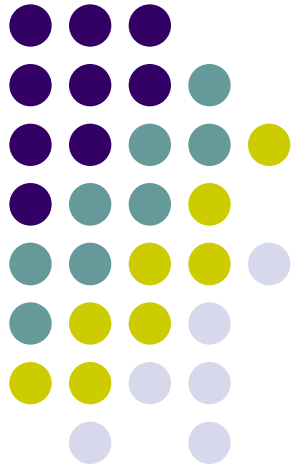
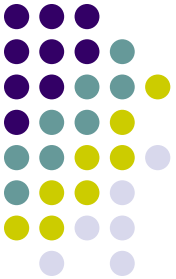
Лекция 4. Метамодель языка UML

Вопросы:

1. Пакеты UML
2. Специфика описания метамодели языка UML
3. Диаграммы UML и их виды
4. Rational Unified Process и его содержание



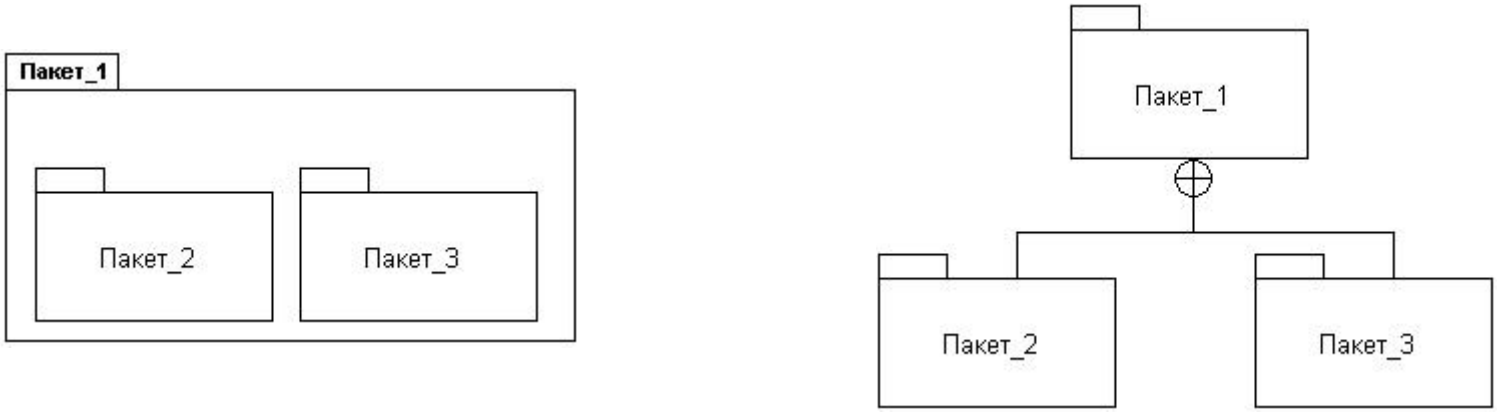
Пакеты в языке UML



- Пакет — основной способ организации элементов модели в языке UML.
- Каждый пакет владеет всеми своими элементами, т. е. теми элементами, которые в него включены
- Каждый элемент может принадлежать только одному пакету
- Пакеты могут быть вложены в другие пакеты, создавая иерархию

Графическая нотация пакетов

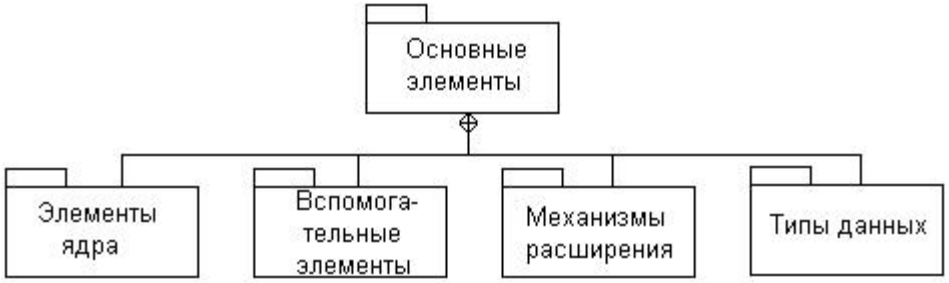
Вложенность пакетов



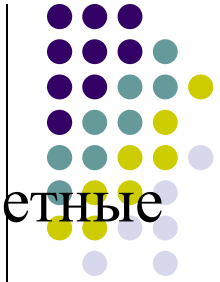
Основные пакеты метамодели языка UML

- Основой представления UML на метамодельном уровне является описание трех его логических блоков или пакетов: Основные элементы, Элементы поведения и Общие механизмы

