Билет 6.

1. Нестабильность схемы FTCS. Метод Лакса.

Задачи с начальными условиями для уравнения сохранения потока.

Большой класс одномерных задач с начальными условиями можно привести к уравнению сохранения потока: $\frac{\partial u}{\partial t}=-\frac{\partial F(u)}{\partial x}$ (2.1) где **u** и **F** векторы, и в некоторых случаях **F** зависит не только от **u**, но и от пространственных производных от **u**.

Например для одномерного волнового уравнения с постоянной скоростью распространения волны

$\frac{\partial^2 u}{\partial t^2}=v^2\frac{\partial^2 u}{\partial^2 x}$ (2.2) можно написать два уравнения первого порядка

$$\frac{\partial r}{\partial t}=v\frac{\partial s}{\partial x}$$
$$\frac{\partial s}{\partial t}=v\frac{\partial r}{\partial x}$$

(2.3) где

$$r=v\frac{\partial u}{\partial x}$$
$$s=\frac{\partial u}{\partial t}$$

(2.4) В этом случае r и s – две компоненты **u**, а поток определяется линейным матричным уравнением $F(u)=\begin{pmatrix}0 & -v\\ -v & 0\end{pmatrix}\cdot u$ (2.5)

Уравнения (2.3) аналогичны уравнениям Максвелла для одномерной электромагнитной волны. Рассмотрим уравнение для скалярной величины u $\frac{\partial u}{\partial t}=-v\frac{\partial u}{\partial x}$ (2.6) Общее решение этого уравнения – волна, распространяющаяся в положительном направлении оси x. $u=f(x-vt)$ (2.7)

Введем дискретные точки по x и t $x_j=x_0+j\Delta x$ $j=0,1,\ldots,J$ $t_n=t_0+n\Delta t$ $n=0,1,\ldots,N$ (2.8)

Пусть u_j обозначает $u(t_n,x_j)$. Мы можем различными способами представить производную по времени, самый очевидный: $\frac{\partial u}{\partial t}|_{j,n}=\frac{u_j^{n+1}-u_j^n}{\Delta t}+O(\Delta t)$ (2.9) Это приближение первого порядка точности по временному шагу. Для пространственной производной используем приближение второго порядка.

$\frac{\partial u}{\partial x}|_{j,n}=\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}+O(\Delta x^2)$ (2.10) Тогда конечно-разностная аппроксимация рассматриваемого уравнения, называемая FTCS (Forward Time Centered Space) примет вид: $\frac{u_j^{n+1}-u_j^n}{\Delta t}=-v\left(\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}\right)$ (2.11) Это выражение легко представляется как выражение для u_j в момент времени n+1. Это явная схема, т.е. u_j в момент времени n+1 можно вычислить явно по известным величинам. FTCS алгоритм - это пример одноуровневой схемы, т.е. для нахождения u_j в момент времени n+1 достаточно значений в момент времени n.

Анализ стабильности фон Неймана

Представим, что коэффициенты рассматриваемых уравнений с частными производными меняются очень слабо в пространстве и времени, и их можно считать

постоянными. В этом случае независимые решения или собственные моды уравнений можно искать в виде $u_j^n=\xi^n e^{ikj\Delta x}$ (2.12)

где k действительное пространственное волновое число, которое может принимать любое значение, и ξ=ξ(k) – комплексное волновое число, которое зависит от k. Временная зависимость единственной собственной моды ни что иное, как последовательные (возрастающие) целые степени комплексной величины ξ. Следовательно, разностные уравнения нестабильны, если |ξ(k)|>1 для каких-то значений k. Подставляя (.1.12) в (.1.11), получим $\xi(k)=1-i\frac{v\Delta t}{\Delta x}\sin k\Delta x$ (2.13)

Как видно из этого выражения |ξ(k)|>1 **для всех k**, т.е. схема FTCS нестабильна.

