Анализ и проектирование на UML

Направления подготовки
«Бизнес-информатика», «Прикладная информатика»

Максим Валерьевич Хлопотов,
старший преподаватель кафедры ИС

Паттерны проектирования

Паттерны проектирования
Образцы проектирования (design patterns) – это сравнительно новая форма обмена программистским опытом и удачными проектными решениями.
Неформально говоря, образец проектирования — это типичное решение типичной проблемы в данном контексте.

Паттерны проектирования

Обычно описание образца состоит из четырех основных элементов.

• Имя.
Ссылаясь на имя образца, мы можем кратко описать проблему проектирования, ее решения и их последствия. Это позволяет проектировать на более высоком уровне абстракции. Словарь общеизвестных имен образцов позволяет эффективно вести обсуждение с коллегами, лаконично документировать принимаемые архитектурные решения. Подбор хорошего имени — одна из важнейших задач при составлении описания образца.

Паттерны проектирования

• Задача.
Описание контекста применения образца проектирования, т. е. описание конкретной проблемы проектирования и перечня условий, при выполнении которых имеет смысл применять данный образец.

• Решение.
Описание элементов проектирования, отношений между ними, функции каждого элемента. Дается абстрактное описание задачи проектирования и ее обобщенное решение.

Паттерны проектирования

- Результаты.

Здесь описываются следствия применения образца: влияние на степень эффективности, гибкости, расширяемости и переносимости системы.

Паттерны проектирования

Разумеется, образами проектирования пользовались и пользуются все разумные архитекторы испокон веков, и не только в области программирования, а буквально во всех областях человеческой деятельности. До недавнего времени считалось, что хорошим архитектором, способным быстро, эффективно и надежно спроектировать сложную программную систему, может быть только достаточно опытный человек. Появление UML радикальным образом изменило ситуацию.

Паттерны проектирования

Появление UML имеет для программирование примерно такое значение, как изобретение нотной записи для музыки или введение буквенных обозначений для математики.

Используя UML, архитектор программной системы может сообщить свои идеи (именно идеи, а не примеры готовых решений на языке программирования) в лаконичной и понятной форме, доступной для восприятия подавляющему большинству разработчиков.

Паттерны проектирования

Синтаксически в UML образец проектирования — это параметрическая кооперация классов (т. е. шаблон кооперации).
Классы, входящие в кооперацию, можно рассматривать как параметры, а всю кооперацию в целом, как шаблон взаимодействия.

Паттерны проектирования

Таким образом, чтобы применить некоторый образец проектирования в определенном контексте, достаточно связать параметры шаблона с конкретными значениями, т. е. указать, какие конкретные классы модели играют роли классификаторов, участвующих в данной кооперации (образце).

Для этого в UML предусмотрен специальный синтаксис: применяемый образец изображается в виде пунктирного овала, внутри которого написано имя кооперации. Этот овал соединяется пунктирными линиями с классами, которые являются фактическими аргументами, причем на линии указывается имя роли, которую класс играет в применяемой кооперации.

Паттерны проектирования

Пример

Классический образец проектирования Observer (он же Publish-Subscribe). (Паттерн сильно упрощён и сокращён)

• Задача.