Эти пакеты в свою очередь делятся на отдельные подпакеты. Например, пакет Основные элементы состоит из подпакетов: Элементы ядра, Вспомогательные элементы, Механизмы расширения и типы данных



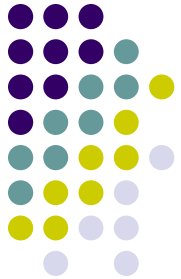
Пакет Элементы ядра



Этот пакет определяет основные абстрактные и конкретные компоненты, необходимые для разработки объектных

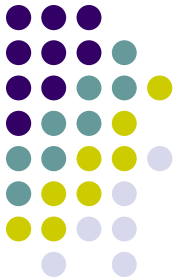
В этот пакет входят основные метаклассы языка UML: класс (Class), атрибут (Attribute), ассоциация (Association), ассоциация-класс (AssociationClass), конец ассоциации (AssociationEnd), свойство поведения (BehavioralFeature), классификатор (Classifier), ограничение (Constraint), тип данных (DataType), зависимость (Dependency), элемент (Element), право на элемент (ElementOwnership), свойство (Feature), обобщение (Generalization), элемент отношения обобщения (GeneralizableElement), интерфейс (Interface), метод (Method), элемент модели (ModelElement), пространство имен (Namespace), операция (Operation), параметр (Parameter), структурное свойство (StructuralFeature), правила правильного построения выражений (Well-formedness rules).

Пакет Вспомогательные элементы



- Он специфицирует дополнительные конструкции языка UML, которые расширяют пакет Элементы ядра
- В этот пакет входят следующие метаклассы: связывание (Binding), комментарий (Comment), компонент (Component), узел (Node), презентация (Presentation), уточнение (Refinement), цепочка зависимостей (Trace), потребление (Usage), элемент представления (ViewElement), зависимость (Dependency), элемент модели (ModelElement), правила правильного построения выражений (Well-formedness rules)

Пакет Механизмы расширения



Он специфицирует порядок включения в модель элементов с уточненной семантикой, а также модификацию отдельных компонентов языка UML для более точного отражения специфики моделируемых систем

Механизм расширения определяет семантику для стереотипов, ограничений и помеченных значений.

в языке UML предусмотрены три механизма расширения, которые могут использоваться совместно или раздельно для определения новых элементов модели с отличающимися семантикой, нотацией и свойствами от специфицированных в метамодели языка UML элементов. Такими механизмами являются: ограничение (Constraint), стереотип (Stereotype) и помеченное значение (TaggedValue).

Таким образом, механизмы расширения языка UML предназначены для выполнения следующих задач:

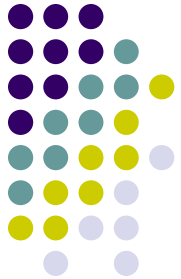
Уточнения существующих модельных элементов при разработке моделей на языке UML.

В спецификации самого языка UML для определения стандартных компонентов, которые либо не являются достаточно интересными, либо сложны для непосредственного определения в качестве элементов мета-модели UML.

Определения таких расширений языка UML, которые зависят от специфики моделируемого процесса или от языка реализации программного кода.

Присоединения произвольной семантической или несемантической информации к элементам модели.

