Методы и инструменты анализа требований

Кулаков Кирилл Александрович

Методы анализа

- Распространены три подхода к формированию требований: метод, основанный на множестве опорных точек зрения, сценарии и этнографический метод.
- Другие подходы, которые могут использоваться в процессе разработки требований, — это методы структурного анализа и методы прототипирования.
- Не существует универсального подхода к формированию и анализу требований. Обычно для разработки требований одновременно используется несколько подходов.

Метод опорных точек зрения

- Различные точки зрения на проблему позволяют увидеть ее с разных сторон. Однако эти взгляды не являются полностью независимыми и обычно перекрывают друг друга, а потому могут служить основой общих требований.
- Подход с использованием различных опорных точек зрения к разработке требований признает эти точки зрения и использует их в качестве основы построения и организации как процесса формирования требований, так и непосредственно самих требований.

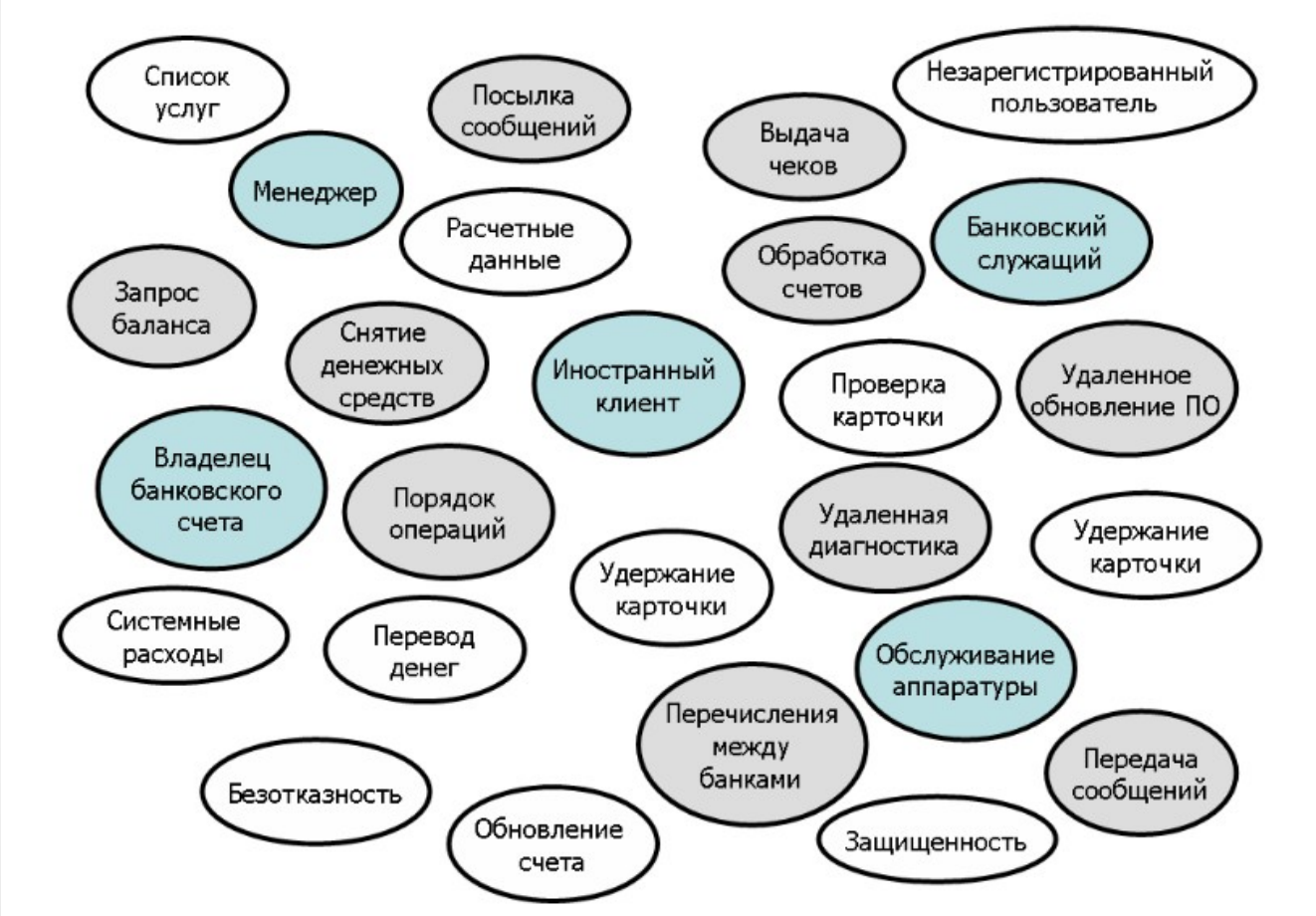
Метод опорных точек зрения

- Различные методы предлагают разные трактовки выражения "точка зрения". Точки зрения можно трактовать следующим образом:
  - Как источник информации о системных данных. В этом случае на основе опорных точек зрения строится модель создания и использования данных в системе. В процессе формирования требований отбираются все такие точки зрения, на их основе определяются данные, которые будут созданы или использованы при работе системы, и способы обработки этих данных.
  - Как получатели системных сервисов. В этом случае точки зрения являются внешними (относительно системы) получателями системных сервисов. Точки зрения помогают определить данные, необходимые для выполнения системных сервисов или их управления.