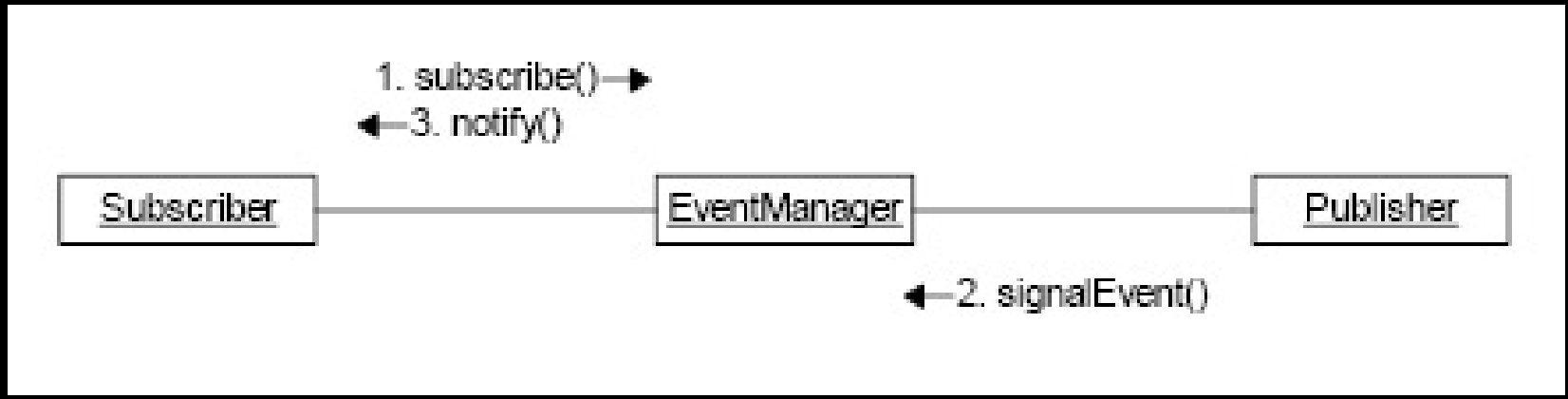
Поведение некоторых объектов системы (подписчиков — экземпляров класса Subscriber) должно зависеть от изменения состояния (события) другого объекта (издателя — экземпляра класса Publisher). Однако издатель не должен прямо взаимодействовать с подписчиками.

Паттерны проектирования

• Решение.

Ввести службу уведомления о событиях, с тем чтобы издатель мог опосредованно уведомлять подписчиков о наступлении события. Для этого вводится (единственный) объект класса EventManager, реализующий данную службу. Класс EventManager имеет метод subscribe, вызывая который подписчик подписывается на уведомлении о наступлении события, и метод signalEvent, посредством которого издатель уведомляет о наступлении события. При вызове метода signalEvent объект EventManager посылает уведомления о событии всем подписчикам, вызывая метод notify, переданный в качестве параметра при подписке.

Паттерны проектирования



Паттерны проектирования

• Результат.

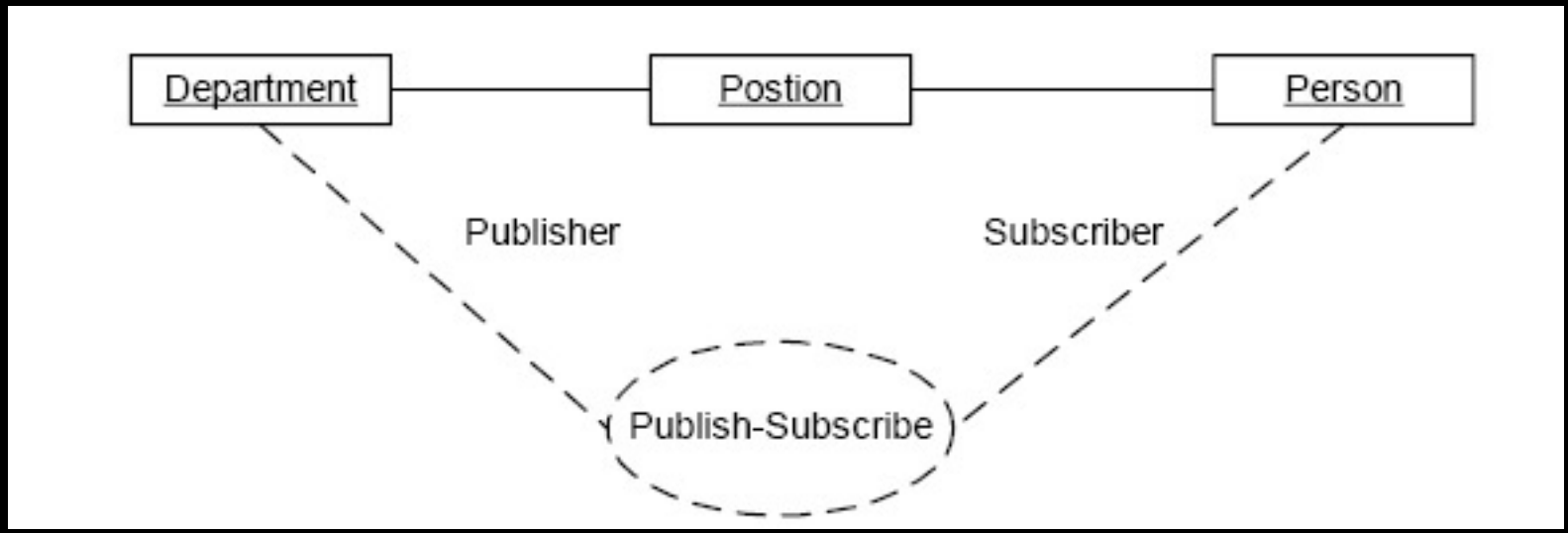
Класс Publisher не зависит от класса Subscriber. Возможны различные модификации образца: при необходимости получать уведомления о различных событиях, нужно добавить соответствующий параметр (например, имя события); для увеличения эффективности можно обязанности ведения службы уведомления о событиях перепоручить прямо классу Publisher, но в этом случае снижается гибкость, поскольку реализовывать данную службу придется в каждом классе-издателе.

