Инструментальные средства разработки программ.

1.Инструменты разработки программных средств.

В процессе разработки программных средств в той или иной мере используется компьютерная поддержка процессов разработки ПС.

Это достигается путем представления хотя бы некоторых программных документов ПС (прежде всего, программ) на компьютерных носителях данных (например, дисках) и предоставлению в распоряжение разработчика ПС **специальных ПС** или включенных в состав компьютера **специальных устройств**, созданных для какой-либо обработки таких документов.

В качестве такого специального ПС можно указать **компилятор** с какого-либо языка программирования.

Компилятор избавляет разработчика ПС от необходимости писать программы на языке компьютера, который для разработчика.

ПС был бы крайне неудобен, - вместо этого он составляет программы на удобном ему языке программирования, которые соответствующий компилятор автоматически переводит на язык компьютера.

В качестве специального устройства, поддерживающего процесс разработки ПС, может служит **эмулятор** какого-либо языка.

Эмулятор позволяет выполнять (интерпретировать) программы на языке, отличном от языка компьютера, поддерживающего разработку ПС, например на языке компьютера, для которого эта программа предназначена.

ПС, предназначенное для поддержки разработки других ПС, будем **называть программным инструментом разработки ПС,** а устройство компьютера, специально предназначенное для поддержки разработки ПС,

будем называть **аппаратным инструментом разработки ПС.**

Инструменты разработки ПС могут использоваться в течении всего жизненного цикла
ПС для работы с разными программными документами.
Так текстовый редактор может использоваться для разработки практически любого
программного документа.
С точки зрения функций, которые инструменты выполняют при разработке ПС,
их можно разбить на следующие четыре группы: ·
редакторы,·
анализаторы,·
преобразователи,·
инструменты, поддерживающие процесс выполнения программ.

Редакторы поддерживают конструирование (формирование) тех или иных программных документов на различных этапах жизненного цикла.

Как уже упоминалось, для этого можно использовать один какой-нибудь универсальный текстовый редактор.

Однако, более сильную поддержку могут обеспечить специализированные редакторы: для каждого вида документов - свой редактор.

В частности, на ранних этапах разработки в документах могут широко использоваться графические средства описания (диаграммы, схемы и т.п.). В таких случаях весьма полезными могут быть графические редакторы.

На этапе программирования (кодирования) вместо текстового редактора может оказаться более удобным синтаксически управляемый редактор, ориентированный на используемый язык программирования.

Анализаторы производят либо статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных (например, проверку соответствия документов указанным стандартам), либо динамический анализ программ (например, с целью выявление распределения времени работы программы по программным модулям).

Преобразователи позволяют автоматически приводить документы к другой форме представления (например, форматеры) или переводить документ одного вида к документу другого вида (например, конверторы или компиляторы), синтезировать какой-либо документ из отдельных частей и т.п.

Инструменты, поддерживающие процесс выполнения программ, позволяют выполнять на компьютере описания процессов или отдельных их частей, представленных в виде, отличном от машинного кода, или машинный код с дополнительными возможностями его интерпретации.

Примером такого инструмента является эмулятор кода другого компьютера. К этой группе инструментов следует отнести и различные отладчики.

По-существу, каждая система программирования содержит программную подсистему периода выполнения, которая выполняет наиболее типичные для языка программирования программные фрагменты и обеспечивает стандартную реакцию на возникающие при выполнении программ исключительные ситуации (такую подсистему мы будем называть исполнительной поддержкой), - также можно рассматривать как инструмент данной группы.

2. Инструментальные среды разработки и сопровождения программных средств.

В настоящее время с каждой системой программирования связываются не отдельные инструменты (например, компилятор), а некоторая логически связанная совокупность программных и аппаратных инструментов поддерживающих разработку и сопровождение ПС на данном языке программирования или ориентированных на какую-либо конкретную предметную область.

Такую совокупность будем называть **инструментальной средой разработки и сопровождения ПС**.

