Методы и средства
проектирования информационных систем и технологий»

1. Понятие информационной системы, информационной технологии, проектирования, информационной модели.
2. Подходы к построению и проектированию информационных систем.
3. Понятие метода проектирования ИС, их классификация.
4. Основные принципы системного подхода к созданию ИС.
5. Понятие технологии проектирования. Требования к технологии проектирования.
6. Классификация технологий проектирования ИС. Выбор технологии проектирования ИС.
7. Классификация средств проектирования ИС.
8. Основные стадии жизненного цикла проектирования ИС.
9. Модели жизненного цикла ИС.
10. Классификация стандартов на проектирование и разработку информационных систем.
11. Международный стандарт ISO/IEC 12207:1995-08-01 (ГОСТ Р ИСО/МЭК 12207).
12. Международный стандарт ISO/IEC 15288:2002 (ГОСТ Р ИСО/МЭК 15288-2005).
13. Комплекс стандартов ГОСТ 34.
14. Фирменные стандарты (Oracle CDM, MSF и др.).
15. Понятие методологии проектирования информационных систем.
16. Методологии структурного подхода (анализа).
17. Методология моделирования функциональной структуры объектов – SADT. Методология моделирования данных – ERD (Entity-Relationship Diagrams) (case-метод Баркера).
18. Методология моделирования работы в реальном времени – STD.
19. Методология функционального моделирования процессов – IDEF0. Характеристика диаграмм. Типы взаимосвязей между блоками.
20. Методология функционального моделирования процессов – IDEF0. Последовательность создания функциональных моделей.
21. Методология анализа взаимосвязей между информационными потоками – IDEF1 (IDEF1X).
22. Методология описания (документирования) и моделирования процессов – IDEF3.
23. Методология моделирования, анализа и реорганизации бизнес-процессов – BPMN.
24. Методологии объектно-ориентированного подхода (анализа).
25. Методологии модельно-ориентированного подхода (анализа).
26. Гибкие методологии проектирования (agile-методы).
27. Понятие CASE-технологии проектирования ИС. Основные принципы Case-технологии.
28. Классификация CASE-средств, стратегия их выбора.

Список литературы и информационных ресурсов

1. Федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 31.12.2014) «Об информации, информационных технологиях и о защите информации». Режим доступа – http://www.consultant.ru/document/cons_doc_LAW_173622/.
2. ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными. Режим доступа – <http://gostexpert.ru/gost/gost-34.321-96>.
3. ISO/IEC 2382-1:1993 Information technology; Vocabulary; Part 1: Fundamental terms (Информационные технологии. Словарь. Часть 1. Основные термины) Режим доступа – <http://rossert.narod.ru/alldoc/info/2z23/g28647.html>.
4. ГОСТ 34.003-90 Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения. Режим доступа – <http://base.garant.ru/187632/>.
5. ISO/IEC 38500:2015 Information technology – Governance of IT for the organization (Информационная технология. Корпоративное управление информационными технологиями) Режим доступа – http://www.iso.org/iso/catalogue_detail.htm?csnumber=62816.
6. Новиков А.М., Новиков Д.А. Методология. – М.: СИНТЕГ. – 663 с. Режим доступа – http://www.methodolog.ru/books/methodology_full.pdf.
7. Теория систем и системный анализ: Учебник/В.М. Вдовин, Л.Е. Суркова, В.А. Валентинов. – М.: Издательско-торговая корпорация «Дашков и К», 2010. – 640 с. Режим доступа – <http://ir.nmu.org.ua/bitstream/handle/123456789/143657/a261cee4cef2ef83d70df07a16c9ded6.pdf?sequence=1>.
8. ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств Режим доступа – <http://www.complexdoc.ru/lib/ГОСТ%20Р%20ИСО%7СМЭК%2012207-99>.
9. РД 50-680-88 Методические указания. Автоматизированные системы. Основные положения. Режим доступа – <http://www.gostinfo.ru/catalog/Details/?id=2033907#.VZIGFC-5GDM>.
10. Избачков Ю.С, Петров В.Н., Васильев А.А., Тепина И.С. Информационные системы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2011. – 544 с: ил.
11. ГОСТ Р ИСО/МЭК 15288-2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем. Режим доступа – <http://vsegost.com/Catalog/20/2011.shtml>.
12. IEEE Std 1220-2005 IEEE Standard for Application and Management of the Systems Engineering Process. Режим доступа – <https://standards.ieee.org/findstds/standard/1220-2005.html>.
13. ГОСТ Р ИСО/МЭК 15504-2009 Информационные технологии. Оценка процессов. Режим доступа – <http://protect.gost.ru/document.aspx?control=7&id=175314>.
14. ГОСТ Р ИСО 10303 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Режим доступа – <http://www.gosthelp.ru/gost/gost18561.html>.