Метод опорных точек зрения

- Наиболее эффективным подходом к анализу интерактивных систем является использование внешних опорных точек зрения. Эти точки зрения взаимодействуют с системой, получая от нее сервисы и продуцируя данные и управляющие сигналы.
- Этот тип точек зрения имеет ряд преимуществ:
  - Точки зрения, внешние к системе, — естественный способ структурирования процесса формирования требований.
  - Сравнительно просто решить, какие точки зрения следует оставить в качестве опорных: они должны отображать какой-либо способ взаимодействия с системой.
  - Данный подход полезен для создания нефункциональных требований, с которыми можно связать какой-либо сервис.

Пример идентификации точек зрения



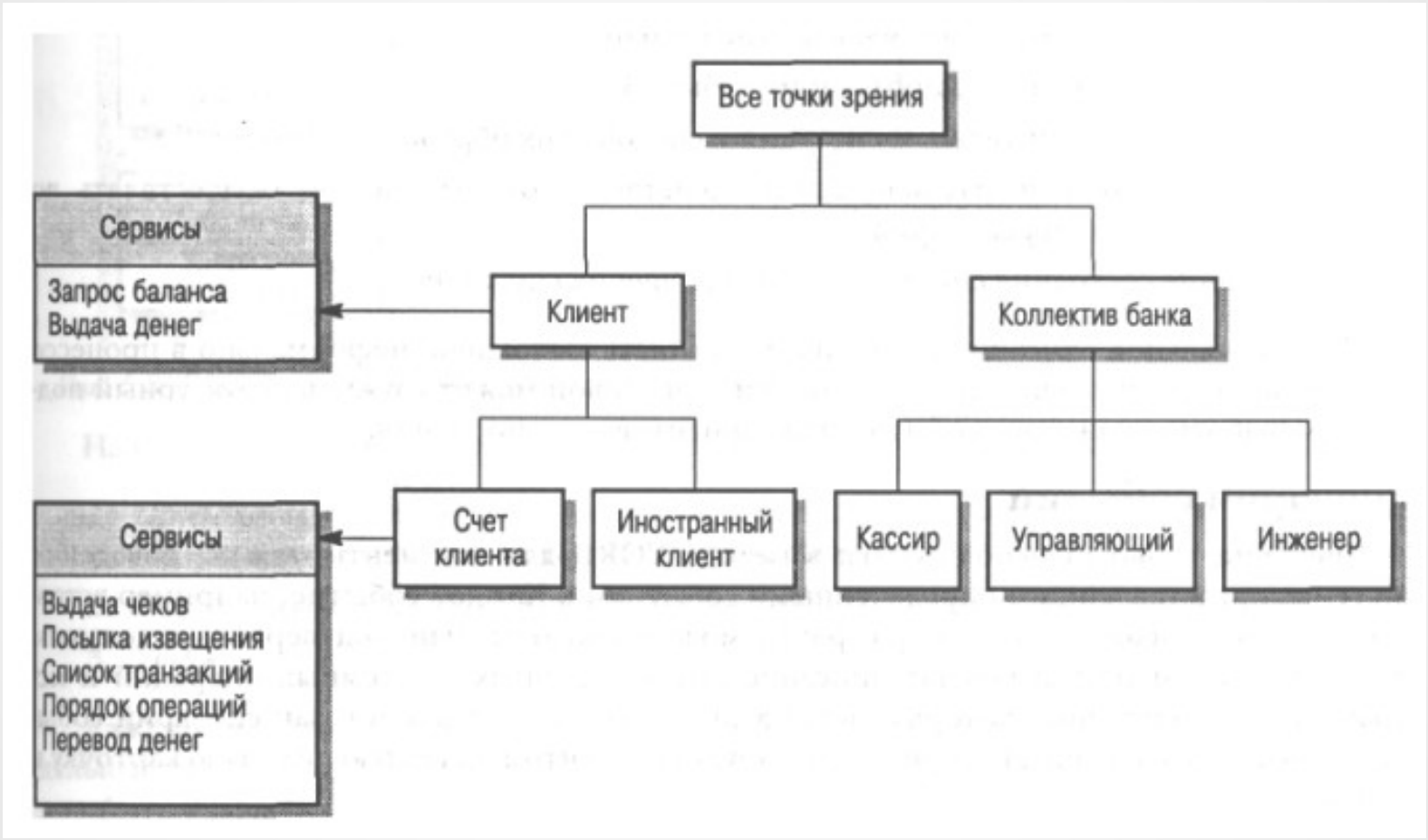
Пример идентификации точек зрения

ВЛАДЕЛЕЦ СЧЕТА	ИНОСТРАННЫЙ КЛИЕНТ	КАССИР БАНКА
Список сервисов	Список сервисов	Список сервисов
Выдача денег Запрос баланса Выдача чеков Посылка сообщения Список транзакций Порядок операций Перевод денег	Выдача денег Запрос баланса	Выполнение диагностики Зачисление денег Обработка счетов Посылка сообщения

Метод опорных точек зрения

- Информация, извлеченная из точек зрения, используется для заполнения форм шаблонов точек зрения и организации точек зрения в иерархию наследования.
- Сервисы, данные и управляющая информация наследуются подмножеством точек зрения.