Для таких инструментальных сред характерно,

во-первых, использование как программных, так и аппаратных инструментов, и,

во-вторых, определенная ориентация либо на конкретный язык программирования, либо на конкретную предметную область.

Инструментальная среда не обязательно должна функционировать на том компьютере, на котором должно будет применяться разрабатываемое с помощью ее ПС. Часто такое совмещение бывает достаточно удобным (если только мощность используемого компьютера позволяет это):не нужно иметь дело с компьютерами разных типов, в разрабатываемую ПС можно включать компоненты самой инструментальной среды.

Различают **три основных класса инструментальных сред разработки и сопровождения ПС**

среды программирования, ·

рабочие места компьютерной технологии,·

инструментальные системы технологии программирования.

Среда программирования предназначена в основном для поддержки процессов программирования (кодирования), тестирования и отладки ПС.

Рабочее место компьютерной технологии ориентировано на поддержку ранних этапов разработки ПС (спецификаций) и автоматической генерации программ по спецификациям.

Инструментальная система технологии программирования предназначена для поддержки всех процессов разработки и сопровождения в течение всего жизненного цикла ПС и ориентирована на коллективную разработку больших программных систем с длительным жизненным циклом.

Основные классы инструментальных сред разработки и сопровождения ПС.

3. Инструментальные среды программирования.

Инструментальные среды программирования содержат прежде всего текстовый редактор, позволяющий конструировать программы на заданном языке программирования, инструменты, позволяющие компилировать или интерпретировать программы на этом языке, а также тестировать и отлаживать полученные программы.

Кроме того, могут быть и другие инструменты, например, для статического или динамического анализа программ.

Взаимодействуют эти инструменты между собой через обычные файлы с помощью стандартных возможностей файловой системы.

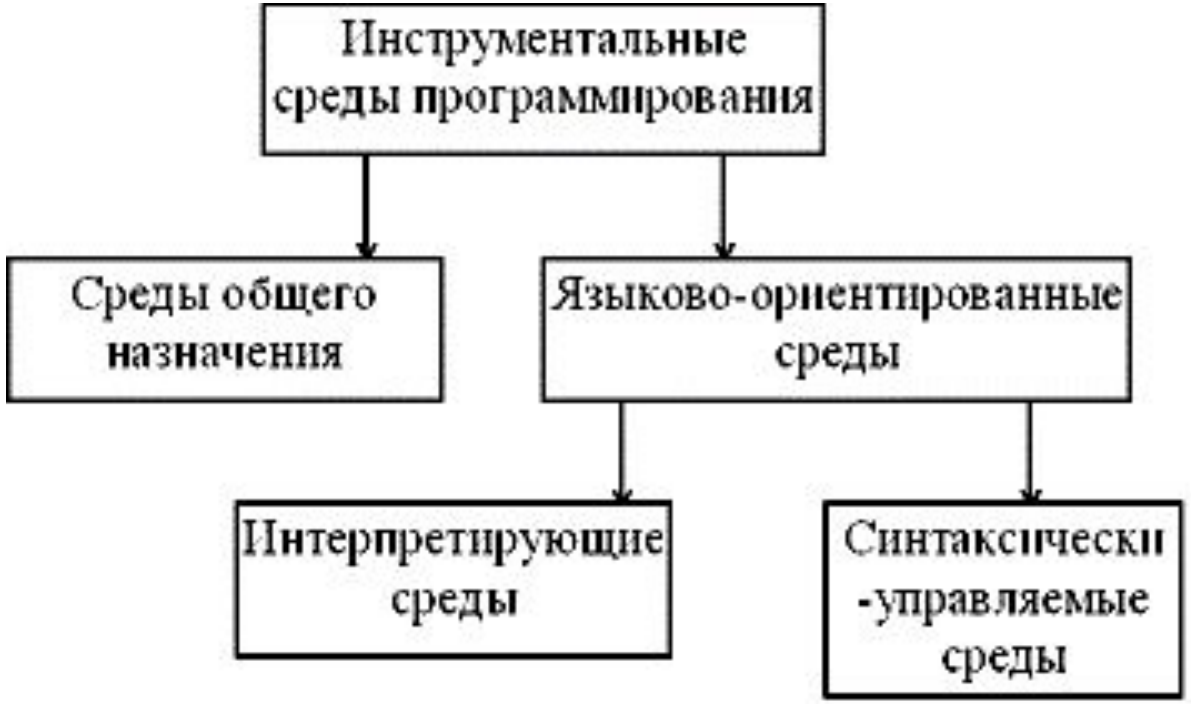
Различают следующие **классы инструментальных сред программирования**: ·

среды общего назначения,·

языково-ориентированные среды.