Паттерны проектирования

Пример

Пусть в системе требуется уведомлять сотрудников об изменении состояния подразделения — вполне естественное требование, поскольку поведение сотрудников существенно зависит от состояния подразделения, и мы не хотим нагружать руководителя подразделения обязанностью персонально извещать каждого подчиненного — у руководителя и без того хлопот полон рот. В этом случае мы можем на диаграмме кооперации показать, что к классам Department и Person следует применить образец проектирования Publish-Subscribe, причем класс Department играет роль Publisher, а класс Person играет роль Subscriber.

Паттерны проектирования



Паттерны проектирования

Простая нотация на рисунке несет значительную семантическую нагрузку. Приведенная диаграмма определяет кооперацию, причем в этой кооперации определено гораздо больше элементов, чем нарисовано на диаграмме. В частности, подразумевается, что в модели определен класс EventManager (хотя он и не присутствует на диаграмме), между классами определены соответствующие ассоциации для вызова методов, сами классы обладают нужными методами для применения образца и т. д.

Паттерны проектирования

В большинстве практических инструментов моделирование применение образцов носит менее интуитивный и более формальный характер. Весьма вероятно, что в используемом инструменте даже не найдется такой фигуры, как пунктирный овал, предназначенный для обозначения применения образца. Но с той же степенью вероятности найдется библиотека готовых образцов.
Как правило, инструменты позволяют включить образец в модель (в форме заготовки диаграммы кооперации) и вручную связать (отождествить или переопределить) элементы образца с элементами модели. Другими словами, общая схема применения образца остается той же самой, но задача понимания образца остается за программистом.

Паттерны проектирования

В UML определено еще одно понятие, тесно связанное с образцами проектирования. Это каркас — совокупность логически связанных образцов. Синтаксически каркас в UML — это пакет со стереотипом «framework». Такой стереотип означает, что в пакете собраны образцы проектирования согласованные таким образом, чтобы их можно было применять совместно и систематически.

Паттерны проектирования

Не все образцы проектирования, описанные в литературе, могут быть адекватно выражены в UML. Наилучшим образом язык подходит для описания таких образцов, в которых решение выражается в форме описания поведения взаимодействующих объектов.

Паттерны проектирования

В настоящее время опубликовано несколько десятков общепризнанных образцов и несколько сот используемых менее часто. Квалифицированный архитектор обязан знать (но не обязан каждый раз использовать) общепризнанные образцы, хотя бы по названиям. Если используемый инструмент моделирования содержит библиотеку образцов, то архитектору следует изучить данную библиотеку во всех деталях. Это настолько же полезно, насколько программисту полезно знать стандартные библиотеки используемого им языка программирования.

