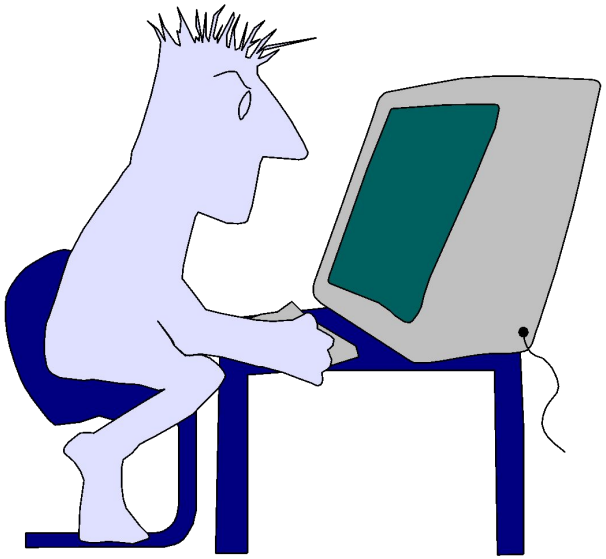
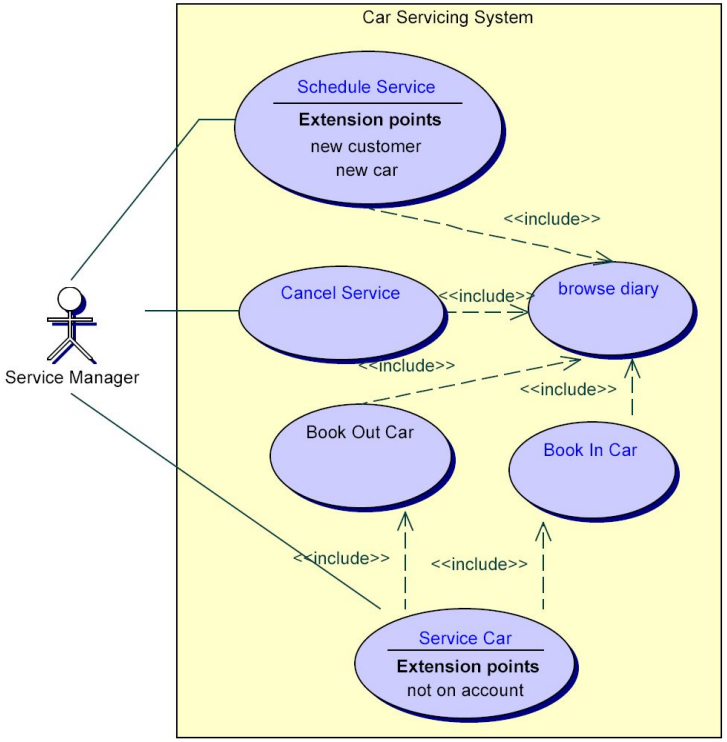
Математическое моделирование в энергетике

Тема: технология объектно-ориентированного моделирования. Ключевые понятия

Лекция №3

Модель (model) - абстракция физической системы, рассматриваемая с определенной точки зрения и представленная на некотором языке или в графической форме.

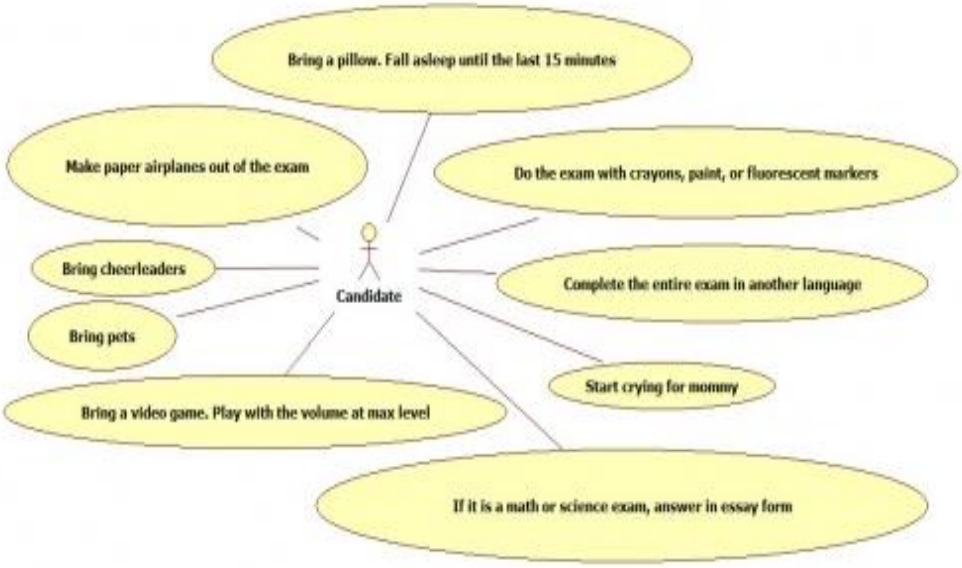


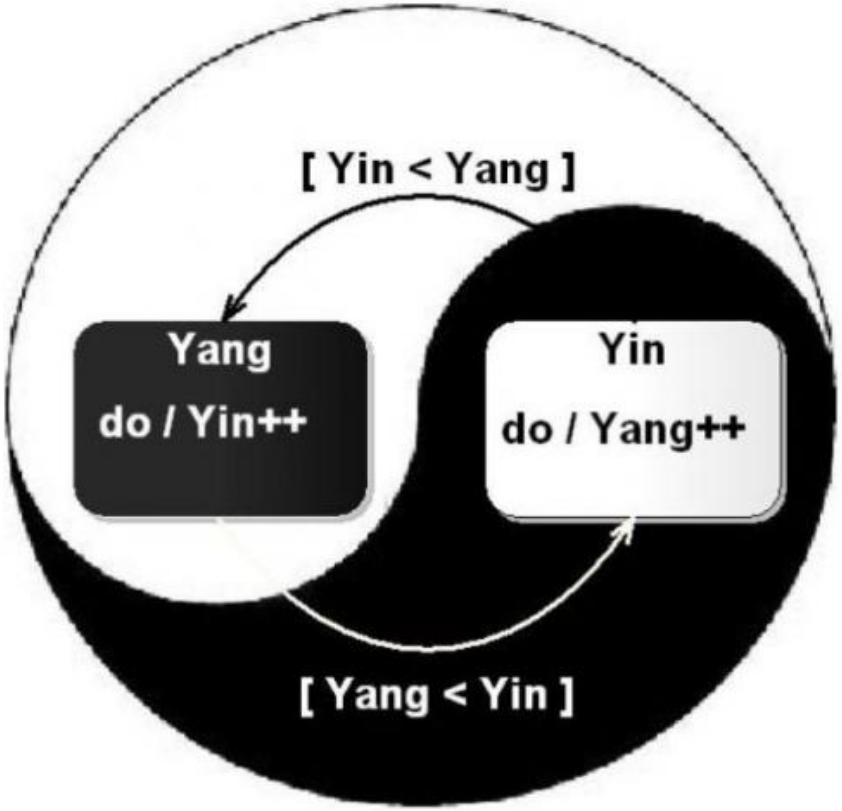


Отсутствие моделей при разработке ПО:

- ✔ Не позволяет справиться с растущей сложностью разрабатываемых программных систем;
- ✔ Не позволяет эффективно управлять разработкой в условиях изменяющихся требований;
- ✔ Создает барьеры непонимания: аналитик не понимает руководителя проекта, разработчик - аналитика, тестировщик - разработчика и пр.
- ✔ Не позволяет обеспечить контроль изменений в процессе выполнения работ;
- ✔ Не позволяет избежать субъективности в оценке качества разрабатываемых продуктов.

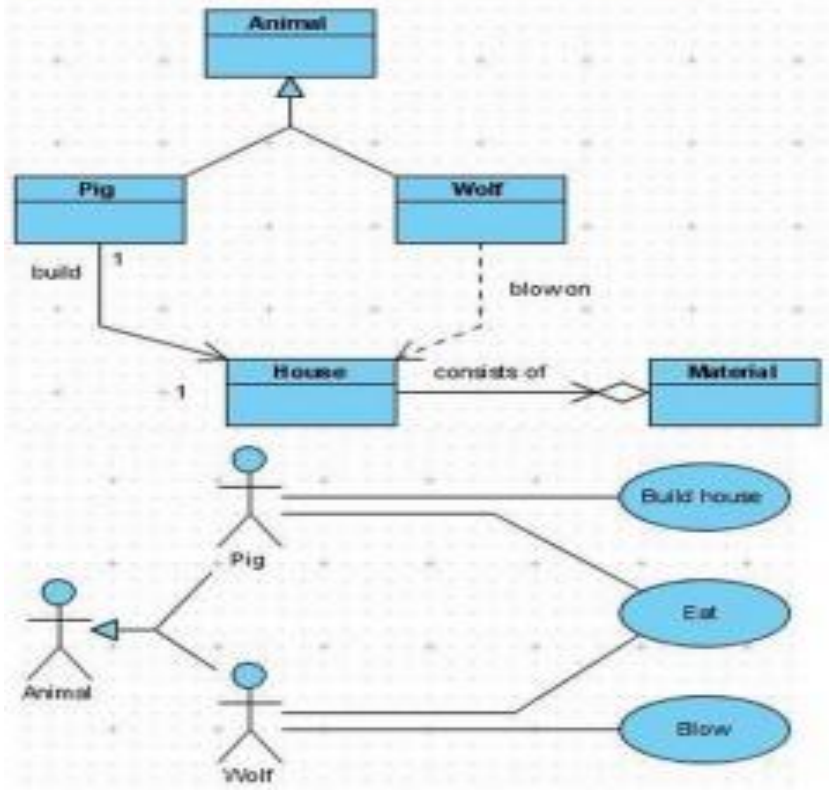
Методология объектно-ориентированного моделирования реализуется с использованием унифицированного языка моделирования Unified Modeling Language (UML).