15. IDEF Family of Methods. A Structured Approach to Enterprise Modeling & Analysis Режим доступа – <http://www.idef.com/>.
16. Documents Associated with Unified Modeling Language (UML) Version 2.5 Режим доступа – <http://www.omg.org/spec/UML/2.5/>.
17. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE Std 1471-2000). Режим доступа – <http://standards.ieee.org/findstds/standard/1471-2000.html>.
18. ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. Режим доступа – <http://vsegost.com/Catalog/10/10698.shtml>.
19. Орлов, С.А. Технологии разработки программного обеспечения: учеб. / С.А. Орлов. – СПб.: Питер, 2002. – 464 с.
20. Проектирование информационных систем: учебное пособие. Авторы: Абрамов Г.В., Медведкова И.Е., Коробова Л.А. Издательство: ВГУИТ, 2012 г.
21. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.
22. Зыков С.В. Основы проектирования корпоративных систем / С.В. Зыков; Нац.исслед.ун-т «Высшая школа экономики»: Изд. дом Высшей школы экономики; Москва; 2012.
23. Дэвид А. Марк, Клемент Мак-Гоуэн Методология структурного анализа и проектирования SADT. М.:Метатехнология, 1993.
24. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с. [Электронный ресурс] Режим доступа – <http://books.ifmo.ru/file/pdf/1720.pdf>.
25. IDEF0 Function Modeling Method. Описание. Режим доступа – <http://www.idef.com/IDEF0.htm>.
26. РД IDEF0–2000 Руководящий документ. Методология функционального моделирования IDEF0. ИПК Издательство стандартов, 2000. [Электронный ресурс] Режим доступа – <http://www.nsu.ru/smk/files/idef.pdf>.
27. Галямина И. Г. Управление процессами: Учебник для вузов. Стандарт третьего поколения. – СПб.: Питер, 2013. – 304 с: ил.
28. Построение диаграмм потоков работ – WFD. [Электронный ресурс] Режим доступа – <http://studopedia.info/2-82371.html>.
29. Barker R. CASE-Method. Entity-Relationship Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
30. Дроздов А.Л., Коптелов А.К. Использование средств описания процессов при внедрении корпоративных информационных систем. Проблемы теории и практики управления, 2006, №10. [Электронный ресурс] Режим доступа – <http://businessprocess.narod.ru/index17.htm>.
31. IDEF1 Information Modeling Method. Описание. Режим доступа – <http://www.idef.com/IDEF1.htm>.
32. IDEF1X Data Modeling Method. Описание. Режим доступа – <http://www.idef.com/IDEF1x.htm>.

33. Integration Definition For Information Modeling (IDEF1X). Режим доступа – <http://www.idef.ru/documents/Idef1x.pdf>.
34. IDEF3 Process Description Capture Method. Описание. Режим доступа – <http://www.idef.com/IDEF3.htm>.
35. IDEF3 Process Description Capture Method report. Режим доступа – <http://www.staratel.com/iso/IDEF/IDEF3/Idef3.pdf>.
36. BPMN (Business Process Model and Notation). Описание. Режим доступа – <http://www.elma-bpm.ru/bpmn2/>.
37. ISO/IEC 19510:2013. Information technology – Object Management Group. Business Process Model and Notation. Режим доступа – http://www.iso.org/iso/catalogue_detail.htm?csnumber=62652.
38. Документация по методологии Microsoft Solutions Framework. Режим доступа – <https://msdn.microsoft.com/en-us/library/jj161047.aspx>.
39. Documents associated with Unified Modeling Language (UML), v2.4.1. Режим доступа – <http://www.omg.org/spec/UML/2.4.1/>.
40. Oracle Unified Method. Режим доступа – <http://www.oracle.com/partners/ru/products/applications/oracle-unified-method/get-started/index.html>.
41. ARIS (ARchitecture of Integrated Information Systems). Учебные материалы. Режим доступа – http://bainr.ru/study_materials_methodology_ARIS.html.
42. MathWorks. Центр компетенций. Модельно-ориентированное проектирование. Режим доступа – <http://matlab.ru/solutions/mbd/mbd>.
43. Manifesto for Agile Software Development. Режим доступа – <http://www.agilemanifesto.org/>.
44. Экстремальное программирование XP (eXtreme Programming). [Электронный ресурс] Режим доступа – <http://kibi.ru/xp/xp>.
45. The Scrum Guide. The definitive Guide to Scrum: The Rules of the Game. Ken Schwaber, Jeff Sutherland. Режим доступа – <http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>.
46. Х. Книберг, М. Скарин. Scrum и Канбан: выжимаем максимум. C4Media, Издательство InfoQ.com. Режим доступа – <http://www.infoq.com/resource/news/2010/01/kanban-scrum-minibook/en/resources/KanbanAndScrum-Russian.pdf>.
47. Сайт консорциума DSDM (Dynamic Systems Development Method). Режим доступа – <http://www.dsdm.org/>.
48. CyberGuru.ru. Новые методологии программирования. [Электронный ресурс] Режим доступа – <http://www.cyberguru.ru/programming/programming-theory/coding-methodology-new-page15.html>.
49. Тайчи Оно. Производственная система Тойоты. Уходя от массового производства. Библиотека ИКСИ. Издательство: Институт комплексных стратегических исследований. 2008. – 194 с.
50. Руководство к своду знаний по управлению проектами PMI PMBOK® Guide 5th Edition (2012). Режим доступа – http://startupseminar.ru/_ld/0/17_301907_2D9D3_pm.pdf.
51. Куперштейн В. И. Microsoft® Project 2013 в управлении проектами. – СПб.: БХВ-Петербург, 2014. – 432 с: ил.