Пример иерархии точек зрения



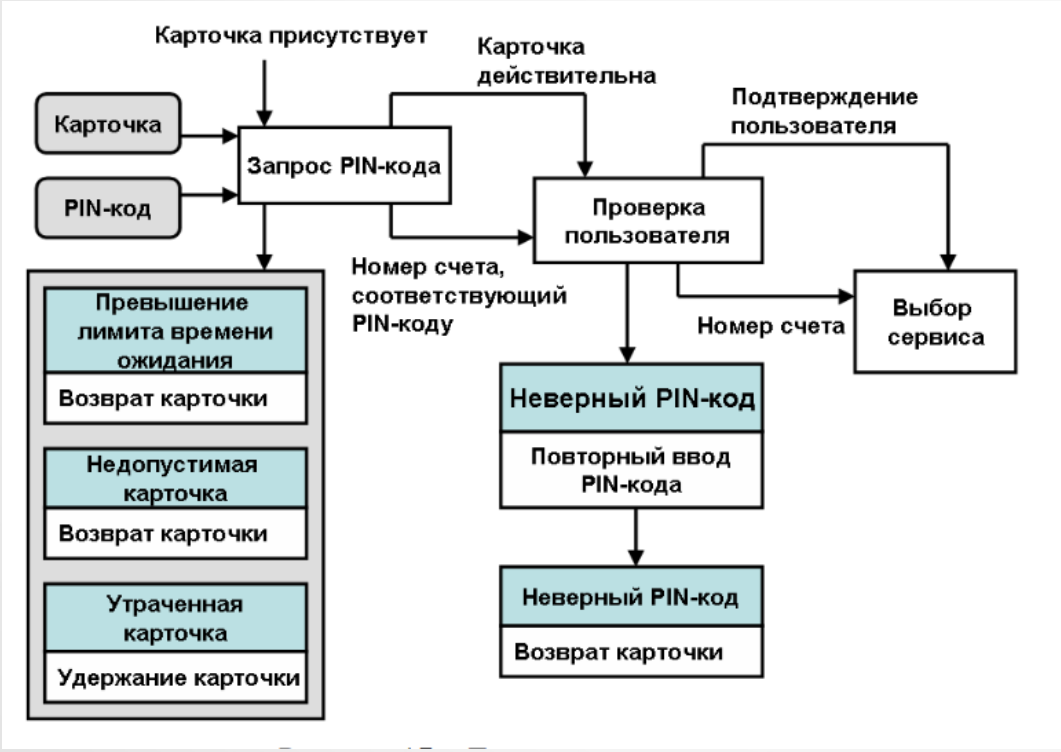
Метод сценариев

- Сценарии особенно полезны для детализации уже сформулированных требований, поскольку описывают последовательность интерактивной работы пользователя с системой. Каждый сценарий описывает одно или несколько возможных взаимодействий.
- Сценарий начинается с общего описания, затем постепенно детализируется для создания полного описания взаимодействия пользователя с системой.
- В большинстве случаев сценарий включает следующее:
  - Описание состояния системы после завершения сценария.
  - Информацию относительно других действий, которые можно осуществлять во время выполнения сценария.
  - Описание исключительных ситуаций и способов их обработки.
  - Описание нормального протекания событий.
  - Описание состояния системы в начале сценария.

Метод сценариев

- Сценарии событий используются для документирования поведения системы, представленного определенными событиями.
- Сценарии включают описание потоков данных, системных операций и исключительных ситуаций, которые могут возникнуть

Пример сценариев



- Условные обозначения:
- Данные, поступающие в систему или исходящие из нее, представлены в эллипсах.
- Управляющая информация показана стрелками в верхней части прямоугольников.
- Внутрисистемные данные показаны справа от прямоугольников.
- Исключительные ситуации показаны в нижней части прямоугольников.
- Имя следующего события, ожидаемого после завершения сценария, приводится в затененном прямоугольнике.

Варианты использования (use-case)

- Варианты использования (use-case) — это методика формирования требований, основанная на сценариях. Они стали основой нотаций в языке моделирования UML при описании объектных моделей систем.