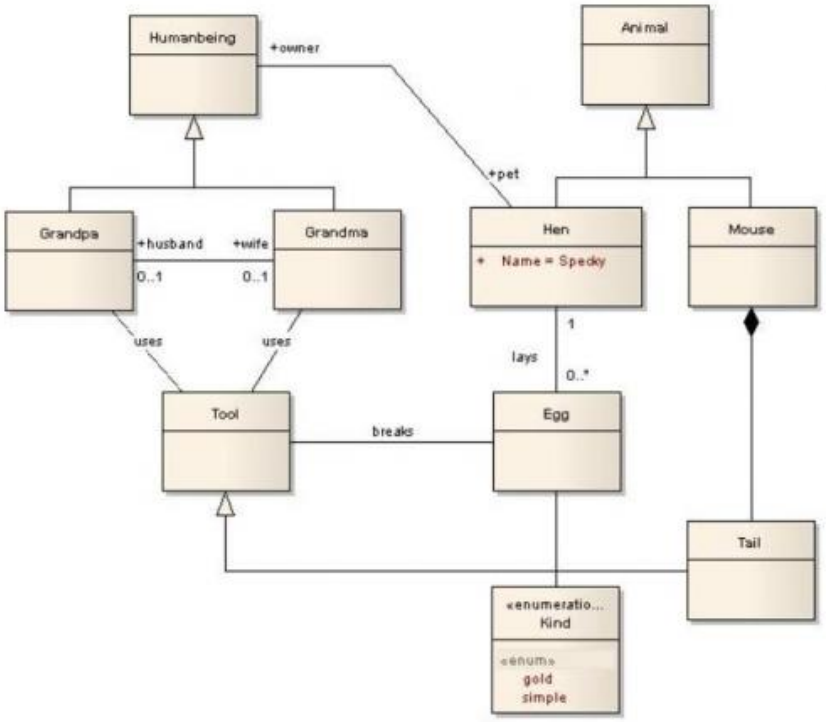


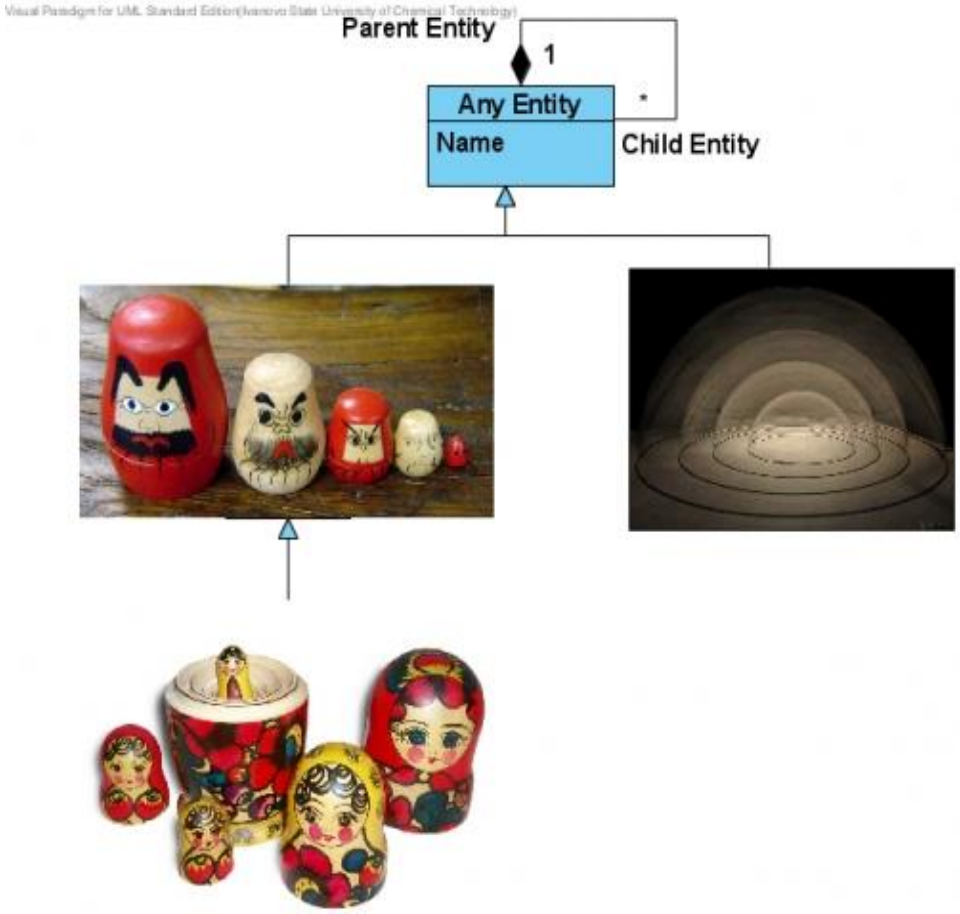
UML (унифицированный язык моделирования) - язык графического описания для объектного моделирования в области разработки программного обеспечения математических моделей.

UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML моделью.

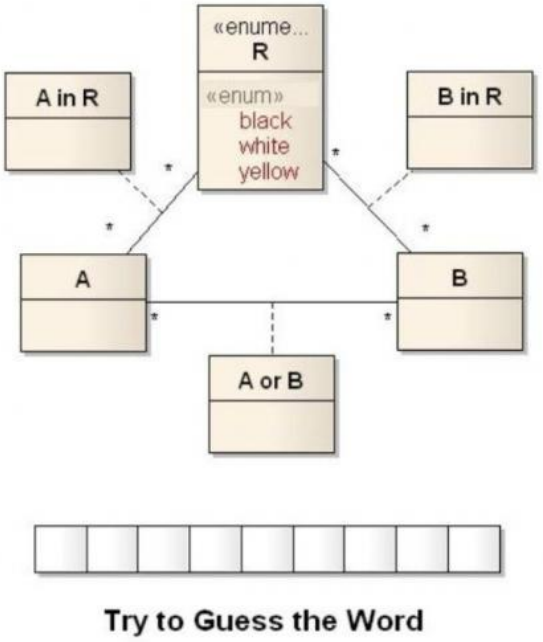


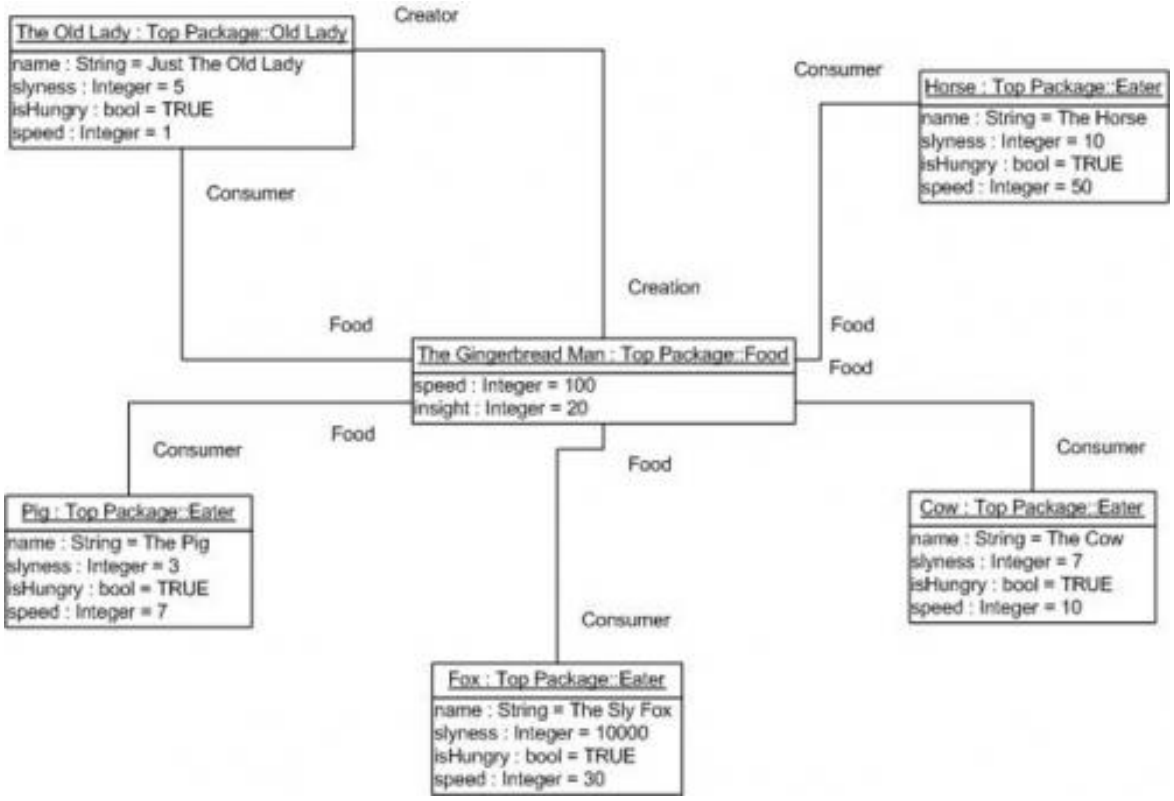
Использование UML не ограничивается моделированием программного обеспечения математических моделей. Его также используют для моделирования процессов, системного проектирования и отображения организационных структур.





UML позволяет разработчикам ПО достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (generalization), объединение (aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.





UML объектно-ориентированный, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках;

UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;

Диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;

UML расширяет и позволяет вводить собственные текстовые и графические стереотипы;

UML – это стандарт, который получил широкое распространение и динамично развивается.

ОСНОВНЫЕ ПОНЯТИЯ ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ В UML

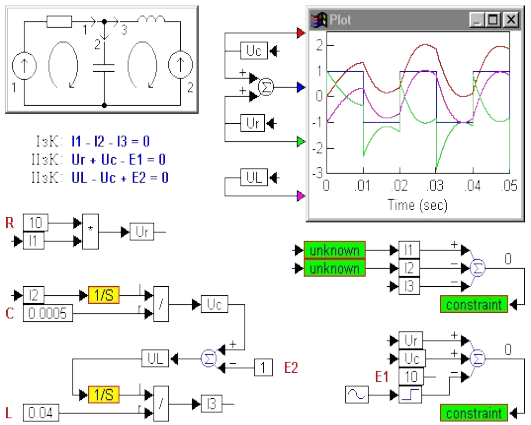
Модель проекта - совокупность подмоделей структуры и поведения;

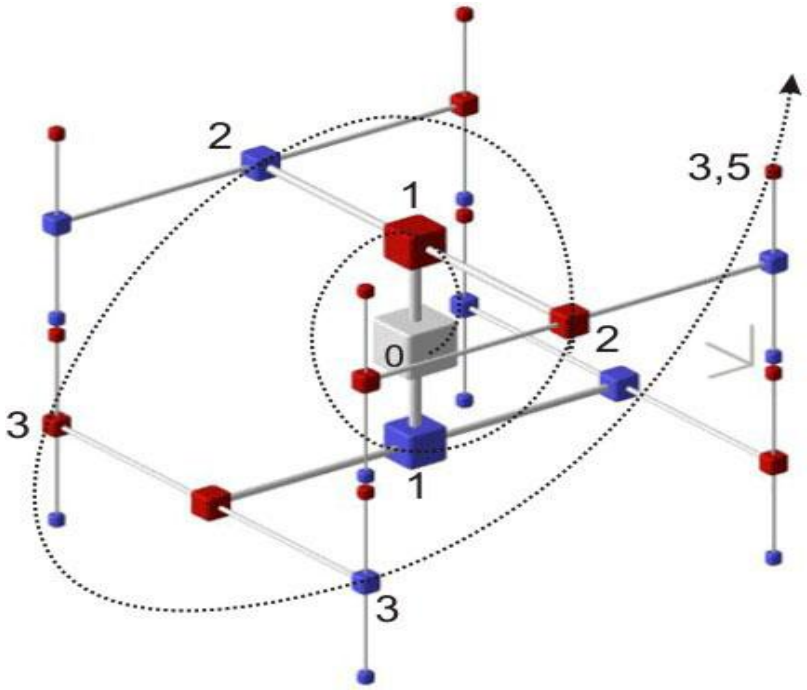
Каждая подмодель представлена набором диаграмм;

Подмодели согласованы между собой.