Паттерны проектирования

UML позволяет наглядно описывать образцы проектирования.
Применение образцов проектирования повышает качество проектирования архитектуры, что, по общему мнению, ускоряет реализацию.
Разумеется, это не единственный фактор положительного влияния UML на процесс разработки программного обеспечения.

Управление моделями

В UML определены несколько понятий, которые имеют особое для значение при определении как структуры модели, так и структуры моделируемого приложения:
модели,
системы и подсистемы,
а также синтаксические средства их выражения в UML.

Управление моделями

При моделировании наше внимание сосредоточено на описании некоторой части реального мира. Например, при моделировании приложения рассматривается его программная реализация, аппаратура, на которой исполняются программы, и пользователи, взаимодействующие с приложением.

Все это вместе называется *физической системой*.

Управление моделями

Одна и та же физическая система может быть описана с различных точек зрения. Описание физической системы называется моделью. Каждая модель описывает всю физическую систему в целом, но модели могут сильно отличаться степенью детальности, используемыми средствами и расстановкой акцентов. Это определяется целью, с которой составляется модель. Например, мы можем различать концептуальную модель, логическую модель и модель реализации для одной и той же физической системы.

Управление моделями

Если физическая система сложна и велика (а именно так обычно и бывает), то ее целесообразно мысленно разбить на части, называемые подсистемами и рассматривать отдельно и детально каждую подсистему.

Управление моделями

Например, информационная система отдела кадров конкретной организации, как физическая система, состоит из некоторого количества компьютеров, программных компонентов, составляющих приложение и развернутых на этих компьютерах, а также служащих отдела кадров, которым вменено в обязанность использовать приложение в повседневной работе.

С точки зрения выполняемых функций в этой физической системе можно выделить две подсистемы: подсистему работы с персоналом и подсистему работы со штатным расписанием. Можно разделить все описание на несколько моделей, чтобы описать систему в целом с различных точек зрения, а можно разделить систему на части и описать их по отдельности как подсистемы. Более того, можно комбинировать эти структуры произвольным образом: разбить модель на несколько подсистем, каждую из которых описать с помощью нескольких моделей и т. д.

Управление моделями

Когда модель достаточно велика, реально возникает проблема управления моделями.
Если модель достаточно велика, ее нужно разделить на части обозримого размера и рассматривать их по отдельности. Для этой цели в UML используется одно универсальное средство — пакет.

Пакет — это группирующая сущность в UML.
На диаграммах пакет изображается в виде фигуры — прямоугольник с закладкой.
Если внутри пакета ничего не изображено, то имя пакета пишется в основном прямоугольнике, в противном случае — в закладке.

Свойства пакета в UML

- Отношение владения.

Говорят, что пакет владеет объявленными в нем элементами модели, а элементы принадлежат владеющему ими пакету. Пакет может владеть любыми элементами модели, в частности, пакетами. Отношение владения является строгой композицией, т. е. каждый элемент модели принадлежит ровно одному пакету. Всегда имеется корневой пакет (с именем по умолчанию). Таким образом, структура пакетов по отношению владения (или вложенности) образует в модели строгую иерархию, подобную иерархии папок и файлов в файловой системе.

Свойства пакета в UML

- Пространство имен.

Каждый пакет в модели образует пространство имен, т. е. область определения и использования имен.

Пространства имен вложены друг в друга. Пространства имен пакетов вложены в соответствии с иерархией отношения владения, пространства имен прочих элементов вложены в соответствии со специфической иерархией элементов. В частности, каждое составное состояние машины состояний образует вложенное пространство имен. Таким образом, пространства имен также образуют строгую иерархию.

Управление моделями

- Отношение владения

Внутри основного прямоугольника фигуры пакета можно помещать любые элементы модели — тем самым моделируется отношение владения: пакет владеет элементом, помещенным внутрь его фигуры.