Инструментальные среды программирования общего назначения содержат набор программных инструментов, поддерживающих разработку программ на разных языках программирования (например, текстовый редактор, редактор связей или интерпретатор языка целевого компьютера) и обычно представляют собой некоторое расширение возможностей используемой операционной системы. Для программирования в такой среде на каком-либо языке программирования потребуются дополнительные инструменты, ориентированные на этот язык (например, компилятор).

.



. Классификация инструментальных сред программирования

4. Понятие компьютерной технологии разработки программных средств и ее рабочие места.

Имеются некоторые трудности в выработке строгого определения CASE-технологии (компьютерной технологии разработки ПС).

CASE - это абревиатура от английского Computer-Aided Software Engineering (Компьютерно-Помогаемая Инженерия Программирования). Но без помощи (поддержки) компьютера ПС уже давно не разрабатываются (используется хотя бы компилятор).

В действительности, в это понятие вкладывается более узкий (специальный) смысл, который постепенно размывается (как это всегда бывает, когда какое-либо понятие не имеет строгого определения).

Первоначально под CASE понималась инженерия ранних этапов разработки ПС (определение требований, разработка внешнего описания и архитектуры ПС) с использованием программной поддержки (программных инструментов).

Теперь под CASE может пониматься и инженерия всего жизненного цикла ПС (включая и его сопровождение), но только в том случае, когда программы частично или полностью генерируются по документам, полученным на указанных ранних этапах разработки. В этом случае CASE-технология стала принципиально отличаться от ручной (традиционной) технологии разработки ПС: изменилось не только содержание технологических процессов, но и сама их совокупность.

В настоящее время компьютерную технологию разработки ПС можно характеризовать

- Использованием программной поддержки для разработки графических требований и графических спецификаций ПС,
- автоматической генерации программ на каком-либо языке программирования или в машинном коде (частично или полностью),
- программной поддержки прототипирования.

Инструментальная система технологии программирования - это интегрированная совокупность программных и аппаратных инструментов, поддерживающая все процессы разработки и сопровождения больших ПС в течение всего его жизненного цикла в рамках определенной технологии.

Из этого определения вытекают следующие основные черты этого класса компьютерной поддержки: ·

комплексность, ·

ориентированность на коллективную разработку, ·

технологическая определенность, ·

интегрированность.

С учетом обсужденных свойств инструментальных систем технологии программирования можно выделить три их основные компоненты:·

база данных разработки (репозиторий),·

инструментарий, ·

интерфейсы.

Репозиторий - центральное компьютерное хранилище информации, связанной с проектом (разработкой) ПС в течении всего его жизненного цикла.

Инструментарий - набор инструментов, определяющий возможности, предоставляемые системой коллективу разработчиков. Обычно этот набор является открытым: помимо минимального набора (встроенные инструменты), он содержит средства своего расширения (импортированными инструментами), - и структурированным, состоящим из некоторой общей части всех инструментов (ядра) и структурных (иногда иерархически связанных) классов инструментов.

Интерфейсы разделяются на

1)пользовательский

2) системные.

Пользовательский интерфейс обеспечивает доступ разработчикам к инструментарию (командный язык и т.п.), реализуется оболочкой системы.

Системные интерфейсы обеспечивают взаимодействие между инструментами и их общими частями. Системные интерфейсы выделяются как архитектурные компоненты в связи с открытостью системы - их обязаны использовать новые (импортируемые) инструменты, включаемые в систему.