UML включает 3 вида объектов:

- ✔ сущности;
- ✔ отношения;
- ✔ диаграммы.

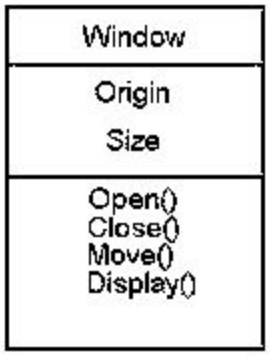




ОСНОВНЫЕ СУЩНОСТИ UML

Интерфейс:

Класс:



Примечание:

Прецедент (use case):

ОСНОВНЫЕ ТИПЫ ДИАГРАММ UML

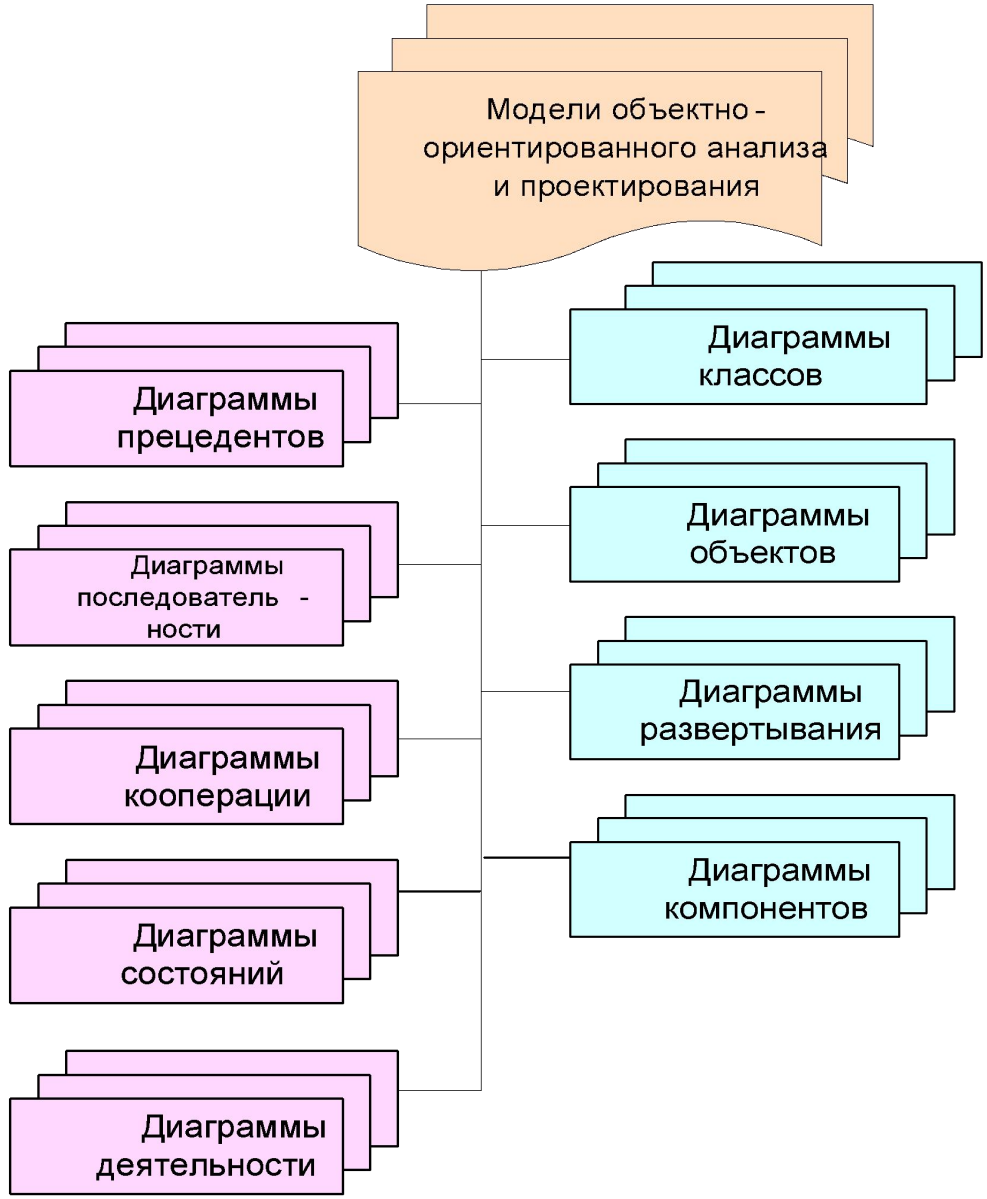
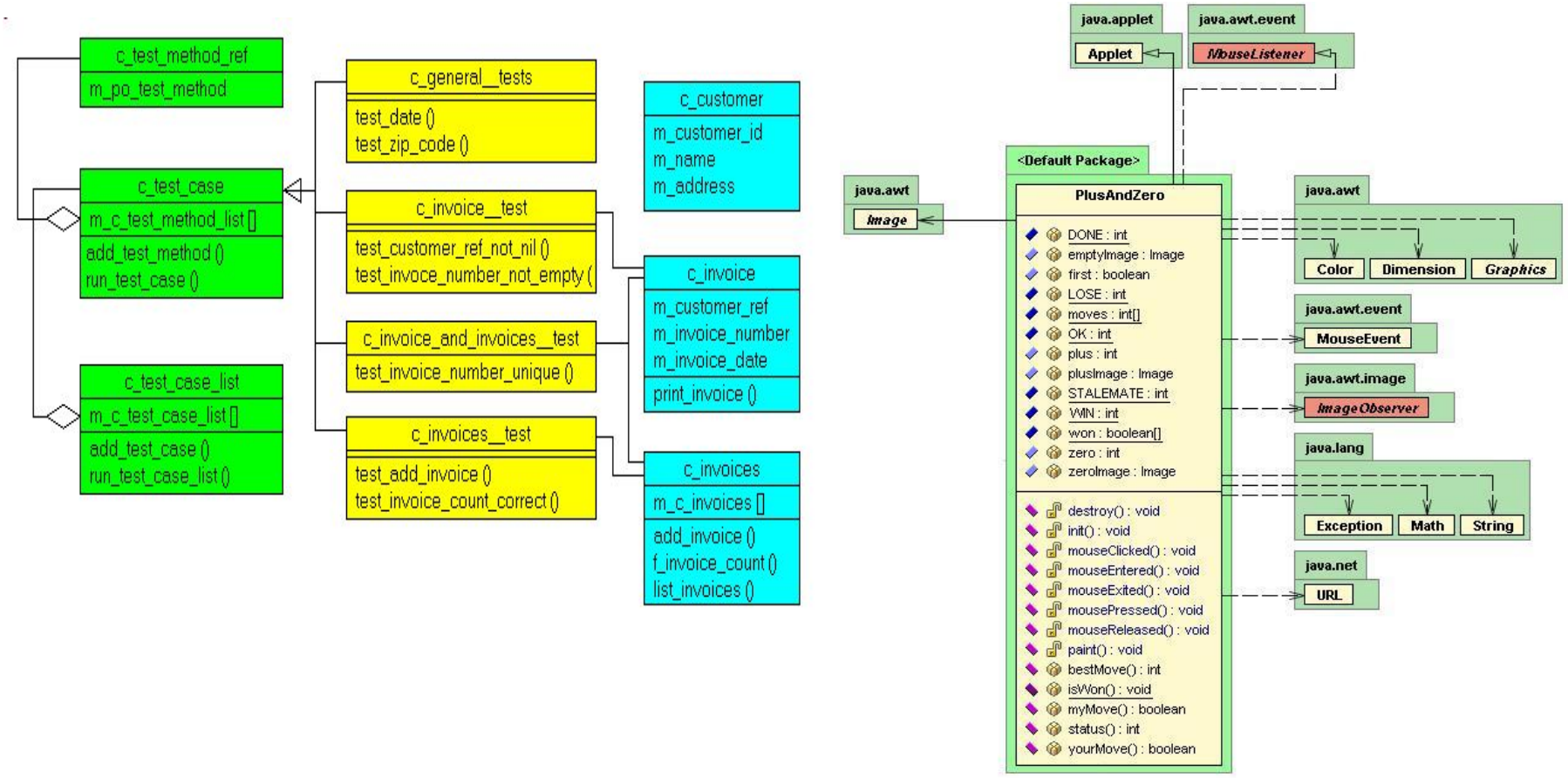
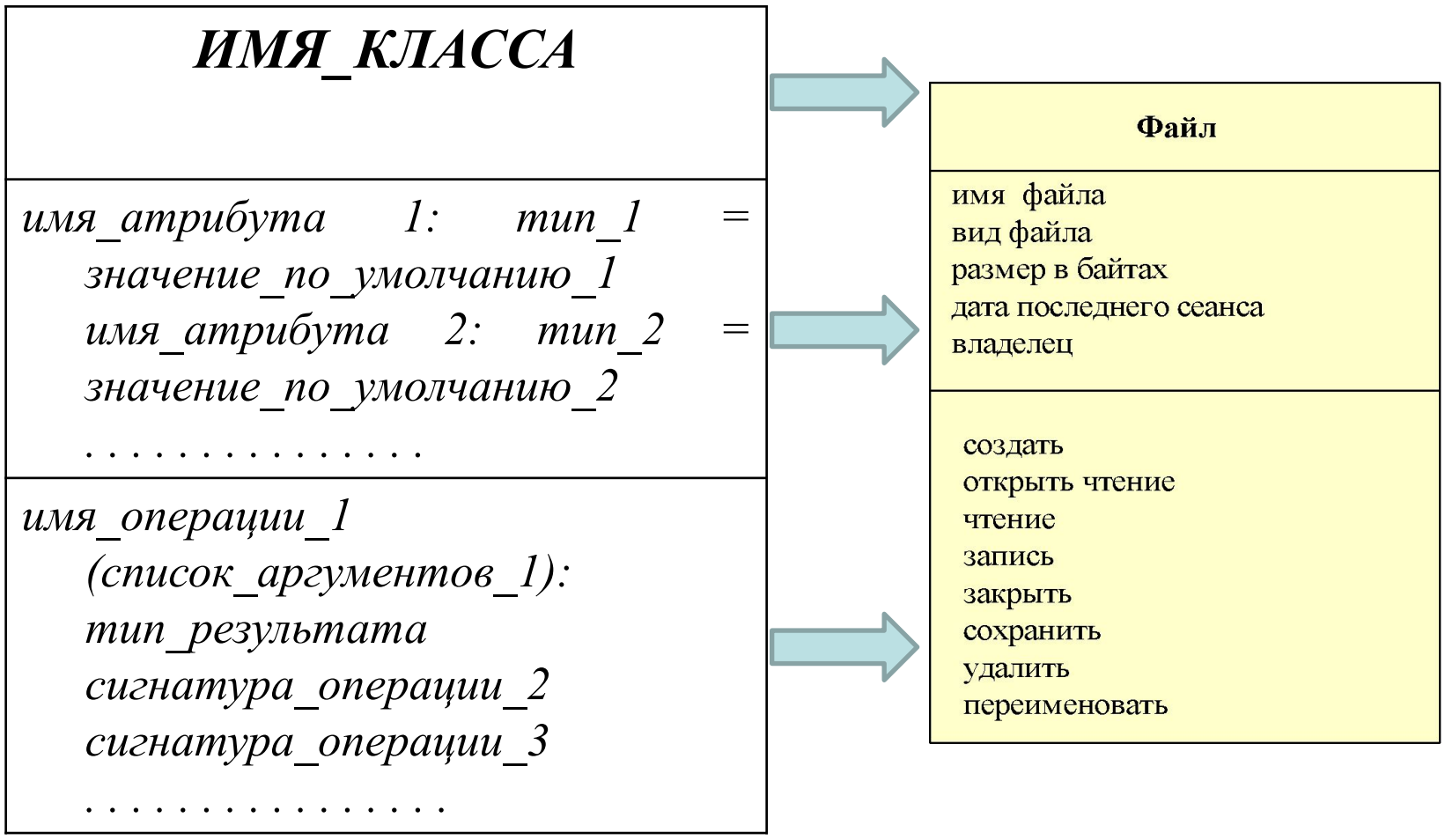
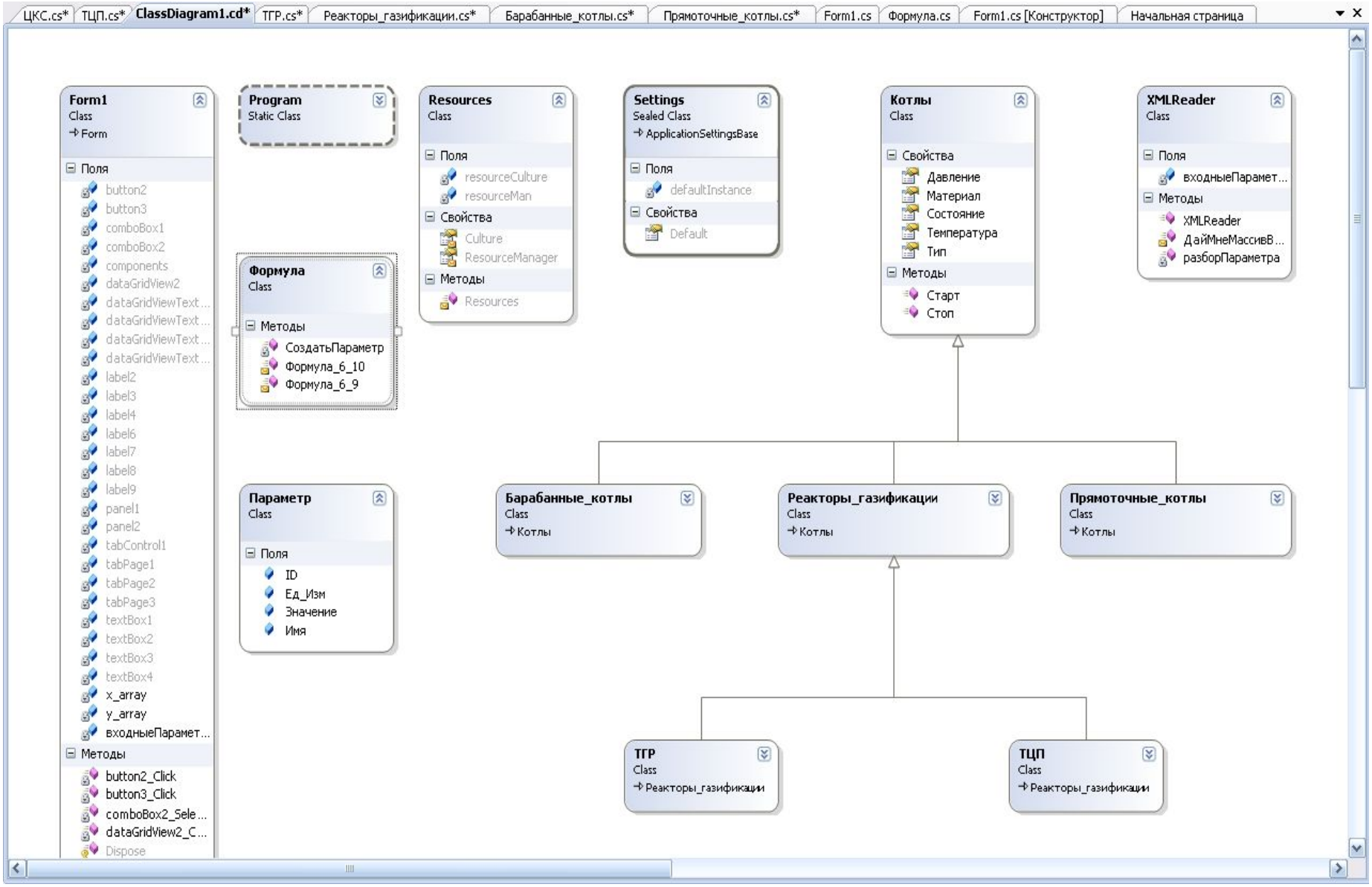