Управление моделями

Модели нуждаются в структуре, в противном случае они не отвечают своему основному назначению — служить средством визуализации, спецификации, проектирования и документирования.

Управление моделями

Если диаграмма в модели не помещается на экран так, чтобы тексты и значки оставались читаемыми, от модели мало проку. Если имена элементов модели состоят из десятков непонятных слов, модель трудно использовать. Если для понимания какой-то характеристики приложения нужно одновременно рассматривать десяток диаграмм, модель не отвечает своему назначению. Если для поиска ответа на конкретный вопрос нужно просмотреть всю модель подряд, она никуда не годится.

Управление моделями

Модель имеет хорошую, удобную для работы структуру, если выполняются следующие три количественных критерия.

- Количество сущностей, изображенных на всех диаграммах, примерно одинаково и равно 7±3. Если сущностей на диаграмме три или меньше, то диаграмма недостаточно информативна, чтобы включать ее в модель отдельно. Такую диаграмму либо не стоит включать в модель, либо можно объединить с другой однородной диаграммой. Если на диаграмме больше десяти сущностей, то она, как правило, слишком трудна для понимания с одного взгляда. Такую диаграмму целесообразно разбить на несколько диаграмм.

Управление моделями

- Ширина ветвления в дереве вложенности пакетов с учетом диаграмм примерно постоянна и равна 7±3. Другими словами, пакету должны принадлежать от трех до десяти пакетов или диаграмм. Если их меньше, то не понятно, нужен ли такой малосодержательный пакет как отдельная сущность. Если больше, то в деталях пакета можно "утонуть".

Управление моделями

- Число использующих вхождений элементов модели должно быть ограничено сверху значением 7±3, при этом больше половины элементов модели должно присутствовать на диаграммах.

Любой данный элемент модели может присутствовать на некотором количестве диаграмм: 0, 1 или больше. Если на диаграммах нарисовано меньше половины существующих в модели элементов, то это не модель, а "вещь в себе", о назначении которой трудно догадаться.

Если один и тот же элемент повторяется в разных местах больше десяти раз, то это "масло масляное".

Управление моделями

Приведенные критерии характеризуют форму модели, но никак не затрагивают ее содержание.
Способы структурирования модели по содержанию:
- По структуре приложения.
В основу структуры пакетов кладется одна из реальных структур приложения, например, компонентная структура.
Вся система разбивается на пакеты, соответствующие компонентам (или подсистемам), они, если нужно, разбиваются на более мелкие части и т. д.
- По фазам процесса разработки.
Модель делится на пакеты в соответствии с фазами ее создания (набор которых зависит от принятой дисциплины моделирования). Например, пакет для диаграмм концептуального уровня, пакет диаграмм детального проектирования, пакет диаграмм реализации и развертывания и т. д.
- По представлениям.
Модель делится на пакеты по представлениям. Например, представление использования, логическое представление, представление реализации и т. д.

Управление моделями

Авторы языка рекомендуют структурировать модель по представлениям. Вслед за ними разработчики большинства инструментов пытаются навязать пользователю структуру модели.
Только в сравнительно простых моделях можно обойтись каким-то одним принципом структуризации.
В более сложных моделях приходится определять достаточно много пакетов (особенно если придерживаться наших критериев локального ограничения сложности).
Поэтому исходить нужно из потребностей модели, руководствуясь опытом и здравым смыслом — указанные принципы структуризации полезны, но все равно требуют размышлений при применении.

Управление моделями

Допустим, что выбрана принципиальная структура пакетов в модели.
Пакеты могут находиться в следующих отношениях:

- отношения владения (вложенности);
- индуцированные отношения;
- стереотипные зависимости;
- обобщение.

Управление моделями

Вложенность пакетов влечет вложенность пространств имен. Всякий элемент модели, определенный в данном пространстве имен, видит все элементы, определенные в этом же пространстве имен и во всех объемлющих пространствах имен. Имена в "параллельных" и вложенных пространствах имен не видимы для данного элемента. Если в цепочке вложенных пространств имен определены несколько разных элементов с одним именем, то для данного элемента видимым является ближайший при просмотре изнутри наружу.