52. Dia. Documentation. Manual. [Электронный ресурс]. Режим доступа – <http://dia-installer.de/doc/en/>.
53. Dia Tutorial (Учебник Dia). [Электронный ресурс]. Режим доступа – <http://younglinux.info/book/export/html/169>.
54. Project Planning with OpenProj the open source Project Management software Jürgen Bruns & Associates. [Электронный ресурс]. Режим доступа – <http://faculty.caes.uga.edu/pthomas/hort4091.web/Instructions.pdf>.
55. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite / С.В. Маклаков. – М.: ДИАЛОГ-МИФИ, 2005. – 432 с.
56. Официальный русскоязычный сайт проекта Ramus. Режим доступа – <http://ramussoftware.com/>.
57. Ramus 1.2.5 (Моделирование бизнес-процессов) [Электронный ресурс]. Режим доступа – <http://www.twirpx.com/file/132655/>.
58. Официальный сайт проекта CA ERwin® Data Modeling. Режим доступа – <http://erwin.com/worldwide/russian-russia>.
59. Проектирование информационных систем с CA ERwin Modeling Suite 7.3 :учебное пособие / В.И. Горбаченко, Г.Ф. Убиенных, Г.В. Бобрышева – Пенза: Изд-во ПГУ, 2012. – 154 с.
60. Официальный сайт консалтинговой компании Business Process Solutions. Режим доступа – <http://bps.org.ua/aris.html>.
61. ARIS Community. Table of contents of ARIS Express help. [Электронный ресурс]. Режим доступа – <http://www.ariscommunity.com/help/aris-express>.
62. Сухарев М. С, Монахов Ю. М. Модель оценки функциональной устойчивости бизнес-процессов в условиях развитой телекоммуникационной инфраструкту-ры//Перспективные технологии в средствах передачи информации: Материалы IV Международной научно-технической конференции (г. Суздаль, 29 июня 1 июля 2011 года)/ Владимир-Суздаль: Изд-во ГОУ ВПО «Владимирский государственный университет им. А. Г. и Н. Г. Столетовых», 2011. 271 с.
63. Серова, Е.Г. Современные методологические и инструментальные подходы моделирования бизнес-задач / Е.Г. Серова // Int. Journal Information Technologies and Knowledge Decision Making and Business Intelligence Strategies and Techniques. 2008. - № 2. [Электронный ресурс]. Режим доступа – <http://simulation.su/uploads/files/default/2008-serova-1.pdf>.
64. Официальный сайт The AnyLogic Company. Учебные материалы. [Электронный ресурс]. Режим доступа – <http://www.anylogic.ru/books>.

Варианты заданий к практическим занятиям и лабораторным работам

Вариант предметной области выдается на группу, состоящую из 3х человек.

Варианты предметных областей

- Вариант 0. Музейно-археологический комплекс.
- Вариант 1. Библиотека научно-исследовательского института.
- Вариант 2. Регистратура лечебного учреждения.
- Вариант 3. Расчетный центр услуг жилищно-коммунального хозяйства (ЖКХ).
- Вариант 4. Реестр бюро технической инвентаризации (БТИ).
- Вариант 5. Организация перевозок опасных грузов.
- Вариант 6. Картотека государственной инспекции безопасности дорожного движения (ГИБДД).
- Вариант 7. Реестр земель сельскохозяйственного назначения.
- Вариант 8. Информационное обслуживание пассажиров авиалиний.
- Вариант 9. Почтовое отделение.

***Замечание:** студент вправе, предложить свой вариант предметной области и принять его к исполнению, после согласования и утверждения ведущим преподавателем.*

Варианты моделей жизненного цикла

- Вариант 0. Поэтапная модель с промежуточным контролем.
- Вариант 1. Инкрементная модель жизненного цикла.
- Вариант 2. Спиральная модель жизненного цикла.

Варианты моделей жизненного цикла распределяются внутри сформированной группы, состоящей из 3х человек.

**Образец оформления и содержания отчета по лабораторной работе и
практическому занятию**

Практическое задание и лабораторная работа №_____

Тема:

Цель:

Основная часть

1. Отчет о выполнении практического задания
- ...
2. Отчет о выполнении задания на лабораторную работу

...

Выводы

...

Список литературы и информационных источников

...

Приложения

Приложение 3

Образец единого титульного листа к отчетам по лабораторным работам и практическим занятиям

Министерство образования и науки Российской Федерации
Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

Кафедра Информационных систем

Сводный отчет по лабораторному практикуму
по дисциплине «Методы и средства проектирования информационных систем и технологий»

| № п/п | Оценка выполнения | | | | Дата | Подпись |
|----------|-------------------|----|----|------|------|---------|
| | Теория | Пз | Лз | Итог | | |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| зачет | | | | | | |

Выполнил: студент(ка) группы ____
ФИО

Принял: должность ФИО

г.Севастополь
20__ г.