Пакет Типы данных



Он специфицирует различные типы данных, которые могут использоваться в языке UML

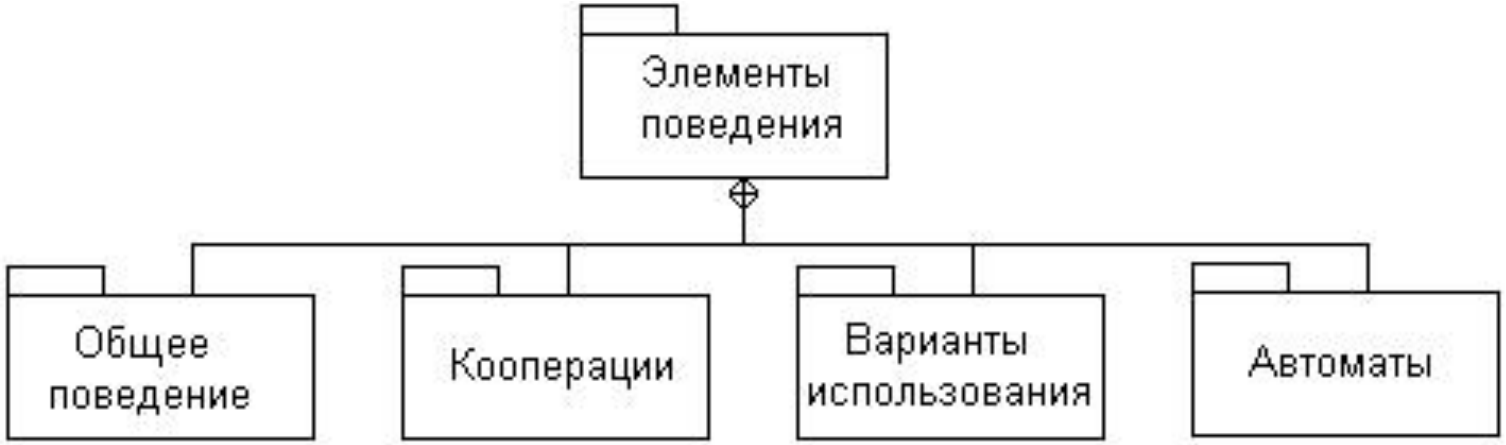
В метамодели UML типы данных используются для объявления типов атрибутов классов. Они записываются в форме строк текста на диаграммах и не имеют отдельного значка "тип данных". Благодаря этому происходит уменьшение размеров диаграмм без потери информации.

Для определения различных типов данных в языке UML используются ***как простые конструкции***: целое число (Integer), строка (String), имя (Name), Булев (Boolean), время (Time), кратность (Multiplicity), тип видимости (VisibilityKind), диапазон кратности (MultiplicityRange),
так и более сложные: выражение (Expression), булевское выражение (BooleanExpression), тип агрегирования (AggregationKind), тип изменения (ChangeableKind), геометрия (Geometry), отображение (Mapping), выражение-процедура (ProcedureExpression), тип псевдосостояния (PseudostateKind), выражение времени (TimeExpression), непрерываемый (Uninterpreted).

Пакет Элементы поведения

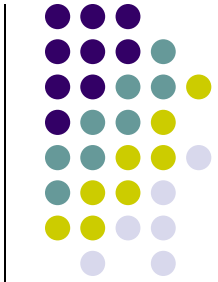
- Он является самостоятельной компонентой языка UML и, как следует из его названия, специфицирует динамику поведения в нотации UML

Состав пакета



В этом пакете специфицирована семантика для динамических элементов, которые включены в другие подпакеты элементов поведения. В пакет Общее поведение входит большое число элементов, таких как объект (Object), действие (Action), последовательность действий (ActionSequence), аргумент (Argument), экземпляр (Instance), исключение (Exception), связь (Link), сигнал (Signal), значение данных (DataValue), связь атрибутов (AttributeLink), действие вызова (CallAction), действие создания (CreateAction), действие уничтожения (DestroyAction).

Пакет Кооперации

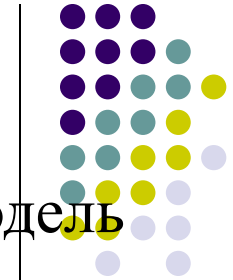


Он специфицирует контекст поведения при использовании элементов модели для выполнения отдельной задачи. В нем задается семантика понятий, которые необходимы для ответа на вопрос: "Как различные элементы модели взаимодействуют между собой с точки зрения структуры?”

В пакет Кооперации входят элементы: кооперация (Collaboration), взаимодействие (Interaction), сообщение (Message), роль ассоциации (AssociationRole), роль классификатора (ClassifierRole), роль конца ассоциации (AssociationEndRole).

Как можно догадаться из названия пакета, его элементы непосредственно используются при построении диаграмм кооперации.

Пакет Варианты использования



Пакет специфицирует поведение при включении в модель специальных конструкций, которые в языке UML называются актерами и вариантами использования. Эти понятия служат для определения функциональности моделируемой сущности, такой как система.
Особенность элементов этого пакета состоит в том, что они используются для первоначального определения поведения сущности без спецификации ее внутренней структуры.