Диаграмма классов

Диаграмма классов, Class diagram - статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.



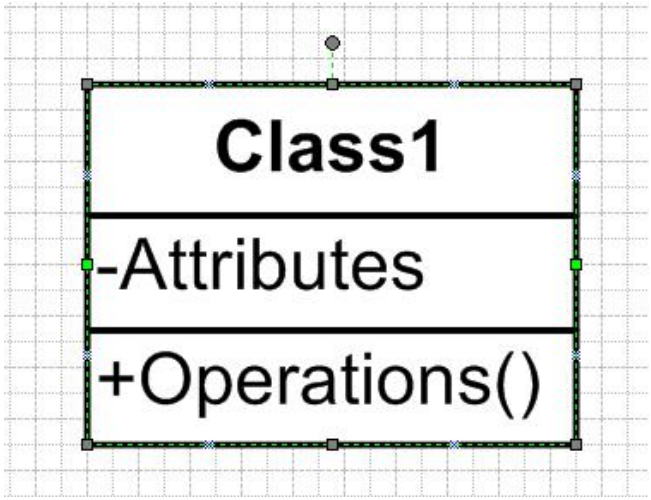

ИМЯ_КЛАССА
имя_атрибута 1: тип_1 =
значение_по_умолчанию_1
имя_атрибута 2: тип_2 =
значение_по_умолчанию_2
. . . . . .
имя_операции_1
(список_аргументов_1):
тип_результата
сигнатура_операции_2
сигнатура_операции_3
. . . . . .
Файл
имя файла
вид файла
размер в байтах
дата последнего сеанса
владелец
создать
открыть чтение
чтение
запись
закрыть
сохранить
удалить
переименовать


ЦКС.cs*
ТЦП.cs*
ClassDiagram1.cd*
ТГР.cs*
Реакторы_газификации.cs*
Барабанные_котлы.cs*
Прямоточные_котлы.cs*
Form1.cs
Формула.cs
Form1.cs [Конструктор]
Начальная страница
Form1
Class
Form
Поля
button2
button3
comboBox1
comboBox2
components
dataGridView2
dataGridViewText...
dataGridViewText...
dataGridViewText...
dataGridViewText...
label2
label3
label4
label6
label7
label8
label9
panel1
panel2
tabControl1
tabPage1
tabPage2
tabPage3
textBox1
textBox2
textBox3
textBox4
x_array
y_array
входныеПарамет...
Методы
button2_Click
button3_Click
comboBox2_Sele...
dataGridView2_C...
Dispose
Program
Static Class
Resources
Class
Поля
resourceCulture
resourceMan
Свойства
Culture
ResourceManager
Методы
Resources
Settings
Sealed Class
ApplicationSettingsBase
Поля
defaultInstance
Свойства
Default
Котлы
Class
Свойства
Давление
Материал
Состояние
Температура
Тип
Методы
Старт
Стоп
XMLReader
Class
Поля
входныеПарамет...
Методы
XMLReader
ДайМнеМассивВ...
разборПараметра
Формула
Class
Методы
СоздатьПараметр
Формула_6_10
Формула_6_9
Параметр
Class
Поля
ID
Ед_Изм
Значение
Имя
Барабанные_котлы
Class
Котлы
Реакторы_газификации
Class
Котлы
Прямоточные_котлы
Class
Котлы
ТГР
Class
Реакторы_газификации
ТЦП
Class
Реакторы_газификации