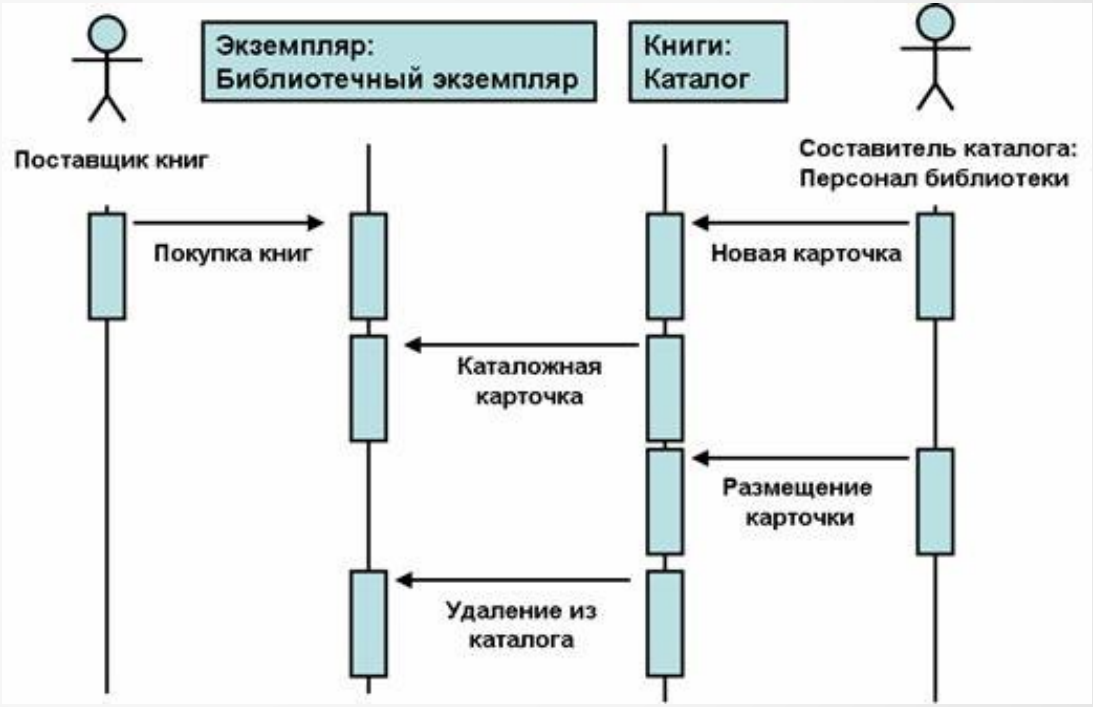
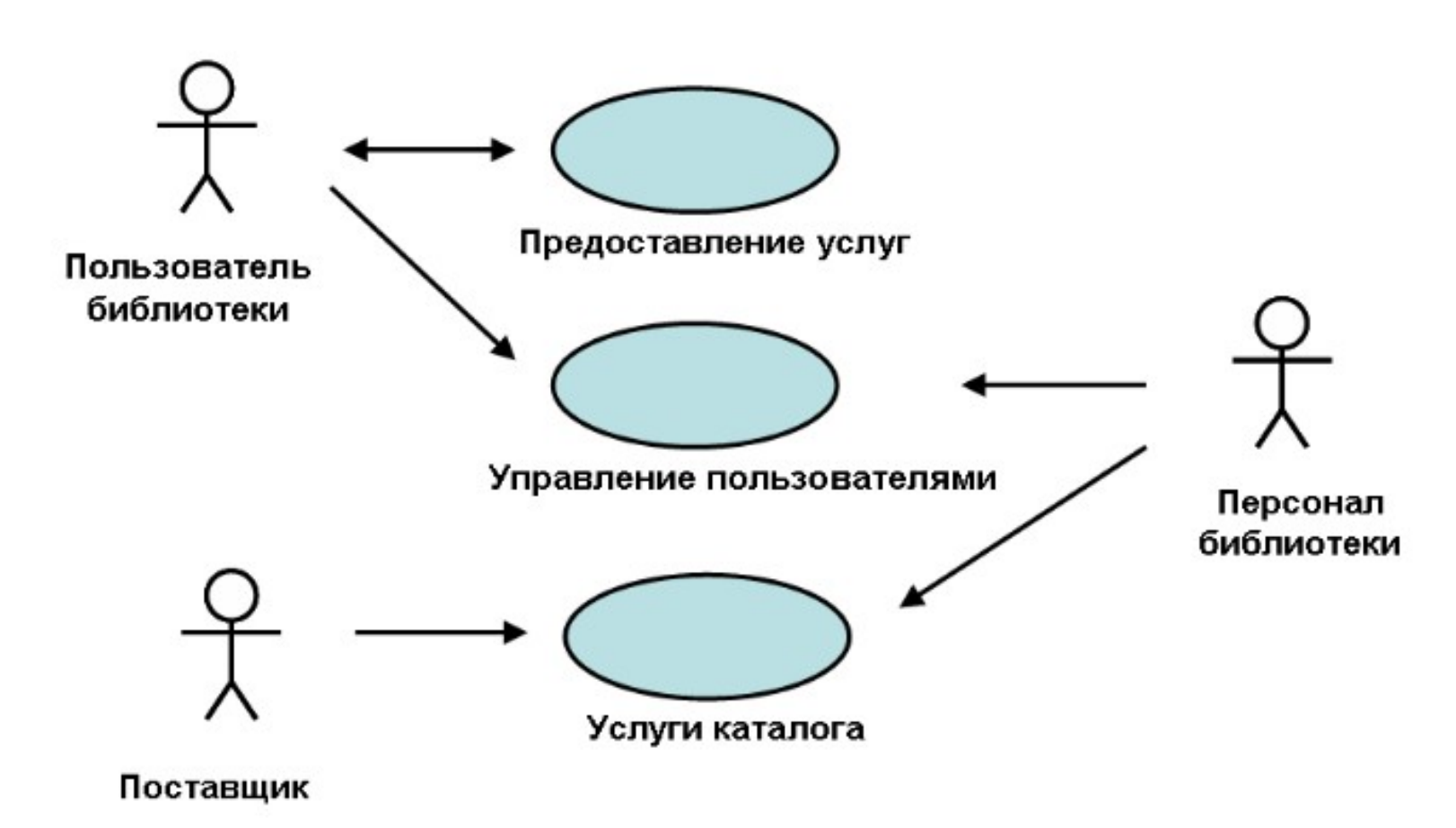


Диаграмма вариантов использования



Этнографический подход

- Этнографический подход к формированию системных требований используется для понимания и формирования социальных и организационных аспектов эксплуатации системы.
  - Разработчик требований погружается в рабочую среду, где будет использоваться система.
  - Его ежедневная работа связана с наблюдением и протоколированием реальных действий, выполняемых пользователями системы.
  - Значение этнографического подхода заключается в том, что он помогает обнаружить неявные требования к системе, которые отражают реальные аспекты ее эксплуатации, а не формальные умозрительные процессы.
- Этнографический подход позволяет детализировать требования для критических систем, чего не всегда можно добиться другими методами разработки требований. Однако, поскольку этот метод ориентирован на конечного пользователя, он не может охватить все требования предметной области и требования организационного характера.