В пакет Варианты использования кроме элементов актер (Actor) и вариант использования (UseCase) входят: расширение (Extension), точка расширения (ExtensionPoint), включение (Include) и экземпляр варианта использования (UseCaseInstance). Более подробно некоторые из этих понятий будут рассмотрены при описании диаграмм вариантов использования

Пакет Автоматы

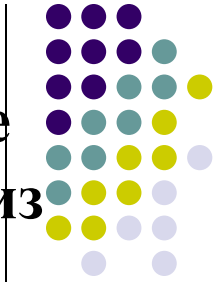
Пакет Автоматы специфицирует поведение при построении моделей с использованием систем переходов для конечного множества состояний. В нем определено множество понятий, которые необходимы для представления поведения модели в виде дискретного пространства с конечным числом состояний и переходов.

Автоматы могут использоваться для моделирования поведения индивидуальных сущностей, таких как экземпляры классов, а также для спецификации взаимодействий между сущностями, таких как кооперации

В пакет Автоматы входят элементы: состояние (State), переход (Transition), событие (Event), автомат (StateMachine), простое состояние (SimpleState), составное состояние CompositeState, псевдосостояние (PseudoState), конечное состояние (FinalState) и некоторые другие.

под состоянием в языке UML понимается абстрактный метакласс, используемый для моделирования ситуации или процесса, в ходе которых имеет место (обычно неявное) выполнение некоторого инвариантного условия. Примером такого инвариантного условия может быть состояние ожидания объектом выполнения некоторого внешнего события, например запроса или передачи управления.

Пакет Общие механизмы



В этом пакете определены общие механизмы, которые применимы ко всем моделям UML. Пакет состоит из единственного подпакета Управления моделями

Пакет Управление моделями (Model Management) специфицирует базовые элементы языка UML, которые необходимы для формирования всех модельных представлений.

В нем определяется семантика модели (Model), пакета (Package) и подсистемы (Subsystem).

Модель является подклассом пакета и представляет собой абстракцию физической системы, которая предназначена для вполне определенной цели. Именно эта цель предопределяет те компоненты, которые должны быть включены в модель и те, рассмотрение которых не является обязательным. Другими словами, модель отражает релевантные аспекты физической системы, оказывающие непосредственное влияние на достижение поставленной цели. В прикладных задачах цель обычно задается в форме исходных требований к системе, которые, в свою очередь, в языке UML записываются в виде вариантов использования системы.

2. Специфика описания метамодели языка UML

Метамодель языка UML описывается на некотором полуформальном языке с использованием трех видов представлений:

- Абстрактного синтаксиса
- Правил правильного построения выражений
- Семантики

К элементам абстрактного синтаксиса относятся некоторые ключевые слова и значения отдельных атрибутов базовых понятий уровня метамодели, которые имеют фиксированное обозначение в виде текста на естественном языке.

Правила правильного построения выражений используются для задания дополнительных ограничений или свойств, которыми должны обладать те или иные компоненты модели. Поскольку исходным понятием ООП является понятие класса, его общими свойствами должны обладать все экземпляры, которые в этом смысле должны быть инвариантны друг другу. Для задания этих инвариантных свойств классов и отношений необходимо использовать специальные выражения некоторого формального языка, в рамках UML получившего название языка объектных ограничений (Object Constraint Language).

Семантика языка UML описывается в основном на естественном языке, но может включать в себя некоторые дополнительные обозначения, вытекающие из связей определяемых понятий с другими понятиями.

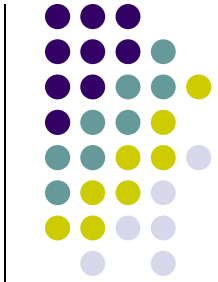
Требования к записи семантических элементов

Рекомендуется использовать следующие правила:

- Явно указывать в тексте экземпляр некоторого метакласса. Речь идет о том, что в естественной речи мы часто опускаем слово "пример" или "экземпляр", говоря просто "класс". Так, фразу "Атрибут возраст класса сотрудник имеет значение 30 лет" следует записать более точно, а именно: "Атрибут возраст экземпляра класса сотрудник имеет значение 30 лет".
- В каждый момент времени используется только то значение слова, которое приписано имени соответствующей конструкции языка UML. Все дополнительные особенности семантики должны быть указаны явным образом без каких бы то ни было неявных предположений.
- Термины языка UML могут включать только один из допустимых префиксов, таких как под-, супер- или мета-. При этом сам термин с префиксом записывается одним словом.

