Требуется разработать модели программы «Игра в “Кости”» (или другой программы на выбор – «Некая информационная система по обработке желаний пользователя») на основе процедурной и объектно-ориентированной парадигм.

Должны быть разработаны следующие диаграммы UML.

Для представления с точки зрения функциональности:

- Диаграмма **прецедентов**;

Для представления с точки зрения проектирования:

- Диаграммы **классов** и **состояний** (для структурного моделирования)
- Диаграммы **взаимодействия** и **деятельности** (для моделирования поведения).

Практическое занятие 1. Спецификация требований и создание вариантов использования

Цель

Уяснить принципы выявления и документирования требований к ПО.

Порядок выполнения

В ходе выполнения заданий вы должны выявить предметную область программной системы – в данном случае уяснить правила программной системы (игры «Игра в “Кости”») и определить требования к программе. Разработать спецификацию требований и варианты использования.

Описание программы

Цель игры заключается в том, чтобы, следуя определенной стратегии получить максимальный выигрыш.

Сценарий.

1. Игра включает в себя сеансы, в каждом из которых игрок сначала делает ставку. После выполнения игровых действий игроков (в качестве соперника может выступать компьютер) в зависимости от результатов ставка изменяется в соответствующую сторону. Игра продолжается до тех пор, пока баланс игроков положительный, а также после определенного розыгрыша игрок может закончить игру и забрать выигрыш.

2. Содержание сеанса (партии): за два или три броска требуется набрать максимальное число очков, но не больше 15 – игрок бросает кубик два раза, если сумма очков за два броска больше 9, то ему предлагается либо остановиться, либо бросить третий раз. Результатом игры будет соответственно сумма очков за броски или в случае превышения 15 очков – “перебор” (автоматический проигрыш).

- ✓ Согласно замыслу задания вы можете добавить любую функциональность в описание игры.

Реализация начальной фазы процесса разработки

На основе исходных данных (сымитировать роли заказчика и пользователя – разрешается вносить свои пожелания и требования и как заказчика по функционированию системы, и как исполнителя) требуется:

1. Сформировать концепцию (**vision**) – понимание проекта в целом, сформулировать бизнес-цели, которых хочет достичь заказчик от внедрения программной системы.
2. Определить разнообразных пользователей с различными ожиданиями и целями, т.е. классифицировать их по отношению к вашему продукту – создать типы пользователей и роли.
3. Разработать пользовательские истории (**user story**) на основе бесед с пользователями.
4. Основываясь на пользовательских историях и бизнес-целях определить основную функциональность (сформулировать функциональные требования), которую должна предоставлять система, представить ее описание в табличном виде (см. табл.1).
5. На основе функциональных требований создать модель прецедентов (описание вариантов использования) в текстовой (сценарной) форме и в виде диаграммы Use Case.
6. Разработать модель предметной области (рекомендуется в виде диаграммы) – определить основные сущности в терминах предметной области основные классы понятий (концептуальные классы) предметной области и отношения между ними и словарь данных (data dictionary), который представляет собой набор подробной информации об используемых в приложении сущностях данных в виде единого ресурса, служащего в дальнейшем для определения типов данных.

Рекомендации по разработке концепции программной системы

Концепция (видение) проекта содержит обобщенное представление долгосрочных целей и назначение нового продукта. В этом документе обычно отражают сбалансированное описание требований, удовлетворяющих различных

заинтересованных лиц. В [1, глава 5, с. 91] описывается порядок создания документа концепции и показан шаблон [с. 97], состоящий из ключевых слов, которые подходят для документа о концепции продукта.

Составьте сжатое положение о концепции проекта, обобщающее долгосрочные цели и назначение нового продукта. В этом документе следует отразить сбалансированную концепцию, удовлетворяющую различных заинтересованных лиц. Она может быть несколько идеалистичной, но должна основываться на существующих или предполагаемых рыночных факторах, архитектуре предприятия, стратегическом направлении развития корпорации или ограничениях ресурсов. Примените шаблон [с. 97], состоящий из ключевых слов, которые подходят для документа о концепции продукта (изучите пример формулировки концепции для приложения системы контроля химикатов):

• **Для** [целевая аудитория покупателей] • **Который** [положение о потребностях или возможностях] • **Эта (этот)** [имя продукта] • **Является** [категория продукта] • **Который(ая)** [основные функции, ключевое преимущество, основная причина для покупки или использования] • **В отличие от** [основной конкурирующий продукт, текущая система или текущий бизнес-процесс] • **Наш продукт** [положение об основном отличии и преимуществе нового продукта].