Схема этнографического подхода

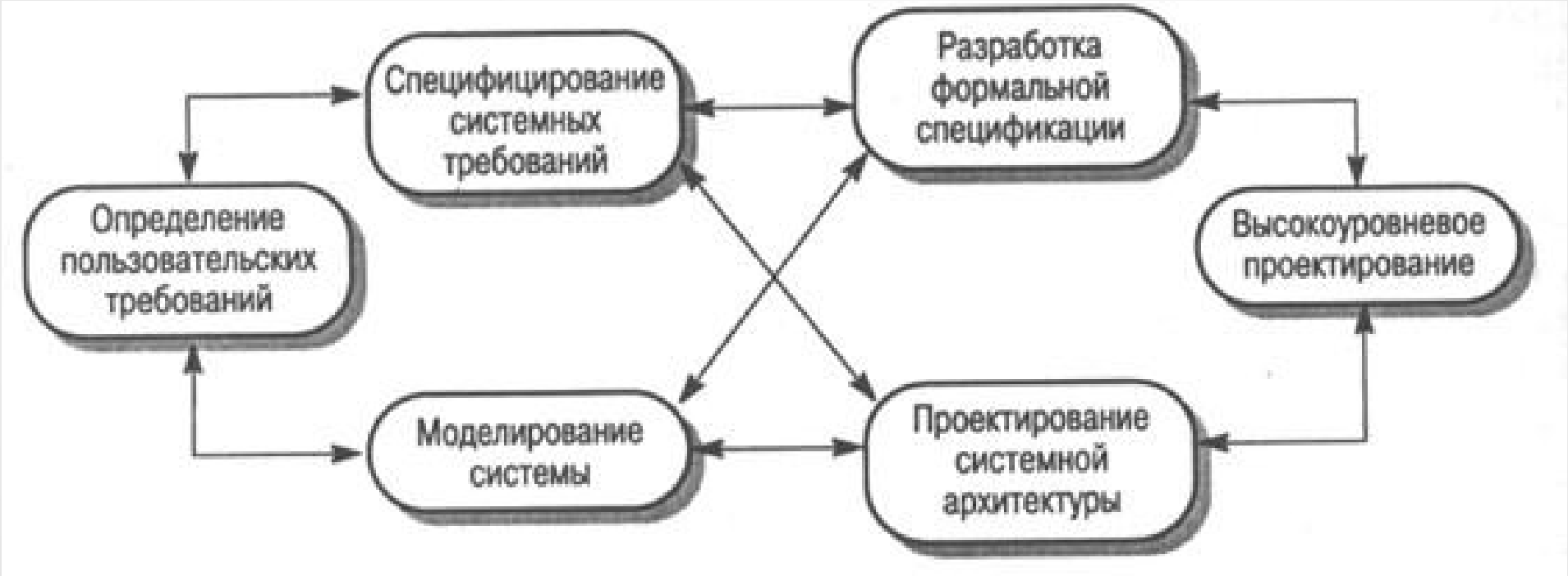
Формальные спецификации

- Так называемые формальные методы разработки программных систем широко не используются.
  - Многие компании, разрабатывающие ПО, не считают экономически выгодным применение этих методов в процессе разработки.
- Термин "формальные методы" подразумевает ряд операций, в состав которых входят:
  - создание формальной спецификации системы
  - анализ и доказательство спецификации
  - реализация системы на основе преобразования формальной спецификации в программы
  - верификация программ.
- Все эти действия зависят от формальной спецификации программного обеспечения.
- Формальная спецификация — это системная спецификация, записанная на языке, словарь, синтаксис и семантика которого определены формально.
- Необходимость формального определения языка предполагает, что этот язык основывается на математических концепциях.
- Здесь используется область математики, которая называется дискретной математикой и основывается на алгебре, теории множеств и алгебре логики.

Формальные спецификации

- В инженерии ПО определены три уровня спецификации программного обеспечения.
  - пользовательские требования
  - системные требования
  - спецификация структуры программной системы.
- Пользовательские требования наиболее обобщенные, спецификация структуры наиболее детальна.
- Формальные математические спецификации находятся где-то между системными требованиями и спецификацией структуры.
- Они не содержат деталей реализации системы, но должны представлять ее полную математическую модель.
- Разработка спецификация и проектирование могут выполняться параллельно, когда информация от этапов разработки спецификации передается к этапам проектирования и наоборот.

Схема разработки требований с использованием формальных спецификаций



Формальные спецификации

- Создание формальной спецификации требует детального анализа системы, который позволяет обнаружить ошибки и несоответствия в спецификации неформальных требований.
- Проблемы в требованиях, которые остаются необнаруженными до последних стадий процесса разработки ПО, обычно требуют больших затрат на исправление.
- Существует два основных подхода к разработке формальной спецификации, которые используются для написания детализированных спецификаций нетривиальных программных систем.
  - Алгебраический подход, при котором система описывается в терминах операций и их отношений.
  - Подход, ориентированный на моделирование, при котором модель системы строится с использованием математических конструкций, таких, как множества и по следовательности, а системные операции определяются тем, как они изменяют со стояния системы.
- Для разработки формальных спецификаций последовательных и параллельных систем в настоящее время создано несколько языков.