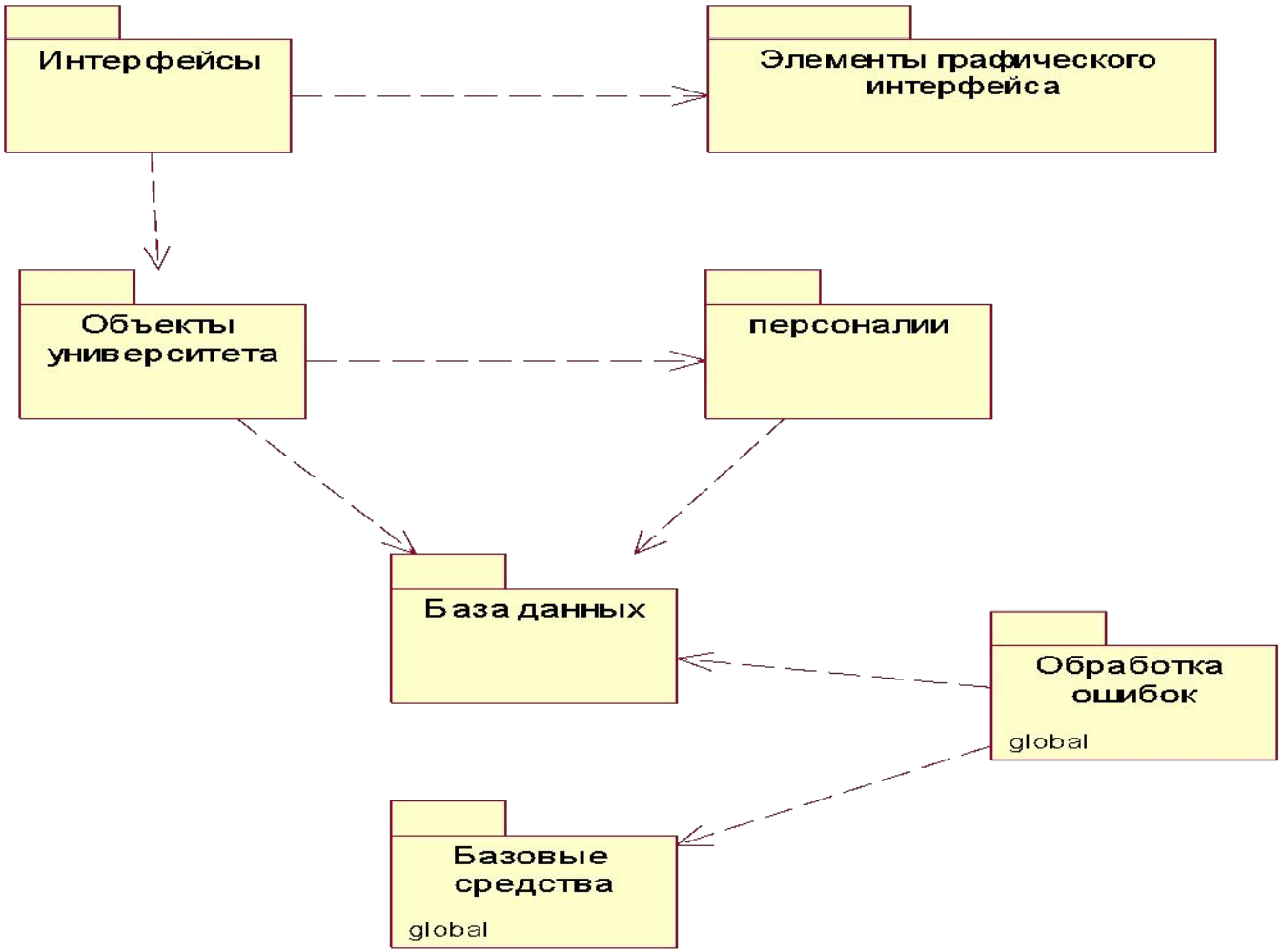
Атрибуты и операции класса

Атрибут - это значение, характеризующее объект в его классе. Примеры атрибутов: тип котла, тип турбины (атрибуты объектов класса энергоблок); имя, возраст, вес (атрибуты объектов класса человек) и т.д.

Операция - это функция (или преобразование), которую можно применять к объектам данного класса. Примеры операций: проверить, запустить, остановить, открыть_на_чтение, читать, закрыть и т.п.



Если система содержит большое количество классов, они могут быть объединены в пакеты, представляющие архитектуру системы



Отношения между классами (пиктограммы)

- Ассоциация

  - Постоянное, структурное отношение, “has a”
  - Непрерывная линия со стрелкой (в случае направленности ассоциации)
- Зависимость

  - Временное отношение, “uses a”
  - Пунктирная линия со стрелкой
- Обобщение

  - Наследование, “is a”
  - Непрерывная линия со стрелкой (треугольной)
- Использование

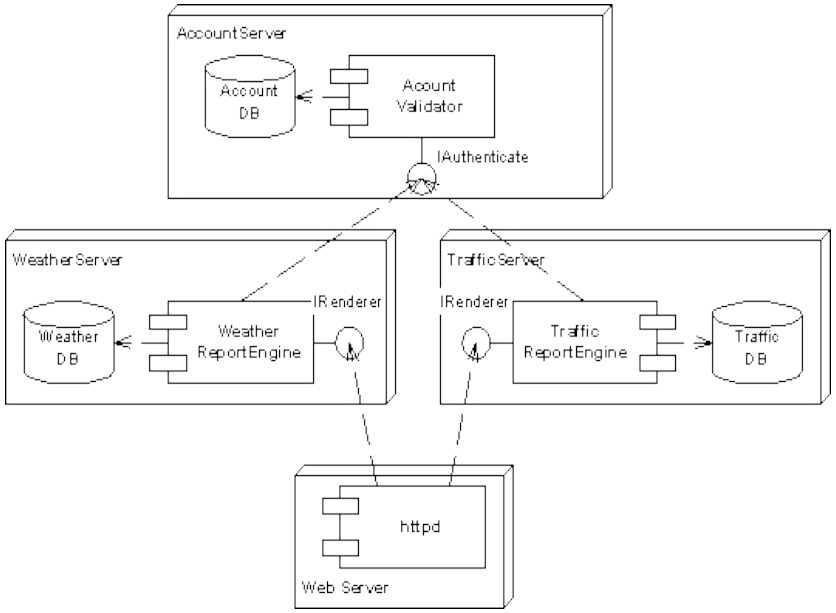
  - Пунктирная линия со стрелкой (треугольной)

Типы отношений

Отношение	Функция
Ассоциация *(Association)*	Описание семантических связей между экземплярами классов
Зависимость *(Dependency)*	Отношение, существующее между двумя элементами модели
Обобщение *(Generalization)*	Отношение между общим описанием и более специфическими его разновидностями; используется при наследовании и для объявления полиморфных типов
Реализация *(Realization)*	Отношение между спецификацией и ее реализацией
Использование *(Usage)*	Ситуация, когда для корректной работы одному элементу системы необходим другой элемент

Диаграмма компонентов

Диаграмма компонентов, Component diagram - статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.



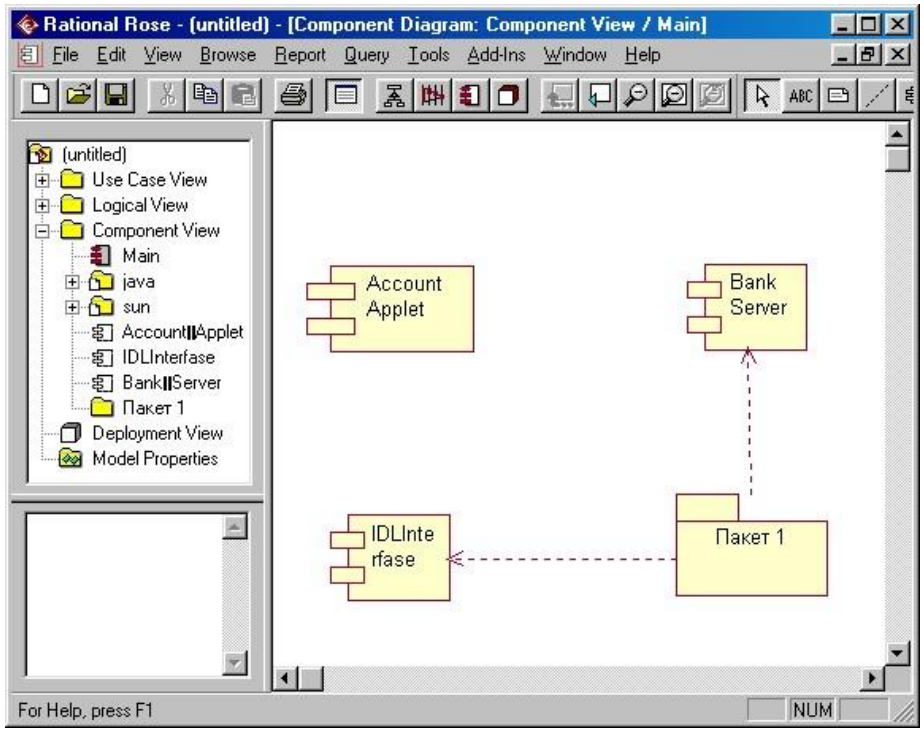


Диаграмма композитной/составной структуры

Диаграмма композитной/составной структуры, Composite structure diagram - статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.

Подвидом диаграмм композитной структуры являются диаграммы кооперации (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования.

Диаграммы композитной структуры могут использоваться совместно с диаграммами классов.

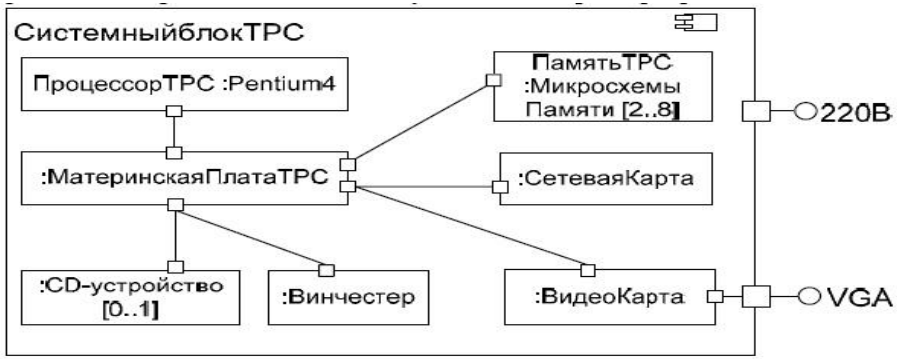
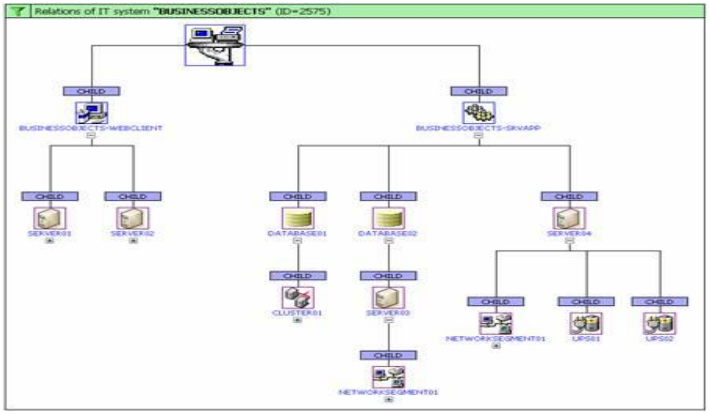


Диаграмма развёртывания

Диаграмма развёртывания, Deployment diagram - служит для моделирования работающих узлов (аппаратных средств, англ. node) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (англ. artifact), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации.