Несмотря на возможно недостаточную строгость метода фон Неймана, он практически всегда дает верный ответ.

Метод Лакса.

Простое изменение схемы, предложенное Лаксом, позволило улучшить ситуацию. Он предложил заменить значение функции u в узле j в момент времени n в производной по времени на среднее значение в узлах j-1 и j+1. Это привело к следующему выражению:

$u_j^{n+1}=\frac{1}{2}(u_{j+1}^n+u_{j-1}^n)-\frac{v\Delta t}{2\Delta x}(u_{j+1}^n-u_{j-1}^n)$ (2.15) В этом случае для величины ξ получим:

$\xi=\cos k\Delta x-i\frac{v\Delta t}{\Delta x}\sin k\Delta x$ (2.16) Условие стабильности - квадрат модуля |ξ| должен быть меньше или равен единицы_ приводит к условию: $\frac{|v|\Delta t}{\Delta x}\le 1$ (2.17)

Это знаменитый критерий стабильности Куранта – Фридриха – Леви. Сравнитм схему Лакса и FTCS, для этого перепишем схему Лакса в виде:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=-v\left(\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}\right)+\frac{1}{2}\left(\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta t}\right) \quad (2.18)$$

Но это схема FTCS для уравнения: $\frac{\partial u}{\partial t}=-v\frac{\partial u}{\partial x}+\frac{(\Delta x)^2}{2\Delta t}\nabla^2 u$ (2.19)

Т.е. в уравнении появился “диссипационный” член. Если |v|Δt в точности не равно Δx, то амплитуда волны спадает.

2. Два типа диаграмм взаимодействий: диаграммы кооперации и диаграммы последовательностей.

На диаграмме последовательностей иллюстрируются события, инициированные в системе исполнителями. Создание диаграмм последовательностей является существенной частью исследования возможных способов построения системы. Поэтому данный процесс выполняется в рамках создания модели анализа.

Для иллюстрации инициируемых в системе событий можно создать диаграмму последовательностей, воспользовавшись при этом входящей в состав языка UML системой обозначений.

Диаграммы последовательностей выполняются на стадии анализа цикла разработки. Этот процесс зависит от предварительного формирования прецедентов.

Прежде чем приступать к разработке логики работы программного приложения, необходимо исследовать и определить ее поведение как "черного ящика" Поведение системы (system behavior) представляет собой описание того, какие действия выполняет система, без определенного механизма их реализации. Одной из составляющих такого описания является диаграмма последовательностей.

Диаграммы последовательностей

Прецеденты определяют, как исполнители взаимодействуют с программной системой. В процессе этого взаимодействия исполнителем генерируются события, передаваемые системе, которые представляют собой запросы на выполнение некоторой операции.

Было бы неплохо отделить и проиллюстрировать операции систем, выполнение которых запрашивает исполнитель, поскольку они важны для понимания поведения системы. В качестве системы обозначений в состав языка UML входят диаграммы последовательностей (sequence diagram), С их помощью можно проиллюстрировать взаимодействие исполнителя i; системой и операции, выполнение которых при этом инициируется.

Диаграмма последовательностей системы (system sequence diagram) является схемой, которая для определенного сценария прецедента показывает, генерируемые внешними исполнителями события, их порядок, а также события, генерируемые внутри самой системы. При этом все системы рассматриваются как черный ящик. Назначение данной диаграммы - отображение событий, передаваемых исполнителями системе через ее границы.

Системные события и операции

Системное событие (system event) — это внешнее входное событие, сгенерированное для системы исполнителем. Событие инициирует выполнение определенной операции. Системная операция (system operation) является операцией, которую система выполняет в ответ на сгенерированное событие. Запись системных операций. В языке UML операции записываются как тип.

В состав языка UML входят диаграммы взаимодействий (interaction diagrams). Они иллюстрируют способ взаимодействия объектов с помощью сообщений и обеспечивают выполнение необходимых задач. Диаграммы взаимодействий создаются на стадии проектирования цикла разработки. Их создание зависит от следующих созданных ранее артефактов.