Языки построения формальных спецификаций

Тип языка	***Последовательные системы***	***Параллельные системы***
Алгебраический	Larch, OBJ	Lotos
Основанный на моделях	Z, VDM, B	CSP, сети Петри

Подсистемы и интерфейсы

- Большие системы обычно разбиваются на подсистемы, которые разрабатываются независимо друг от друга.
  - Подсистемы могут использовать другие подсистемы, поэтому необходимой частью процесса специфицирования является определение интерфейсов подсистем.
  - Если интерфейсы определены и согласованы, подсистемы можно разрабатывать независимо друг от друга.
- Интерфейс подсистемы часто определяется как набор абстрактных типов данных и объектов, при этом только через интерфейс доступны описание данных и операции над ними.
  - Спецификацию интерфейса подсистемы можно рассматривать как объединение спецификаций компонентов, что в итоге и составит описание интерфейса подсистемы.
- Точные спецификации интерфейсов подсистем необходимы для разработчиков, которые пишут программный код, обращающийся к сервисам других подсистем.
  - Спецификации интерфейсов содержат информацию о том, какие сервисы доступны в других подсистемах и как получить к ним доступ.
  - Ясный и однозначный интерфейс подсистем уменьшает вероятность ошибок во взаимоотношениях между ними.

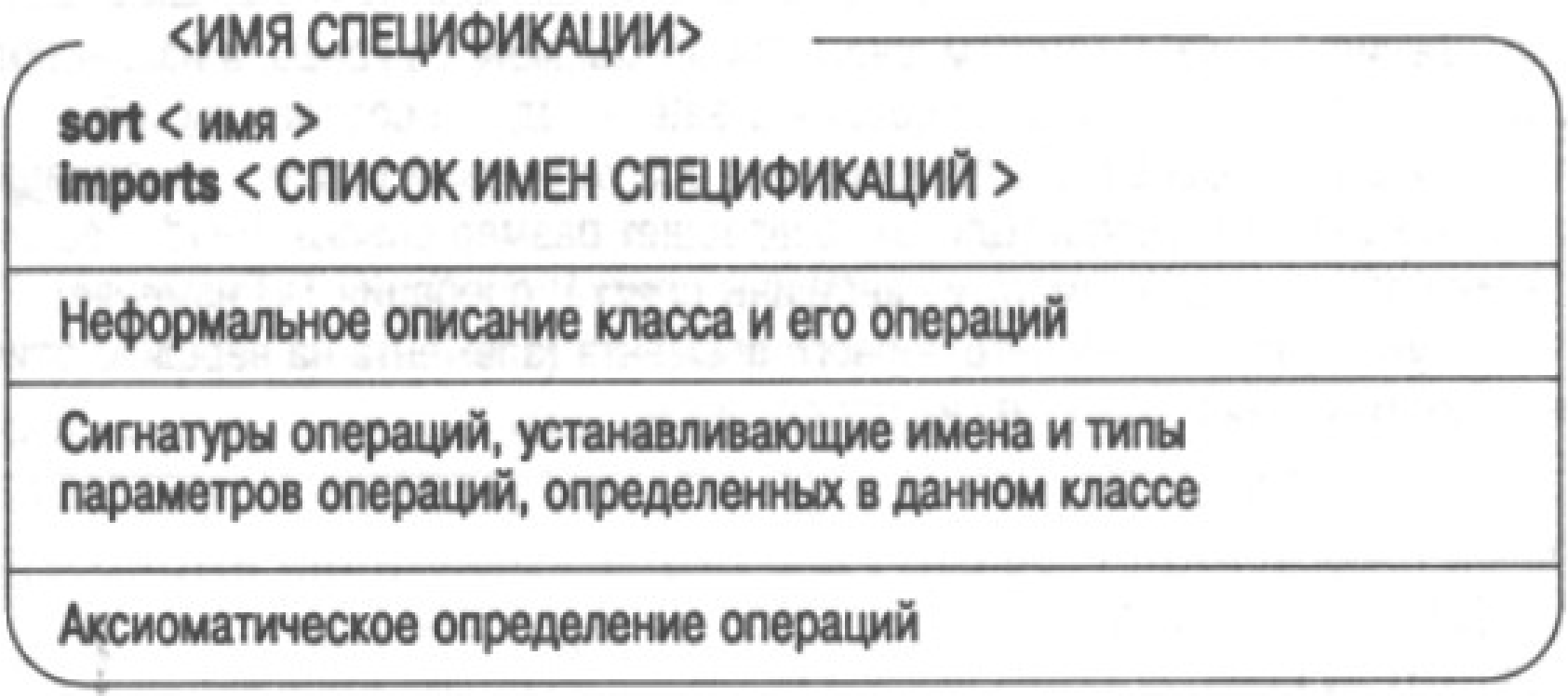
Алгебраический подход

- Алгебраический подход первоначально был разработан для описания интерфейсов абстрактных типов данных, где типы данных определяются скорее спецификациями операций над данными, чем способом представления самих данных.
- Это очень похоже на определение классов объектов.
- Алгебраический подход к формальным спецификациям определяет абстрактный тип данных в терминах операций над данными.

Алгебраический подход

- Структура спецификации объекта состоит из четырех компонентов:
  - Введение, где объявляется класс (sort) объектов. Класс — это общее название для множества объектов. Он обычно реализуется как тип данных. Введение может также включать объявление импорта (imports), где указываются имена спецификаций, определяющие другие классы. Импортирование спецификаций делает эти классы доступными для использования.
  - Описательная часть, в которой неформально описываются операции, ассоциированные с классом. Это делает формальную спецификацию более простой для понимания. Формальная спецификация дополняет это описание, обеспечивая однозначный синтаксис и семантику операций.
  - Часть сигнатур, в которой определяется синтаксис интерфейса объектного класса или абстрактного типа данных. Здесь описываются имена операций, количество и типы их параметров, а также классы выходных результатов операции.
  - Часть аксиом, где определяется семантика операций посредством создания ряда аксиом, которые характеризуют поведение абстрактного типа данных. Эти аксиомы связывают операции создания объектов класса с операциями, проверяющими их значения.