Управление моделями

Индуцированное отношение — это отношение между пакетами, которое просто отражает наличие такого же отношения между некоторыми элементами данных пакетов.

Например, если указано, что пакет X зависит от пакета Y, то это означает, что один или несколько элементов модели, которыми владеет пакет X, зависят от одного или нескольких элементов модели, которыми владеет пакет Y. При этом вовсе не подразумевается, что все элементы пакетов находятся в данном отношении — достаточно наличия одной пары.

Управление моделями

Существуют два специальных стандартных стереотипа отношения зависимости — «access» и «import», которые имеют сходное назначение, но различаются в некоторых деталях.

В обоих случаях это отношение зависимости между пакетами, которое указывает на то, что зависимый пакет имеет доступ к открытым элементам независимого пакета (т. е. зависимый пакет "видит" открытое содержимое независимого пакета).

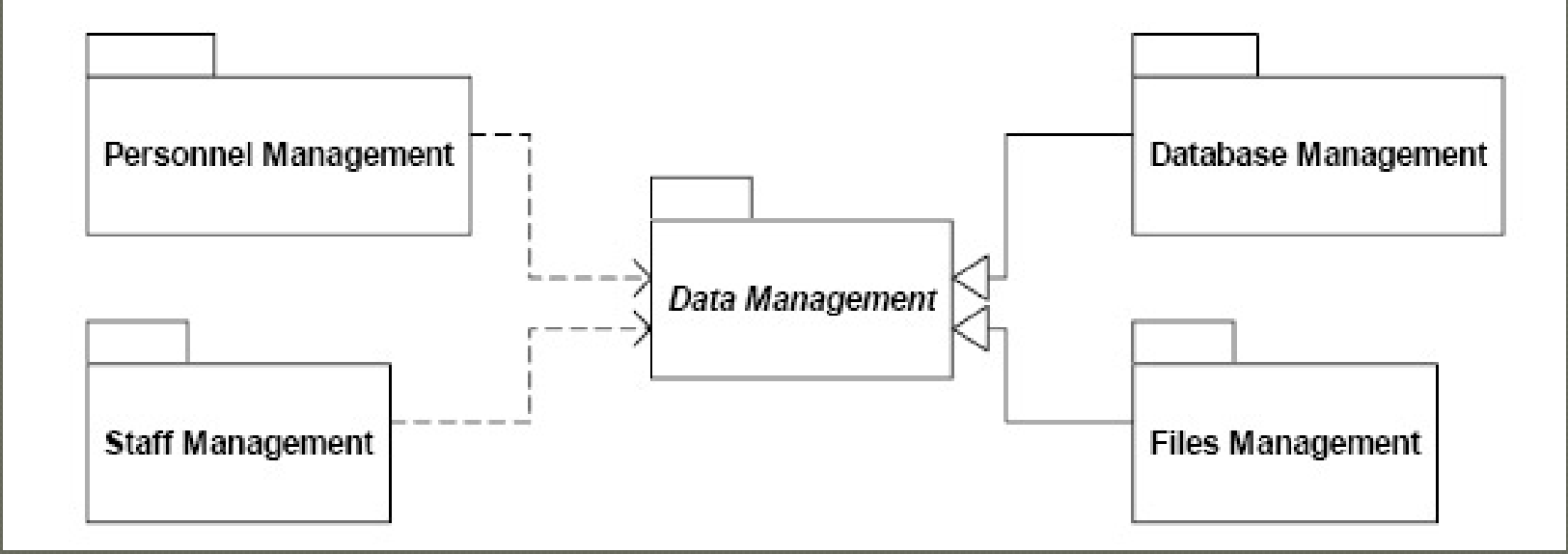
Различие между ними в том, что использование зависимости «access» не влияет на пространство имен зависимого пакета — для доступа к элементу независимого пакета нужно указывать составное имя, а при использовании зависимости «import» происходит расширение пространства имен зависимого пакета пространством имен независимого пакета.