В дополнение к этому используются следующие правила :

- Если используются ссылки на конструкции языка UML, а не на их представления в метамодели, следует применять обычный текст без какого бы то ни было выделения.
- Имена метаклассов являются элементом нотации языка UML и представляют собой существительное и, возможно, присоединенное к нему прилагательное. В этом случае имя метакласса на английском записывается одним словом с выделением каждой составной части имени заглавной буквой (например, ModelElement, StructuralFeature).
- Имена метаассоциаций и ассоциаций классов записываются аналогичным образом (например, ElementReference).
- Имена других элементов языка UML также записываются одним словом, но должны начинаться с маленькой буквы (например, ownedElement, allContents).



- Имена метаатрибутов, которые принимают булевы значения, всегда начинаются с префикса "is" (например, isAbstract).
- Перечислимые типы должны всегда заканчиваться словом "Kind" (например, AggregationKind).
- При ссылках в тексте на метаклассы, метаассоциаций, метаатрибуты и т. д. должны всегда использоваться в точности те их имена, которые указаны в модели.
- Имена стандартных обозначений (стереотипов) заключаются в кавычки и начинаются со строчной буквы (например, "type").

3. Диаграммы UML и их виды

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций, получивших название диаграмм. В терминах языка UML определены следующие виды диаграмм:

- Диаграмма вариантов использования (use case diagram)
- Диаграмма классов (class diagram)
- Диаграммы поведения (behavior diagrams)
  - Диаграмма состояний (statechart diagram)
  - Диаграмма деятельности (activity diagram)
  - Диаграммы взаимодействия (interaction diagrams)
    - Диаграмма последовательности (sequence diagram)
    - Диаграмма кооперации (collaboration diagram)
- Диаграммы реализации (implementation diagrams)
  - Диаграмма компонентов (component diagram)
  - Диаграмма развертывания (deployment diagram)

Из перечисленных выше диаграмм некоторые служат для обозначения двух и более других подвидов диаграмм. При этом в качестве самостоятельных представлений в языке UML используются следующие диаграммы:

- Диаграмма вариантов использования.
- Диаграмма классов.
- Диаграмма состояний.
- Диаграмма деятельности .
- Диаграмма последовательности .
- Диаграмма кооперации.
- Диаграмма компонентов.
- Диаграмма развертывания.

Перечень этих диаграмм и их названия являются каноническими в том смысле, что представляют собой неотъемлемую часть графической нотации языка UML. Более того, процесс ООАП неразрывно связан с процессом построения этих диаграмм. При этом совокупность построенных таким образом диаграмм является самодостаточной в том смысле, что в них содержится вся информация, которая необходима для реализации проекта сложной системы.

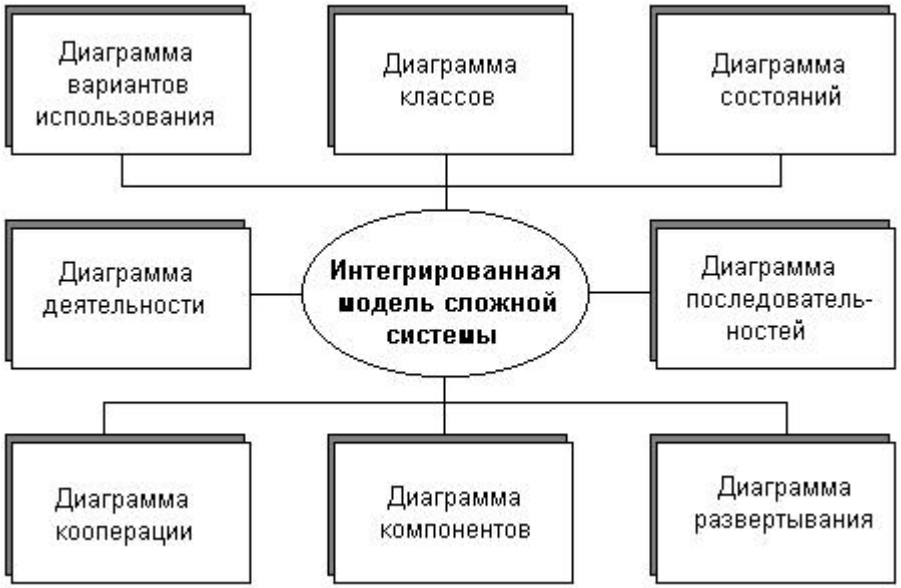


Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.