Различают **два класса инструментальных систем технологии программирования:**

1)инструментальные системы поддержки проекта и

2) языково-зависимые инструментальные системы.

Инструментальная система поддержки проекта - это открытая система, способная поддерживать разработку ПС на разных языках программирования после соответствующего ее расширения программными инструментами, ориентированными на выбранный язык. Такая система содержит ядро (обеспечивающее, в частности, доступ к репозиторию), набор инструментов, поддерживающих управление (management) разработкой ПС, независимые от языка программирования инструменты, поддерживающие разработку ПС (текстовые и графические редакторы, генераторы отчетов и т.п.), а также инструменты расширения системы.

Языково-зависимая инструментальная система - это система поддержки разработки ПС на каком-либо одном языке программирования, существенно использующая в организации своей работы специфику этого языка. Эта специфика может сказываться и на возможностях ядра (в том числе и на структуре репозитория), и на требованиях к оболочке и инструментам.

Унифицированный язык моделирования UML

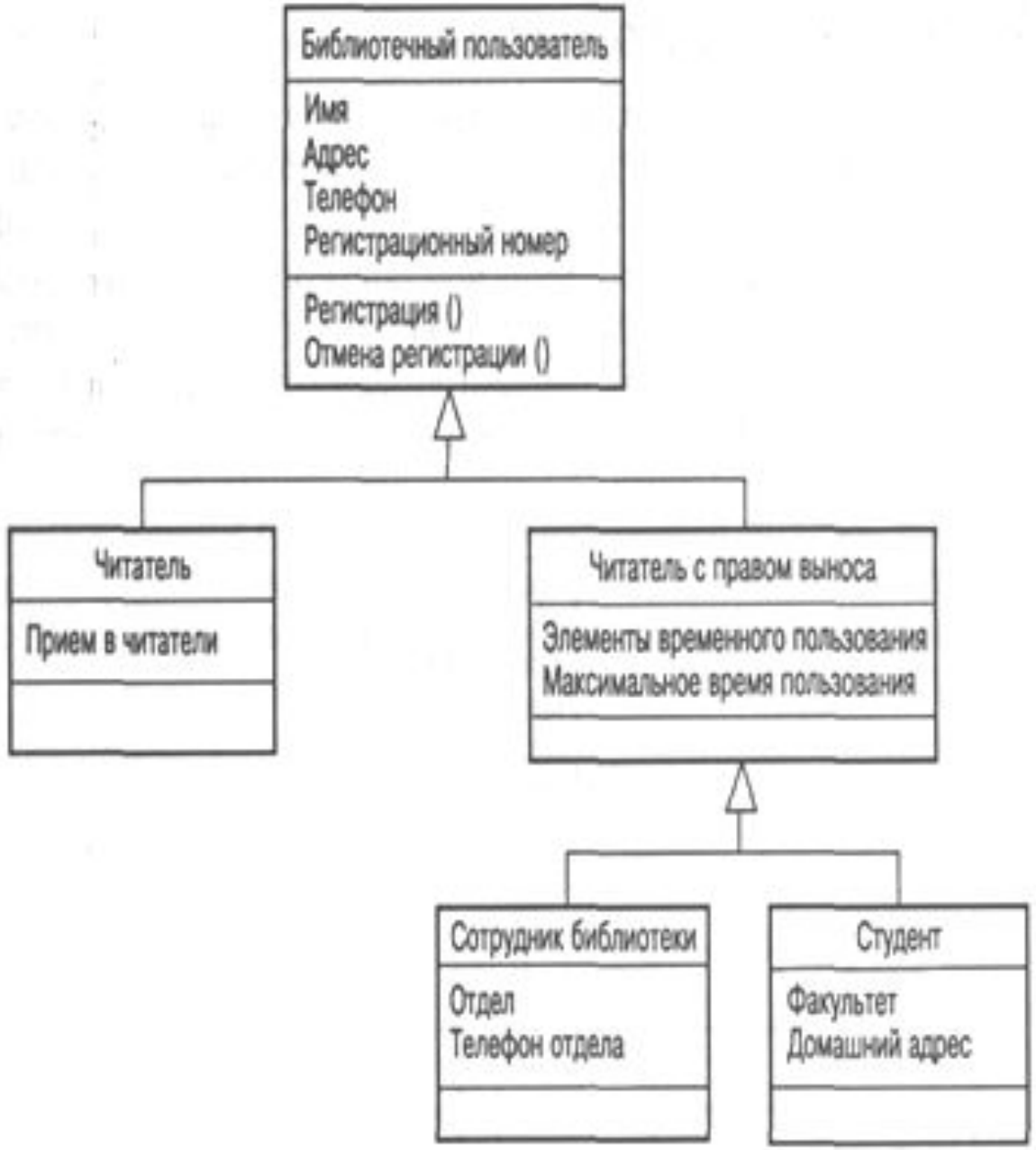
- Большинство существующих методов объектно-ориентированного анализа и проектирования (ООАП) включают как язык моделирования, так и описание процесса моделирования. **Язык моделирования** – это нотация (в основном графическая), которая используется методом для описания проектов.
- **Нотация** представляет собой совокупность графических объектов, которые используются в моделях; она является синтаксисом языка моделирования. Например, нотация диаграммы классов определяет, каким образом представляются такие элементы и понятия, как класс, ассоциация и множественность.
- **Процесс** – это описание шагов, которые необходимо выполнить при разработке проекта.
- **Унифицированный язык моделирования UML** (Unified Modeling Language) – это преемник того поколения методов ООАП, которые появились в конце 80-х и начале 90-х гг.