Управление моделями

Хотя пакеты не являются классификаторами, тем не менее для них можно указать отношение обобщения.
Обычно отношение обобщения для пакетов применяется в следующей ситуации.
Допустим, что имеется несколько однородных в некотором смысле пакетов, например, несколько альтернативных вариантов реализации одной и той же функциональности. В таком случае можно определить в модели абстрактный пакет, который обобщает конкретные варианты.

Управление моделями

Например, на рисунке. представлена одна из возможных структур пакетов информационной системы отдела кадров, касающаяся управления данными.
Пакеты Personnel Management и Staff Management зависят от абстрактного пакета Data Management, который является обобщением пакетов Database Management и Files Management.



Управление моделями

Спецификация UML предусматривает довольно большое число стандартных стереотипов для пакетов,
«facade»
Пакет, который содержит только ссылки на элементы, определенные в других пакетах. Используется для описания "внешнего вида" других пакетов.
«framework»
Пакет, содержащий образцы и шаблоны. Используется для описания повторно используемых архитектурных решений.
«metamodel»
Модель, которая описывает другую модель. Например, метамодель UML.
«modelLibrary»
Пакет, содержащий определения элементов моделирования, предназначенных для использования в других пакетах.
«profile»
Пакет, содержащий определения элементов моделирования, предназначенных для моделирования в определенной предметной области.
«systemModel»
Модель, содержащая несколько моделей одной физической системы.
«topLevel»
Пакет, который является конем иерархии вложенности пакетов.

Управление моделями

UML позволяет придать описанию модели любую структуру — моделирующий субъект должен приложить определенные мыслительные усилия, чтобы выбрать хорошую структуру, подходящую для данного случая.

Управление моделями

UML позволяет придать описанию модели любую структуру — моделирующий субъект должен приложить определенные мыслительные усилия, чтобы выбрать хорошую структуру, подходящую для данного случая.

Применение UML

Технологию программирования целесообразно рассматривать в трех аспектах.

• Модель процесса, т. е. порядок проведения типового проекта по разработке программного обеспечения. Сюда относятся понятия жизненного цикла программного обеспечения, определение модели процесса — выделение в нем фаз, вех, потоков работ и других составляющих. Характерным для данного аспекта является рассмотрение на уровне программирующей организации в целом.

Применение UML

• Модель команды, т. е. отношения между людьми в процессе разработки. Сюда относятся определение обязанностей работников — участников процесса, регламенты их взаимодействия, рабочие процедуры и т. п. Характерным для данного аспекта является рассмотрение на уровне группы (команды) или проекта.

Применение UML

• Дисциплина программирования, т. е. методы создания конкретных артефактов, входящих в состав программного обеспечения. Сюда относятся описание и применение образцов проектирования, стандарты кодирования, методы тестирования и отладки и т. д. Характерным для данного аспекта является рассмотрение на уровне отдельного работника.

Применение UML

Задачей технологии программирования является исследование и улучшение процессов разработки программного обеспечения. При этом можно преследовать различные цели, т. е. улучшать процесс в различных направлениях.

Применение UML

Приведем несколько примеров задач, исследуемых и решаемых технологией программирования.

- Повышение надежности программного обеспечения.
- Снижение совокупной стоимости владения программным обеспечением.
- Повышение продуктивности программирования.

Применение UML

Применение UML оказывает положительное влияние на продуктивность процесса разработки программного обеспечения. Причина влияния заключается в том, что применение UML унифицирует представления информации в циклах повышения продуктивности. Унификация обеспечивает ускорение прохождения циклов, повышение сохранности, облегчение восприятия и повышение надежности принимаемых решений.

Следует подчеркнуть, что названные причины имеют место, только если UML действительно применяется по существу.

Применение UML

Применение элементов UML
Первый вопрос, который надлежит поставить перед собой, приступая к применению UML: чего мы хотим достичь?