Диаграмма классов является, по своей сути, логической моделью, отражающей статические аспекты структурного построения сложной системы.

Диаграммы поведения также являются разновидностями логической модели, которые отражают динамические аспекты функционирования сложной системы.

Диаграммы реализации служат для представления физических компонентов сложной системы и поэтому относятся к ее физической модели

Рекомендации по оформлению диаграмм

- Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области. Речь идет о том, что в процессе разработки диаграммы необходимо учесть все сущности, важные с точки зрения контекста данной модели и диаграммы. Отсутствие тех или иных элементов на диаграмме служит признаком неполноты модели и может потребовать ее последующей доработки.
- Все сущности на диаграмме модели должны быть одного концептуального уровня. Здесь имеется в виду согласованность не только имен одинаковых элементов, но и возможность вложения отдельных диаграмм друг в друга для достижения полноты представлений. В случае достаточно сложных моделей систем желательно придерживаться стратегии последовательного уточнения или детализации отдельных диаграмм.
- Вся информация о сущностях должна быть явно представлена на диаграммах. Речь идет о том, что, хотя в языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию (например, в случае неявного указания видимости атрибутов и операций классов), необходимо стремиться к явному указанию свойств всех элементов диаграмм.

- Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем использовании на практике. Например, наличие замкнутых путей при изображении отношений агрегирования или композиции приводит к ошибкам в программном коде, который будет реализовывать соответствующие классы. Наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен также приводит к неоднозначной интерпретации и может служить источником проблем.
- Диаграммы не следует перегружать текстовой информацией. Принято считать, что визуализация модели является наиболее эффективной, если она содержит минимум пояснительного текста. Как правило, наличие больших фрагментов развернутого текста служит признаком недостаточной проработанности модели или ее неоднородности, когда в рамках одной модели представляется различная по характеру информация. Поскольку общая декомпозиция модели на отдельные типы диаграмм способна удовлетворить самые детальные представления разработчиков о системе, важно уметь правильно отображать те или иные сущности и аспекты моделирования в соответствующие элементы канонических диаграмм.

- Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов. Любые пояснительные тексты, которые не являются собственными элементами диаграммы (например, комментариями), не должны приниматься во внимание разработчиками. В то же время отдельные достаточно общие фрагменты диаграмм могут уточняться или детализироваться на других диаграммах этого же типа, образуя вложенные или подчиненные диаграммы. Таким образом, модель системы на языке UML представляет собой пакет иерархически вложенных диаграмм, детализация которых должна быть достаточной для последующей генерации программного кода, реализующего проект соответствующей системы.
- Количество типов диаграмм для конкретной модели приложения не является строго фиксированным. Речь идет о том, что для простых приложений нет необходимости строить все без исключения типы диаграмм. Некоторые из них могут просто отсутствовать в проекте системы, и этот факт не будет считаться ошибкой разработчика. Например, модель системы может не содержать диаграмму развертывания для приложения, выполняемого локально на компьютере пользователя. Важно понимать, что перечень диаграмм зависит от специфики конкретного проекта системы.

4. Rational Unified Process и его содержание

Процесс ООАП в контексте языка UML получил специальное название — рациональный унифицированный процесс (Rational Unified Process, RUP). Концепция RUP и основные его элементы разработаны А. Джекобсоном в ходе его работы над языком UML.

Суть концепции RUP заключается в последовательной декомпозиции или разбиении процесса ООАП на отдельные этапы, на каждом из которых осуществляется разработка соответствующих типов канонических диаграмм модели системы.

На начальных этапах RUP строятся логические представления статической модели структуры системы, затем — логические представления модели поведения, и лишь после этого — физические представления модели системы.

В результате RUP должны быть построены канонические диаграммы на языке UML, при этом последовательность их разработки в основном совпадает с их последовательной нумерацией.