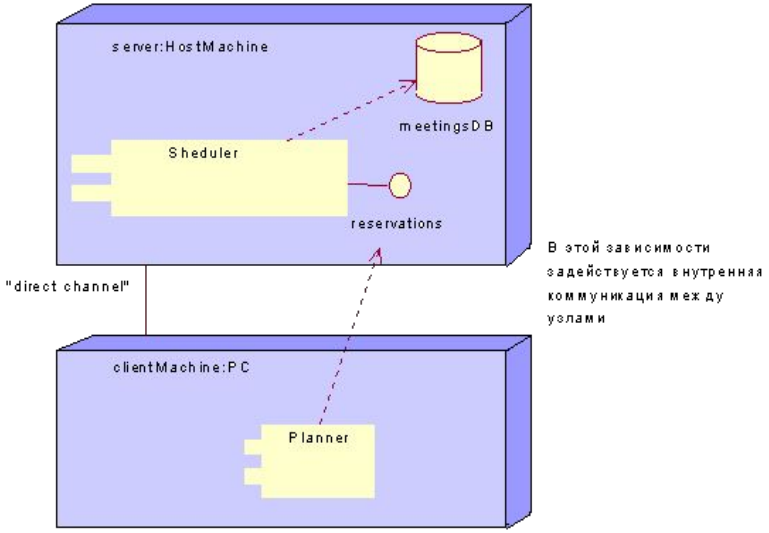
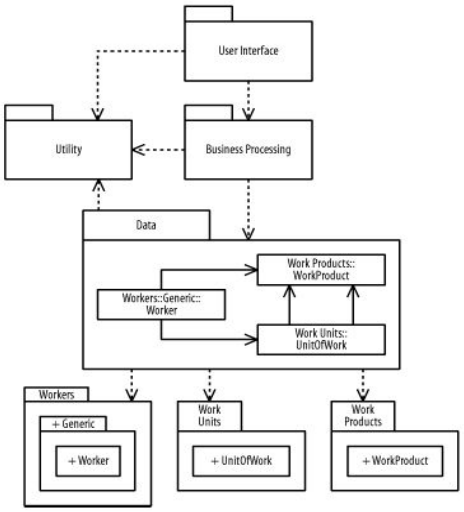


Диаграмма объектов

Диаграмма объектов, Object diagram - демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами.



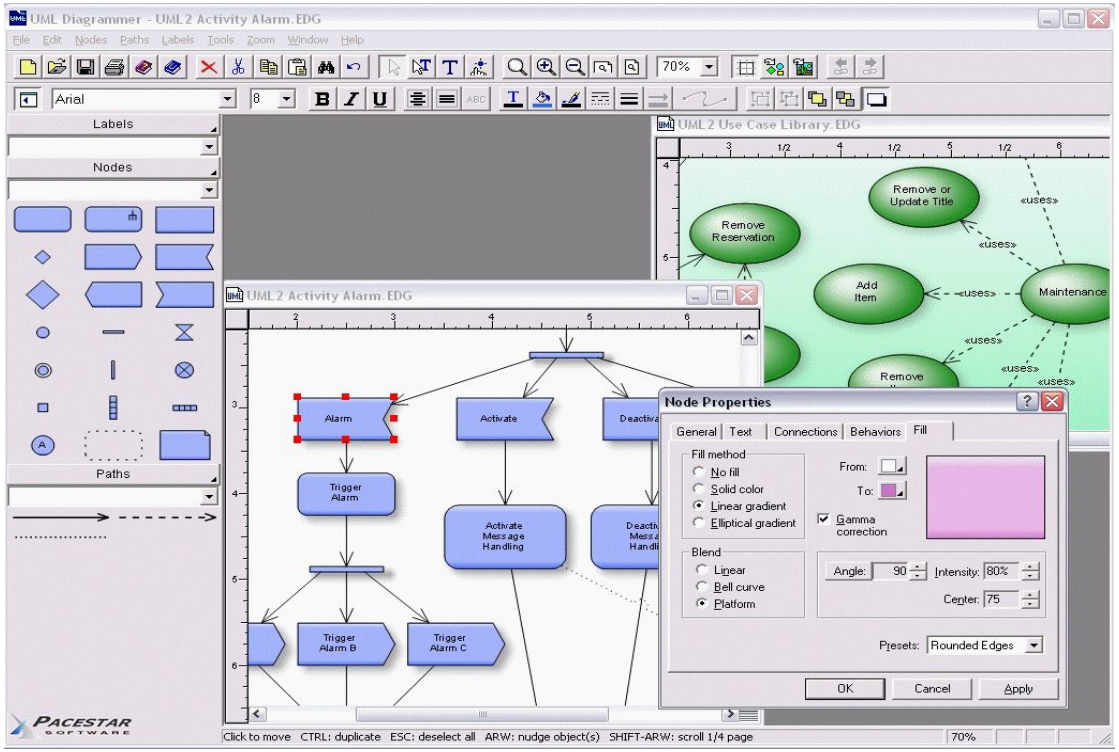
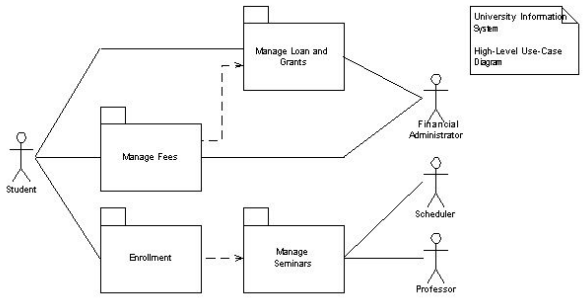


Диаграмма пакетов

Диаграмма пакетов, Package diagram - структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Жёсткого разделения между разными структурными диаграммами не проводится, поэтому данное название предлагается исключительно для удобства и не имеет семантического значения (пакеты и диаграммы пакетов могут присутствовать на других структурных диаграммах). Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы.



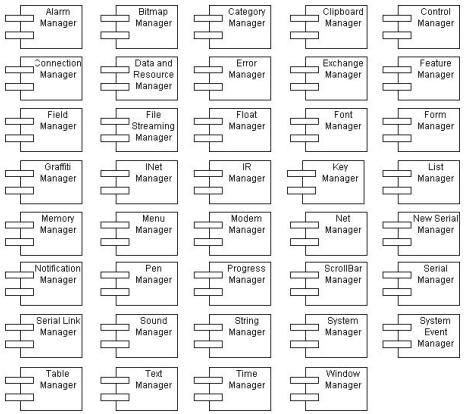


Диаграмма деятельности

Диаграмма деятельности, Activity diagram - диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов - вложенных видов деятельности и отдельных действий (англ. action), соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. Аналогом диаграмм деятельности являются схемы алгоритмов по ГОСТ 19.701-90.

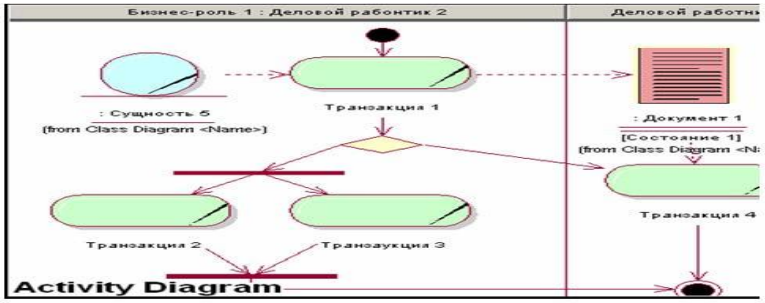


Диаграмма автомата

Диаграмма автомата, State Machine diagram (диаграмма конечного автомата, диаграмма состояний) - диаграмма, на которой представлен конечный автомат с простыми состояниями, переходами и композитными состояниями.

Конечный автомат (англ. State machine) - спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

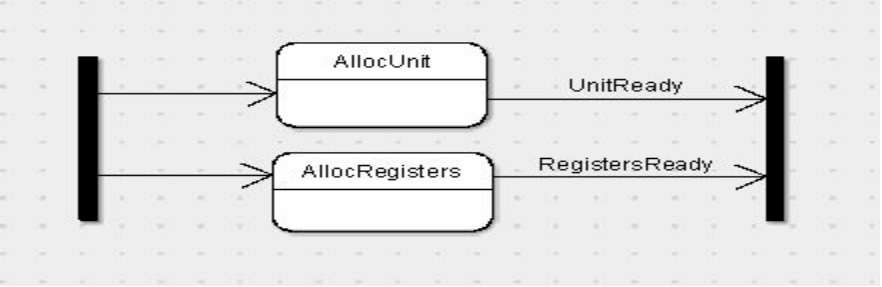
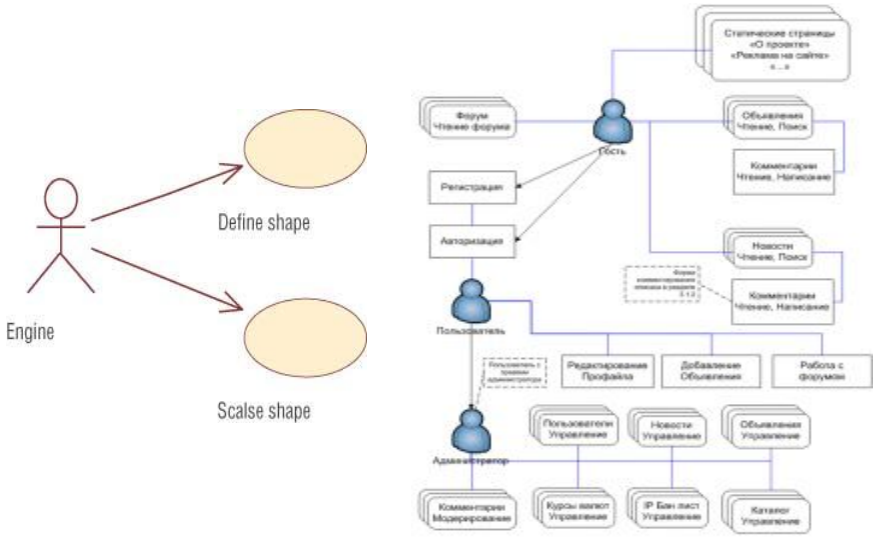
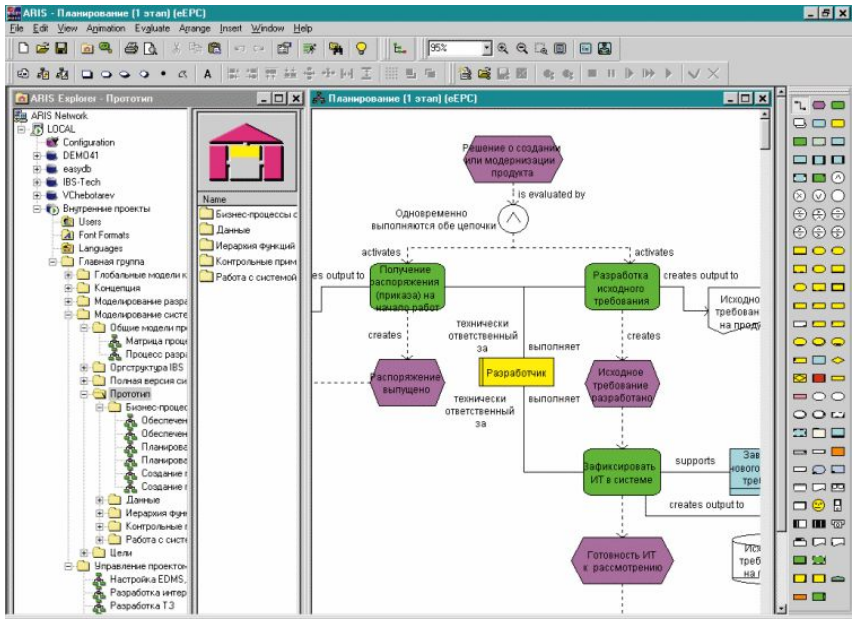


Диаграмма прецедентов

Диаграмма прецедентов, Use case diagram (диаграмма вариантов использования) - диаграмма, на которой отражены отношения, существующие между актерами и прецедентами.