• Концептуальная модель. С ее помощью разработчик может определить программные классы, соответствующие понятиям. Объекты этих классов участвуют во взаимодействиях, иллюстрируемых на диаграммах взаимодействий.

• Описание системных операций. С помощью описаний разработчик идентифицирует обязанности и постусловия, которым должны удовлетворять диаграммы взаимодействий.

• Реальные (или идеальные) прецеденты. Из описания прецедентов разработчик может почерпнуть информацию о том, выполнению каких задач должны удовлетворять диаграммы взаимодействий. Эта информация дополняет данные, содержащиеся в описаниях системных операций.

Диаграмма взаимодействий иллюстрирует процесс обмена сообщениями между экземплярами (и классами) в модели классов. Отправной точкой такого взаимодействия является удовлетворение постусловий в описаниях операций.

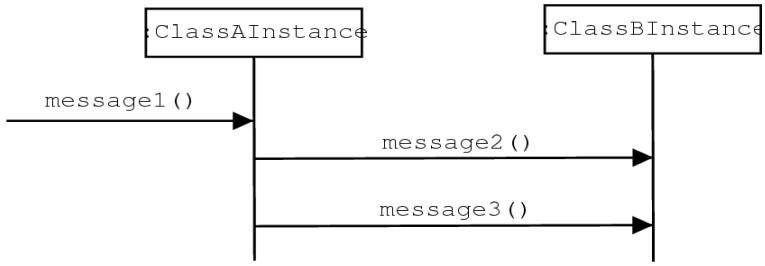
В языке UML определено два типа диаграмм взаимодействий, которые могут использоваться для иллюстрации либо похожего, либо идентичного обмена сообщениями.

1. Диаграммы кооперации (collaboration diagram).

2. Диаграммы последовательностей (sequence diagram).

Диаграммы кооперации иллюстрируют взаимодействие объектов в формате графа или сети.

Диаграммы последовательностей иллюстрируют взаимодействие в форме, показанной ниже.



Диаграммы кооперации являются исключительно выразительными и способны представлять больше информации, вытекающей из контекста. Кроме того, диаграммы кооперации с точки зрения занимаемого пространства оказываются более экономными. Однако обе системы обозначений позволяют представить подобные конструкции.

Диаграммы взаимодействий — это важный артефакт

Для успешного конструирования диаграмм взаимодействий принципы разработки предварительно можно систематизировать и проанализировать. Такой подход к пониманию и использованию этих принципов основывается на шаблонах (patterns), представляющих собой структурированные рекомендации и принципы.

Создание диаграмм кооперации. При создании диаграмм кооперации руководствуйтесь следующими рекомендациями.

1. В текущем цикле разработки создавайте отдельную диаграмму для каждой системной операции. Для сообщения системной операции разрабатывайте диаграмму таким образом, чтобы это сообщение было входным.

2. Если диаграмма оказалась слишком сложной (например, ее очень трудно разместить на листе бумаги формата А4), разбейте ее на диаграммы меньшего размера.

3. В качестве отправной точки используйте обязанности и постусловия, указанные в описании операции, а также описание прецедентов. Разрабатывайте диаграмму взаимодействий с учетом решения этих задач. Для создания профессиональных диаграмм применяйте GRASP и другие шаблоны,

В прецедентах неявно представлены системные события, которые явно отображаются на диаграммах последовательностей

• Наилучшие исходные предположения о результатах выполнения системных операций описываются в контрактах

• Системные операции определяют сообщения, которые являются отправной точкой создания диаграмм взаимодействий; эти диаграммы иллюстрируют, как объекты взаимодействуют между собой и обеспечивают выполнение поставленных задач

В языке UML для иллюстрации экземпляров объектов используется простой и непротиворечивый подход.