Структура алгебраической спецификации



Формальные спецификации

- Процесс разработки формальной спецификации интерфейса подсистемы включает следующие действия.
  - Структурирование спецификации. Представление неформальной спецификации интерфейса в виде множества абстрактных типов данных или объектных классов. Также неформально определяются операции, ассоциированные с каждым классом.
  - Именование спецификаций. Задаются имена для каждой спецификации абстрактного типа, определяются параметры спецификаций (если они необходимы) и имена определяемых классов.
  - Определение операций. На основании списка выполняемых интерфейсом функций для каждой спецификации определяется связанный с ней набор операций. Необходимо предусмотреть операции по созданию экземпляров классов, по изменению значений экземпляров классов и по проверке этих значений. Вероятно, придется добавить новые функции к первоначально определенному списку функций интерфейса.
  - Неформальная спецификация операций. Написание неформальной спецификации для каждой операции, где должно быть указано, как операции воздействуют на определяемый класс.
  - Определение синтаксиса операций. Определение синтаксиса и параметров для каждой операции. Это часть сигнатуры формальной спецификации.
  - Определение аксиом. Определение семантики операций путем описания условий, которые должны выполняться для различных комбинаций операций.

Формальные спецификации

- Операции над абстрактным типом данных обычно относятся к одному из двух классов.
  - Операции конструирования, которые создают или изменяют объекты класса. Обычно их называют Create (Создать), Update (Изменить), Add (Добавить) или Cons (Конструирование).
  - Операции проверки, которые возвращают атрибуты класса. Обычно им дают имена, соответствующие именам атрибута, или имена, подобные Eval (Значение), Get (Получить) и т.п.
- Хорошим эмпирическим правилом для написания алгебраической спецификации является создание аксиом для каждой операции конструирования с применением всех операций проверки. Это означает, что если есть m операций конструирования и n операций проверки, то должно быть определено m x n аксиом.
- Операции конструирования, связанные с абстрактным типом данных, часто очень сложны и могут определяться через другие операции конструирования и проверки. Если операции конструирования определены посредством других операций, то необходимо определить операции проверки, используя более примитивные конструкции.
- В спецификации списка операциями конструирования являются Create, Cons и Tail, которые создают списки. Операциями проверки являются Head и Length, которые используются для получения значений атрибутов списка. Операция Tail не является примитивной конструкцией, поэтому для нее можно не определять аксиомы с использованием операций Head и Length, но в таком случае Tail необходимо определить посредством примитивных конструкций.

Формальные спецификации

- При написании алгебраических спецификаций часто используется рекурсия. Результат операции Tail — список, сформированный из входного списка путем удаления верхнего элемента. Это определение подсказывает, как использовать рекурсию для построения данной операции. Операция определяется на пустых списках, затем рекурсивно переходит на непустые списки и завершается, когда результатом снова будет пустой список.
- Иногда проще понять рекурсивные преобразования, используя короткий пример. Предположим, что есть список [5, 7], где элемент 5 — начало (вершина) списка, а элемент 7— конец списка. Операция Cons([5, 7], 9) должна возвратить список [5, 7, 9], а операция Tail, примененная к этому списку, должна возвратить список [7, 9]. Приведем последовательность рекурсивных преобразований, приводящую к этому результату.
- Tail ([5,7,9]) =
- = Tail (Cons ([5, 7], 9)) =
- = Cons (Tail ([5, 7]), 9) =
- = Cons (Tail (Cons ([5], 7)), 9) =
- = Cons (Cons (Tail ([5]), 7), 9) =
- = Cons (Cons (Tail (Cons ([], 5)), 7), 9) =
- = Cons (Cons ([Create], 7), 9) =
- = Cons ([7], 9) =
- = [7, 9]

Формальные спецификации

- Здесь систематически использовались аксиомы для Tail, что привело к ожидаемому результату. Аксиому для операции Head можно проверить подобным способом.
- Простые алгебраические методы подходят для описания интерфейсов, когда операции, ассоциированные с объектом, не зависят от состояния объекта. Тогда результаты любой операции не зависят от результатов предыдущих операций. Если это условие не выполняется, алгебраические методы могут стать громоздкими. Более того, я думаю, что алгебраические описания поведения систем часто искусственны и трудны для понимания.
- Альтернативным подходом к созданию формальных спецификаций, который широко используется в программных проектах, является спецификация, основанная на моделях системы. Такие спецификации используют модели состояний системы. Системные операции определяются посредством изменений состояний системной модели. Таким образом определяется поведение системы.

Модели систем

- Одной из широко используемых методик документирования системных требований является построение ряда моделей системы. Эти модели используют графические представления, показывающие решение как исходной задачи, для которой создается система, так и разрабатываемой системы. Модели являются связующим звеном между процессом анализа исходной задачи и процессом проектирования системы.
- Модели могут представить систему в различных аспектах:
  - Внешнее представление, когда моделируется окружение или рабочая среда системы.
  - Описание поведения системы, когда моделируется ее поведение.
  - Описание структуры системы, когда моделируется системная архитектура или структуры данных, обрабатываемых системой.