Основная задача - представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.





Диаграммы коммуникации и последовательности

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую. Диаграмма коммуникации, Communication diagram (в UML 1.x - диаграмма кооперации, collaboration diagram) - диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. Диаграмма последовательности, Sequence diagram - диаграмма, на которой изображено упорядоченное во времени взаимодействие объектов. В частности, на ней изображаются участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются.

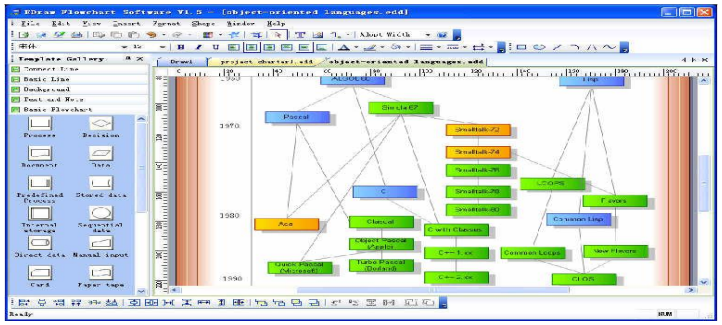


Диаграмма обзора взаимодействия

Диаграмма обзора взаимодействия, Interaction overview diagram - разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления.

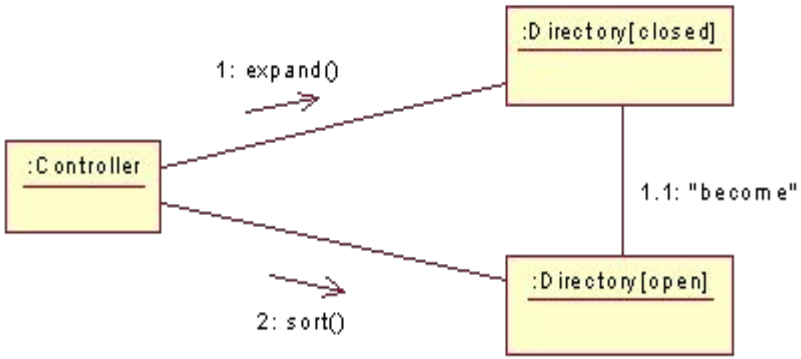


Рис. 33. Метаморфоза

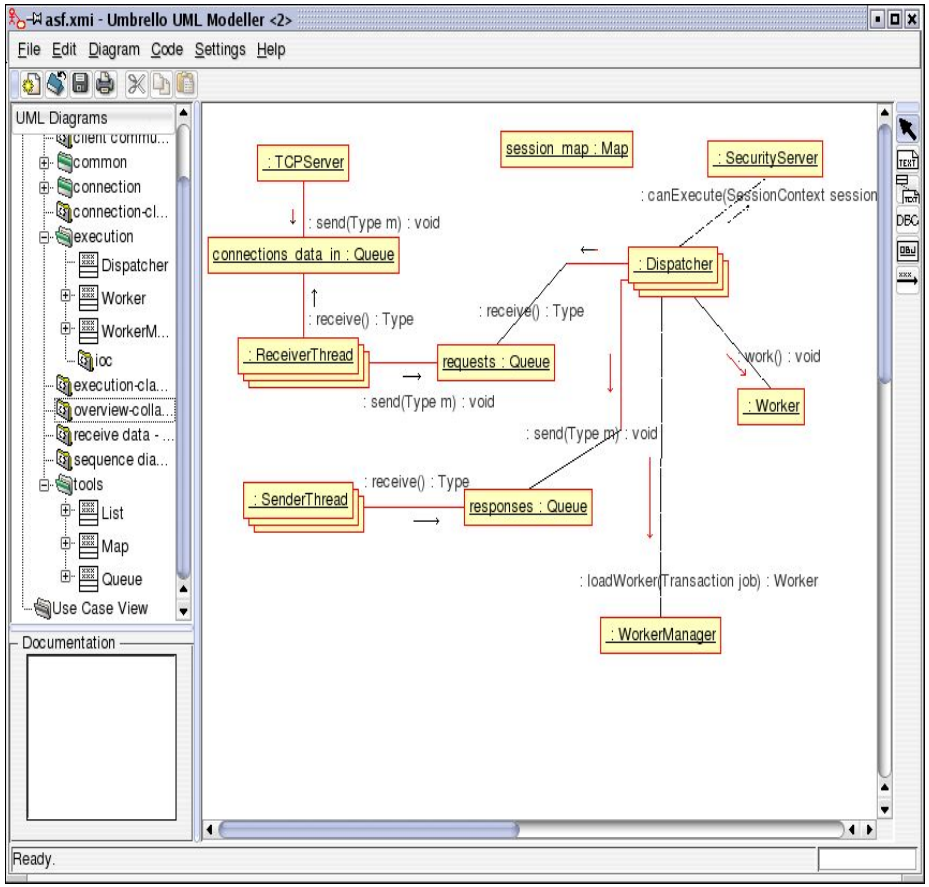
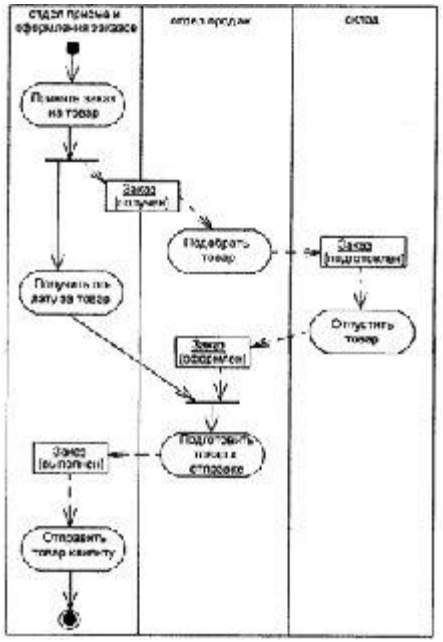
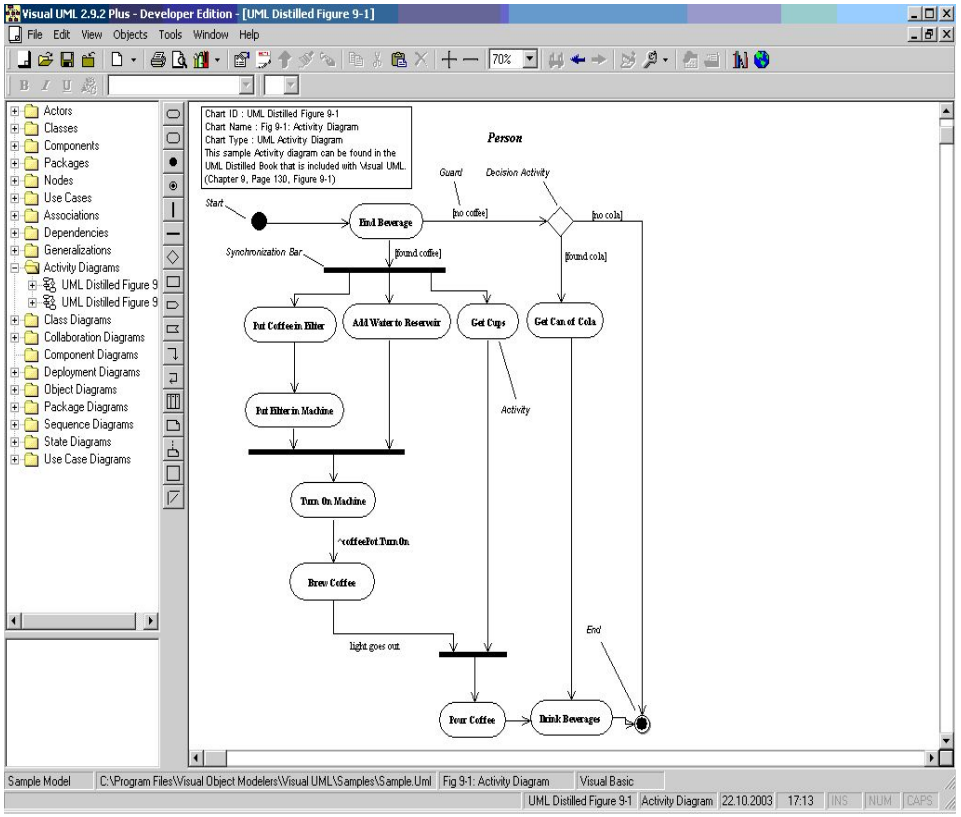


Диаграмма синхронизации

Диаграмма синхронизации, Timing diagram - альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени. Может быть полезна в приложениях реального времени.





Методология Rational Unified Process

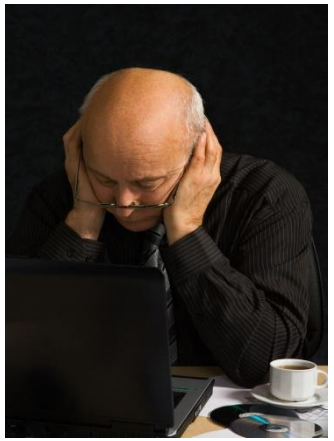
Rational Unified Process (RUP) - это процесс разработки программного обеспечения. Его цель состоит в том, чтобы гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого графика и бюджета выполнения. RUP обеспечивает строгий подход к решению задач проектирования и ответственности разработчиков.