Концептуальное моделирование.
Спецификация требований.
Детальное проектирование.

Концептуальное моделирование

В этом варианте целью моделирования является достижение понимания моделируемой физической системы (приложения, бизнес-процесса…), ее назначения и области применения.
Уровни понимания:
• первый уровень — когда появляется приятное чувство, что все понял;
• второй уровень — когда можешь повторить сказанное;
• третий уровень — когда можешь найти ошибку.
Для понимания системы на первом уровне моделирование на UML не обязательно — достаточно устного объяснения и, может быть, нескольких наглядных слайдов с картинками. Второй уровень легко достигается с помощью связного текста на естественном языке. Применение UML оправдано, когда нужно достичь третьего уровня — не просто познакомиться с идеей системы, но увидеть ее в целом и, может быть, найти пробелы или ошибки.

Применение UML

Для концептуального моделирования полные диаграммы использования обязательны, а также желательны и полезны диаграммы классов (с минимумом подробностей) в объеме словаря предметной области, диаграмма компонентов или размещения для перечисления основных артефактов и, может быть, диаграммы взаимодействия или деятельности для типовых сценариев основных вариантов использования. В некоторых случаях (не во всех, а только в тех, когда это существенно для понимания логики работы приложения) оказываются полезны диаграммы состояний. Очень важно не забыть упомянуть в примечаниях те ключевые аспекты, которые не отражены в графической нотации.

Главные требования к концептуальной модели — обозримость и согласованность.

Диаграмм не должно быть много и они не должны быть сложными, но расхождения в обозначениях и названиях крайне нежелательны.

Спецификация требований

Основной целью моделирования является получение артефакта, который мог бы послужить техническим заданием на разработку.
Техническое задание должно определять требования к системе и обеспечивать эффективную проверку того, что требования действительно выполнены. Также как и в случае концептуального проектирования, основой спецификации требований являются диаграммы использования, но их детальность и полнота должны быть существенно выше.

Спецификация требований

Реализация вариантов использования в форме диаграмм взаимодействия обязательна — это основа для построения набора тестов приемо-сдаточных испытаний.

На диаграммы использования возлагается значительно большая ответственность, поэтому разрабатывать их необходимо намного тщательнее. Например, если на диаграмме не изображен некоторый вариант использования, то это означает, что система не должна иметь соответствующую функциональность.

Спецификация требований

В спецификации требований диаграмма компонентов не просто желательна, а обязательна — техническое задание должно определять, хотя бы на уровне названий, что именно должно быть разработано. Между концептуальной моделью и спецификацией требований нет непроходимой границы — довольно часто концептуальная модель за несколько итераций перерастает в спецификацию требований.

Удовлетворительная спецификация требований оказывается в 3–5 раз объемнее концептуальной модели.

Применение UML

"Единственно правильной" модели нет и не может быть. Если программирование — это искусство, то что говорить об архитектурном проектировании и концептуальном моделировании? Это, видимо, элитарное искусство.

Не переоценивайте возможности инструмента.
Не пренебрегайте возможностями инструмента.
Не переоценивайте собственные возможности
Не пренебрегайте собственными возможностями

Л.р. 4

- Титульный лист
- Описание объекта автоматизации
- Программная и системная архитектура

- Не менее 10 диаграмм
- Обязательные диаграммы: представление использования, диаграмма классов (с операциями), диаграммы взаимодействия, контекстная диаграмма
- При необходимости: диаграмма пакетов, диаграмма состояний, диаграммы реализации
- Дополнительно: использование паттернов проектирования, состояния на диаграммах взаимодействия

Выводы

- 7 ± 3 сущности на одной диаграмме.
- Диаграмма должна охватываться «одним взглядом».
- Управление моделями – для того, кто моделирует, а не для компьютера.
- Образцы проектирования полезно знать.
- Нет наилучшего процесса для всех типов проектов и всех типов организаций, но для каждого типа проектов и для каждого типа организаций в отдельности — есть.