• Для экземпляра любого элемента языка UML (класса, исполнителя и т.д.) используется то же графическое обозначение, что и для типа, однако при этом соответствующая определяющая строка подчеркивается.

Таким образом, для отображения экземпляра класса на диаграмме взаимодействий используется обычное, графическое, условное обозначение класса, однако при этом его имя подчеркивается. Кроме того, на диаграмме кооперации перед именем класса всегда должно указываться двоеточие (:).

Связь (link) является соединением между двумя экземплярами классов, определяющим некоторую форму перемещения и видимости между ними. Более строго можно сказать, что связь является экземпляром ассоциации. При взаимодействии клиента с сервером имеется два таких экземпляра. Существование маршрута перемещения от клиента к серверу означает, что сообщения могут передаваться от клиента серверу.

Передаваемые между объектами сообщения представляются в виде помеченных стрелок над линиями связей. Над одной линией связи может быть указано любое количество сообщений. Для отображения порядка следования сообщений в текущем потоке управления рядом с сообщением приводится порядковый номер.

Параметры сообщения могут указываться внутри круглых скобок, находящихся за именем сообщения Дополнительно может быть указан также тип параметра.

Для сообщения может быть указано возвращаемое значение. Для этого перед сообщением нужно добавить имя переменной с этим значением, а также символ операции присваивания. Дополнительно может быть указан также тип возвращаемого значения.

В языке UML определен стандартный синтаксис описания сообщений. return:= message(parameter: parameterType) : returnType. Сообщение может передаваться объектом самому себе.

Итерационный процесс можно отобразить, указав за порядковым номером сообщения символ "*". Это означает, что сообщение передается получателю несколько раз, в цикле . Можно также указать циклическое условие, определив тем самым, что значения возвращаются многократно. Для представления нескольких сообщений, передаваемых в одном и том же цикле (например, набора сообщений в цикле for), повторяйте условие итерации для каждого сообщения.

Сообщением создания объекта, не зависящим от используемого языка, является create (создать). Оно передается создаваемому экземпляру объекта. Дополнительно новый экземпляр объекта может включать символ "new". Сообщение new может содержать параметры, определяющие передаваемые начальные значения.

Порядок передачи сообщений иллюстрируется с помощью порядковых номеров (sequence number). При этом используется следующая схема нумерации.

1. Первое сообщение не нумеруется. Таким образом, сообщение msg1( ) является непронумерованным.
2. Порядок и вложенность последующих сообщений отображается в соответствии с принятой схемой нумерации. При ее использовании к вложенным сообщениям добавляется номер. Вложенность означает, что к номеру исходящего сообщения добавляется номер входящего сообщения.
Условное сообщение изображается с помощью номера, за которым в квадратных скобках указывается условное выражение, аналогичное условию цикла. Сообщение передается только в том случае, если условный оператор возвращает значение true.
Представление коллекций. Сложный объект обычно реализуется в виде группы экземпляров объектов, содержащихся в контейнере или объекте-коллекции, например в объекте vector стандартной библиотеки шаблонов (STL — Standard Template Library) языка C++, объекте Vector языка Java. Однако такие объекты могут и отсутствовать. В этом случае сложный объект будет представлять логический набор экземпляров объектов.
Сообщение, передаваемое сложному объекту, направляется самому объекту. В языке UML версии 1.1 сообщения, передаваемые сложному объекту, не направляются каждому его элементу (как было в предыдущих версиях языка).
Сообщения могут передаваться самому классу, а не его экземплярам. Это может понадобиться, например, для вызова методов класса. Например, в языке Java такие методы являются статическими, а в языке Smalltalk — методами класса.
Все достаточно просто. Сообщение изображается как обычно, однако в условном обозначении класса его имя не подчеркнуто. Тем самым указывается, что это сообщение передается самому классу, а не его экземпляру. Очень важно, чтобы там, где это нужно, имена экземпляров были подчеркнуты. В противном случае передаваемые сообщения могут быть неверно интерпретированы.