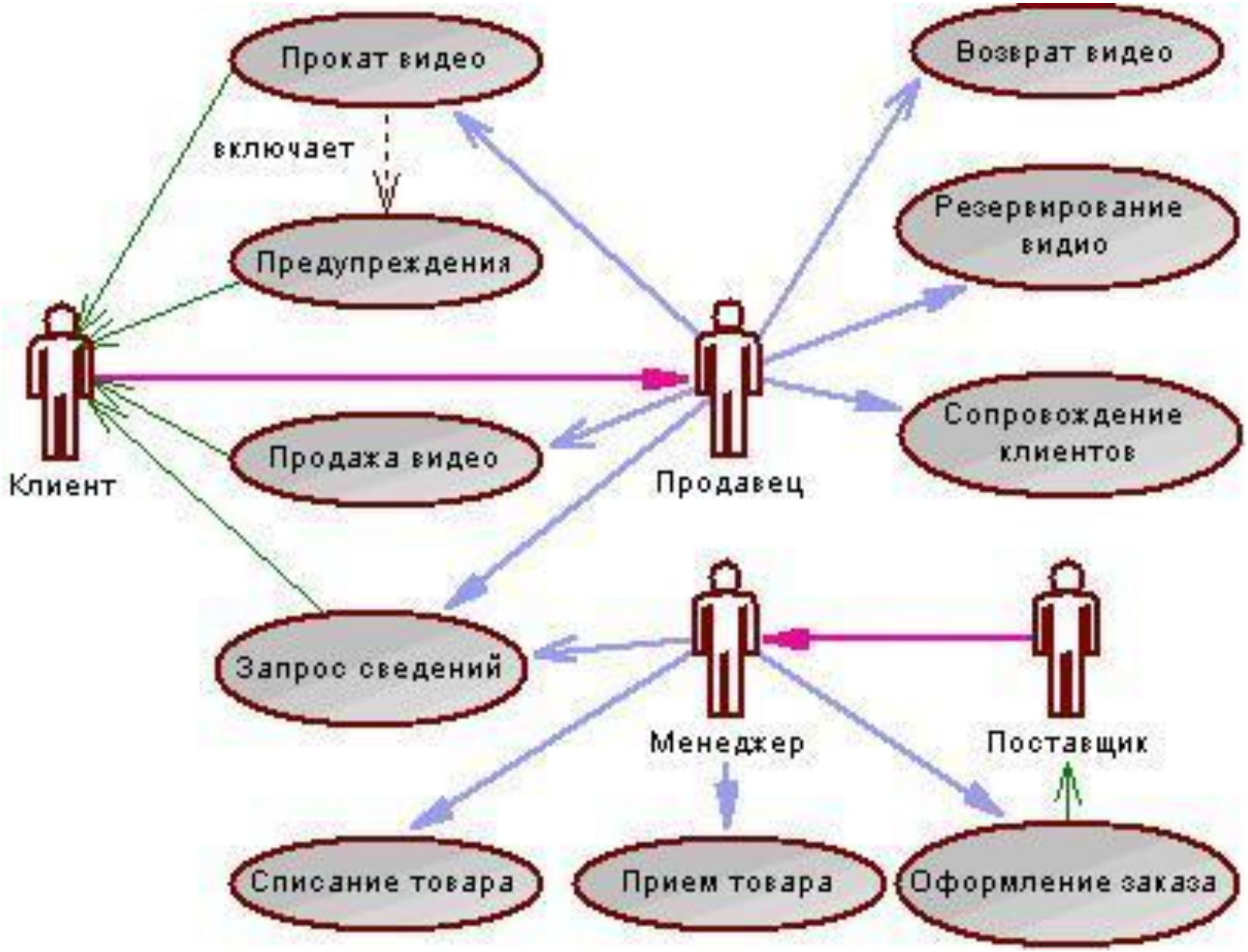
- **Язык UML** представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.
- Конструктивное использование языка UML основывается на понимании общих принципов моделирования сложных систем и особенностей процесса объектно-ориентированного проектирования (ООП) в частности. Выбор выразительных средств для построения моделей сложных систем предопределяет те задачи, которые могут быть решены с использованием данных моделей. При этом одним из основных принципов построения моделей сложных систем является принцип абстрагирования, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого предназначения. При этом все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели.

- UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов.
- **Диаграмма в UML** – это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения.
- **Диаграмма** – в некотором смысле одна из проекций системы. Как правило, за исключением наиболее тривиальных случаев, диаграммы дают свернутое представление элементов, из которых составлена система. Один и тот же элемент может присутствовать во всех диаграммах, или только в нескольких (самый распространенный вариант), или не присутствовать ни в одной (очень редко).
- Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций, соответствующих пяти наиболее употребительным видам, которые составляют архитектуру программной системы.

- UML выделяют следующие типы диаграмм:
- – **диаграммы вариантов использования** (usecase diagrams) – для моделирования бизнес-процессов организации (требований к системе);
- – **диаграммы классов** (class diagrams) – для моделирования статической структуры классов системы и связей между ними. На таких диаграммах показывают классы, интерфейсы, объекты и кооперации, а также их отношения. При моделировании объектно-ориентированных систем этот тип диаграмм используют чаще всего. Диаграммы классов соответствуют статическому виду системы с точки зрения проектирования;
- – **диаграммы поведения системы** (behavior diagrams);
- **диаграммы взаимодействия** (interaction diagrams) – для моделирования процесса обмена сообщениями между объектами.
- – **диаграммы состояний** (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в

- – **диаграммы деятельностей** (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования или моделирования деятельностей.
- – **диаграммы реализации** (implementation diagrams): *диаграммы компонентов* (component diagrams) – для моделирования иерархии компонентов (подсистем) системы; *диаграммы размещения* (deployment diagrams) – для моделирования физической архитектуры системы.


Библиотечный пользователь
Имя
Адрес
Телефон
Регистрационный номер
Регистрация ()
Отмена регистрации ()
Читатель
Прием в читатели
Читатель с правом выноса
Элементы временного пользования
Максимальное время пользования
Сотрудник библиотеки
Отдел
Телефон отдела
Студент
Факультет
Домашний адрес


Прокат видео
включает
Предупреждения
Возврат видео
Резервирование видио
Сопровождение клиентов
Продажа видео
Клиент
Продавец
Запрос сведений
Менеджер
Поставщик
Списание товара
Прием товара
Оформление заказа