Проблемы обходятся на два-три порядка дороже, если они возникают и устраняются после развертывания программного обеспечения.

Для достижения целей в рамках установленных ресурсов необходимы контроль и управление качеством.

Оценка качества всех действий и их участников выполняется с использованием объективных измерений и критериев.

Испытание (тестирование) качества производится на всех итерациях жизненного цикла математической модели.



Критерии качества программных реализаций моделей

Полнота - все необходимые части программы должны быть представлены и полностью реализованы.

Надёжность - отсутствие отказов и сбоев в работе программ, а также простота исправления дефектов и ошибок:

Эффективность - насколько рационально программа относится к ресурсам (память, процессор) при выполнении своих задач.

Краткость - отсутствие лишней, дублирующейся информации.

Портируемость - лёгкость в адаптации программы к другому окружению: другой архитектуре, платформе, операционной системе или её версии.

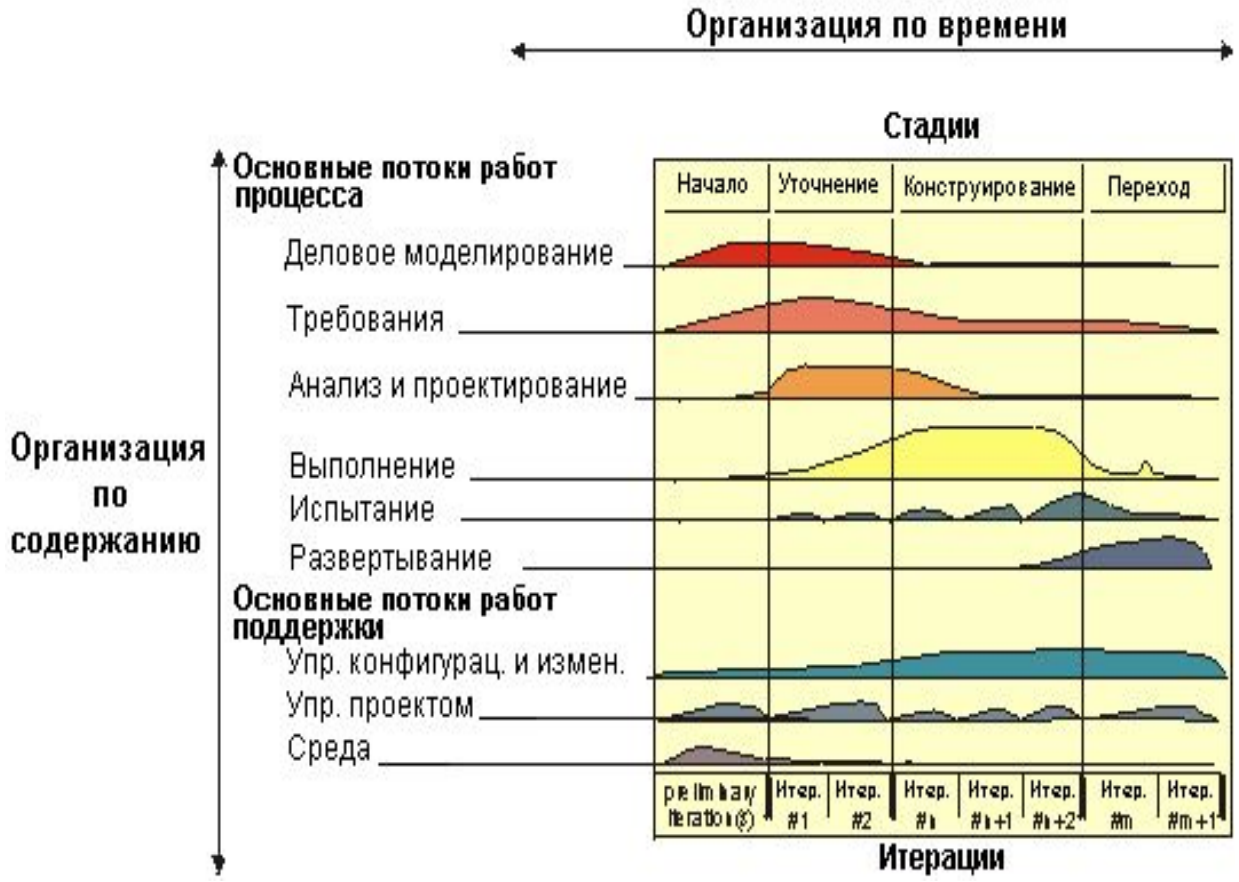
Сопровождаемость - насколько сложно изменить программу для удовлетворения новых требований. Это требование также указывает, что программа должна быть хорошо документирована.

Тестируемость - Позволяет ли программа выполнить проверку приёмочных характеристик, поддерживается ли возможность измерения производительности.

Цель управления качеством:

Гарантировать получение высококачественной программной системы, отвечающей потребностям заказчиков, в пределах предсказуемого временного графика и бюджета.

СХЕМА ОРГАНИЗАЦИИ RUP:



Структура жизненного цикла модели

Жизненный цикл разбивается на циклы, каждый из которых работает над новым поколением модели.

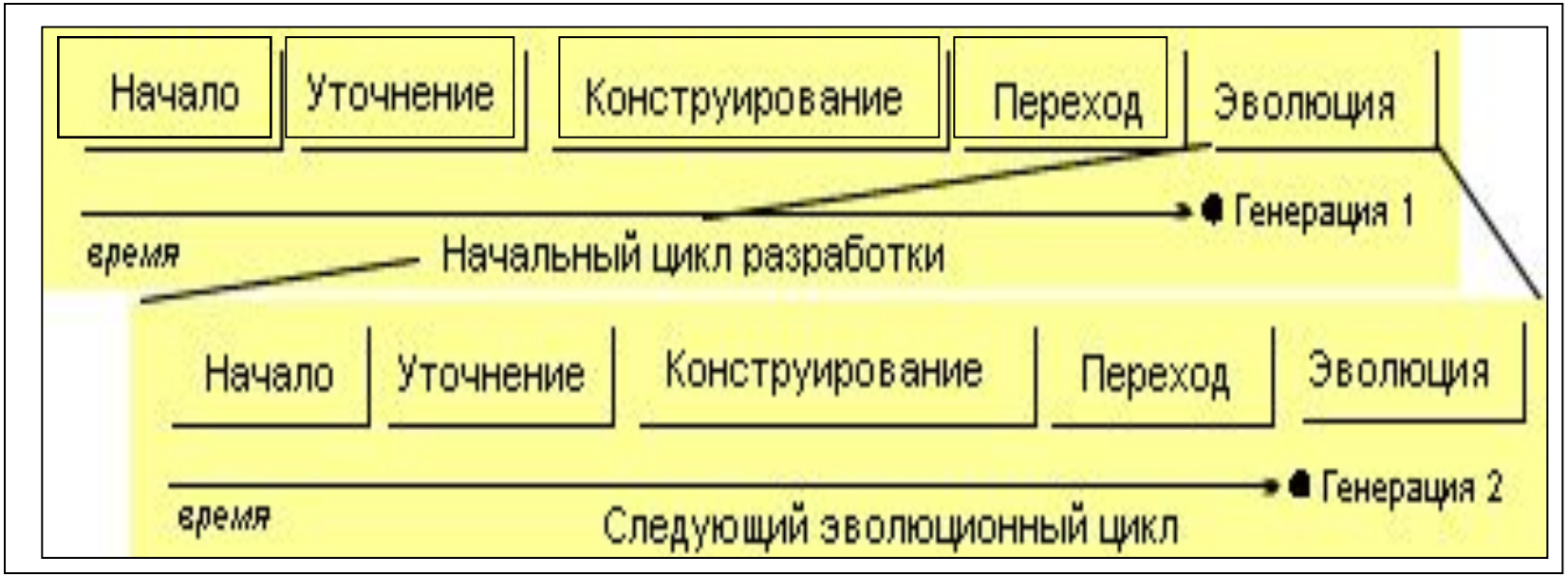
Каждый цикл развития состоит из четырех последовательных стадий.

Стадии завершаются главными вехами.

Первый цикл выполнения этих стадий - начальный цикл.

Последующие циклы развития - эволюционные циклы.

Каждый эволюционный цикл проходит те же четыре стадии.



Итерации

Каждая стадия может быть разбита на итерации.

Итерация - цикл, приводящий к выпуску изделия (внутренней или внешней версии) или подмножества конечного продукта, возрастающего от итерации к итерации до законченной системы.

Инструменты UML-моделирования - Бесплатные программы

Acceleo: (http://www.acceleo.org/pages/home/en) - основанная на Eclipse и EMF шаблонная система для генерации исходного кода из UML моделей.

ArgoUML: (http://argouml.tigris.org/ написано на языке Java)

Astade: (http://astade.tigris.org/) - платформо-независимое UML-средство на основе wxWidgets.

ATLAS Transformation Language: (http://www.eclipse.org/m2m/atl/) - QVT-инструмент, который способен трансформировать UML модели в другие модели. Доступно из Eclipse GMT project (Generative Modeling Tools).

BOUML: (http://bouml.free.fr/) - мультиплатформенное UML 2.0 средство, генерирует код C++/Java/IDL. Очень высокая производительность (написано на C++, на Qt). Лицензия GNU GPL.

Dia: GTK+/GNOME средство для построения диаграмм, которое также поддерживает UML (Лицензия GNU GPL)

Gaphor: (http://gaphor.sourceforge.net/) - GTK+/GNOME среда моделирования UML 2.0, написанная на Python

Kivio: (http://www.koffice.org/kivio/) - часть проекта Koffice

Инструменты UML-моделирования - Коммерческие системы

ARIS: (http://www.ids-scheer.com/en/ARIS/ARIS_Software/3730.html)

Borland Together: (http://www.borland.com/together/index.html)

Enterprise Architect: (http://www.sparxsystems.com.au)

IBM Rational Rose: (http://ibm.com/software/awdtools/developer/rose/)

MagicDraw: (http://magicdraw.com/)

Microsoft Visio: - редактор диаграмм для Windows

ModelMaker Tools: (www.modelmakertools.com/)

ObjectDomain: (http://objectdomain.com/welcome.do)

Poseidon: (http://www.gentleware.com/produsts/download.php4)

Sybase PowerDesigner: (http://www.sybase.ru/products/powerdesigner)

SmartDraw: (http://www.smartdraw.com/)

Telelogic Rhapsody: - среда разработки на основе визуального моделирования для разработчиков встраиваемых систем реального времени

UML Studio: (http://www.pragsoft.com/products.html)

СПАСИБО ЗА ВНИМАНИЕ!