Модели систем

- Наиболее важным аспектом системного моделирования является то, что оно опускает детали. Модель является абстракцией системы и легче поддается анализу, чем любое другое представление этой системы. В идеале представление системы должно сохранять всю информацию относительно представляемого объекта. Абстракция является упрощением и определяется выбором наиболее важных характеристик системы.
- Типы системных моделей, которые могут создаваться в процессе анализа систем:
  - Модель обработки данных. Диаграммы потоков данных показывают последовательность обработки данных в системе.
  - Композиционная модель. Диаграммы "сущность-связь" показывают, как системные сущности составляются из других сущностей.
  - Архитектурная модель. Эти модели показывают основные подсистемы, из которых строится система.
  - Классификационная модель. Диаграммы наследования классов показывают, какие объекты имеют общие характеристики.
  - Модель "стимул-ответ". Диаграммы изменения состояний показывают, как система реагирует на внутренние и внешние события.

Модели систем

- Модели системного окружения
  - Простые структурные модели обычно дополняются моделями других типов, например моделями процессов, которые показывают взаимодействия в системе, или моделями потоков данных, которые показывают последовательность обработки и перемещения данных внутри системы и между другими системами в окружающей среде.
- Модели потока данных
  - это интуитивно понятный способ показа последовательности обработки данных внутри системы. Нотации, используемые в этих моделях, описывают обработку данных с помощью системных функций, а также хранение и перемещения данных между системными функциями.
  - В диаграммах потоков данных используются следующие обозначения: закругленные прямоугольники соответствуют этапам обработки данных; стрелки, снабженные примечаниями с названием данных, представляют потоки данных; прямоугольники соответствуют хранилищам или источникам данных.
  - Модели потоков данных показывают функциональную структуру системы, где каждое преобразование данных соответствует одной системной функции. Иногда модели потоков данных используют для описания потоков данных в рабочем окружении системы. Такая модель показывает, как различные системы и подсистемы обмениваются информацией. Подсистемы окружения не обязаны быть простыми функциями.

Модели систем

- Поведенческие модели
  - Эти модели используются для описания общего поведения системы. Обычно рассматривают два типа поведенческих моделей — модель потоков данных и модель конечного автомата. Эти модели можно использовать отдельно или совместно, в зависимости от типа разрабатываемой системы.
- Модели конечных автоматов
  - используются для моделирования поведения системы, реагирующей на внутренние или внешние события. Такая модель показывает состояние системы и события, которые служат причиной перехода системы из одного состояния в другое.
  - Модели конечных автоматов являются неотъемлемой частью методов проектирования систем реального времени. Такие модели определяются диаграммами состояний, которые стали основой системы нотаций в языке моделирования UML.

Модели систем

- Модели «сущность-связь»
  - Наиболее широко используемой методологией моделирования данных является моделирование типа "сущность-связь". Для описания структуры обрабатываемой информации модели данных часто используются совместно с моделями потоков данных.
  - Проекты структуры ПО представляются ориентированными графами. Они состоят из набора узлов различных типов, соединенных дугами, отображающими связи между структурными узлами.
  - Подобно всем графическим моделям, модели "сущность-связь" недостаточно детализированы, поэтому они обычно дополняются более подробным описанием объектов, связей и атрибутов, включенных в модель. Эти описания собираются в словари данных или репозитории.
- Объектные модели
  - могут использоваться как для представления данных, так и для процессов их обработки. В этом отношении они объединяют модели потоков данных и семантические модели данных. Они также полезны для классификации системных сущностей и могут представлять сущности, состоящие из других сущностей.
  - Класс объектов - это абстракция множества объектов, которые определяются общими атрибутами и сервисами (операциями).
  - Объекты - это исполняемые сущности с атрибутами и сервисами класса объектов. Объекты представляют собой реализацию класса. На основе одного класса можно создать много различных объектов.

Модели систем

- Важным этапом объектно-ориентированного моделирования является определение классов объектов, которые затем систематизируются. Это подразумевает создание схемы классификации, которая показывает, как классы объектов связаны друг с другом посредством общих атрибутов и сервисов.
- Схема классификации организована в виде иерархии наследования, на вершине которой представлены наиболее общие классы объектов. Более специализированные объекты наследуют их атрибуты и сервисы. Эти объекты могут иметь собственные атрибуты и сервисы.
- В нотации UML наследования показываются сверху вниз. Здесь стрелка (с окончанием в виде треугольника) выходит из класса, который наследует атрибуты и операции, и направлена к родительскому классу.
- Объекты могут создаваться из нескольких объектов. Такой объект агрегируется из совокупности других объектов. Классы, представляющие такие объекты, можно смоделировать, используя модель агрегирования объектов.
- В UML для показа агрегирования объектов используется связь с окончанием ромбовидной формы.

CASE-средства

- Средства, которые входят в пакет инструментальных средств:
  - Редакторы диаграмм предназначены для создания диаграмм потоков данных, иерархий объектов, диаграмм "сущность-связь" и т.д.
  - Средства проектирования, анализа и проверки выполняют проектирование ПО и создают отчет об ошибках и дефектах в системной архитектуре.
  - Центральный репозиторий позволяет проектировщику найти нужный проект и соответствующую проектную информацию.
  - Словарь данных хранит информацию об объектах, которые используются в структуре системы.
  - Средства генерирования отчетов на основе информации из центрального репозитория автоматически генерируют системную документацию.
  - Средства создания форм определяют форматы документов и экранных форм.
  - Средства импортирования и экспортирования позволяют обмениваться информацией из центрального репозитория различным инструментальным средствам.
  - Генераторы программного кода автоматически генерируют программы на основе проектов, хранящихся в центральном репозитории.