Рекомендации по выявлению пользователей программной системы

Тип (класс) пользователей является подмножеством пользователей продукта, которые являются подмножеством клиентов продукта, а те в свою очередь являются подмножеством заинтересованных лиц. Один человек может принадлежать разным классам пользователей, например, администратор иногда работает с системой как рядовой пользователь. Помимо всего прочего, пользователей продукта можно подразделять по характерным признакам и группировать их на отдельные классы. Изучите предлагаемые в [1, с.115] квалификационные различия и выделите отдельные классы и роли пользователей. На данном этапе рекомендуется обратить внимание на следующие признаки: по привилегиям доступа и уровню безопасности (например, рядовой пользователь, пользователь-гость, администратор), а также по задачам, которые им приходится решать при выполнении бизнес-операций.

Рекомендации по разработке модели программной системы

Модель программной системы показывает функции системы (собственно варианты использования), их окружение (актеры) и связи (отношения) между прецедентами и актерами.

Пользовательские истории составляются в свободной форме, в виде историй или некоторых сценариев использования системы [4, 5]. Каждая история имеет условного рассказчика (автора, пользователя) истории, повествующего о наиболее значимых для исполнения требований к проектируемой программной системе.

Рекомендуется пользовательские истории составлять в соответствии с неким шаблоном, например, «*Как <тип пользователя>, я хочу <цель>, чтобы <причина>»* [1, с.168]. Эти истории являются основой программной модели, которая показывает функции системы (собственно варианты использования), их окружение (актеры) и связи (отношения) между прецедентами и актерами.

Эти истории являются основой программной модели, которая показывает функции системы (собственно варианты использования), их окружение (актеры) и связи (отношения) между прецедентами и актерами.

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. Вариант использования (use case) описывает последовательность взаимодействия системы и внешнего действующего лица, в результате которого действующее лицо получает полезный результат [1, с.171]. Класс пользователя (который не обязательно должен быть человеком) в пользовательской истории соответствует основному действующему лицу в варианте использования. Изучите примеры соответствия вариантов использования и пользовательских историй [1, с.168, таблица 8-2].

Разработка диаграммы вариантов использования преследует следующие цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы ([1] глава 5).
- сформулировать общие требования к функциональному поведению проектируемой системы ([1] глава 8).
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей ([1] главы 1-4).

– подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями ([1] глава 10.).

Таким образом, диаграммы вариантов использования позволяют получить высокоуровневое визуальное представление о требованиях пользователей. Изучите пример диаграммы варианта использования в нотации UML [1, с.172, рис. 8-2].

На диаграммах вариантов использования каждое действующее лицо и вариант использования рекомендуется сопровождать описанием (сценарием), поскольку такие текстовые описания служат средством связи между лицами, часто не имеющими специальной подготовки. Поэтому простой текст обычно является наилучшим выбором. Описание варианта использования должно включать в себя пояснение, предусловие, потоки событий (основной и альтернативные, если таковые есть) и постусловие [1, с.175-180].

Изучите возможные варианты представления текстовых сценариев на основе предлагаемых шаблонов [1, с.174; 2, (вся книга об этом); 3, с. 101-107], выберите наиболее подходящий для вашей задачи.

Составьте несколько вариантов использования для вашего текущего проекта на основании выбранного шаблона. Включите в них все альтернативные направления развития и исключения.

На основе вариантов использования следует сформулировать функциональные требования, необходимые для успешной реализации каждого варианта использования. В этом случае придется установить соответствие между вариантами использования и связанными с ними функциональными требованиями. Функциональные требования можно писать с точки зрения того, что делает система или что делает пользователь. Требования следует излагать последовательно, например «Система должна» или «Пользователь должен», затем — активный глагол, а после — наблюдаемый результат, также надо указать инициирующие условия или триггеры, вследствие которых система ведет себя определенным образом. Изучите общие шаблоны требований [1, с. 243] и характерные примеры удачных и неудачных формулировок [1, с. 244-258].

Функциональные требования к системе («рамки решения») и соответствующие им **задачи разработчика** («рамки проекта») требования рекомендуется оформить в таблице (см. табл. 1).

Таблица 1.

ID	Описание «рамки решения»	ID	Описание «рамки проекта»
Функция: *Название функции*			
1	Система обеспечивает возможность пользователям……	101	Создание меню……
		102	Реализация просмотра результатов….

Проверьте варианты использования с клиентами (если их нет, то можно провести мыслительный эксперимент), чтобы убедиться в правильности определения модели программной системы, а также в том, что рассмотрены все вариации основного направления, учтены и обработаны все исключения, которые клиенты посчитали разумными.

Рекомендации по разработке модели предметной области и словаря данных

Словарь данных (data dictionary) представляет собой набор подробной информации об используемых в приложении сущностях данных. Сбор информации о составе, типах данных, разрешенных значениях и т.п. в виде единого ресурса, служащего для определения критериев проверки данных, помогает разработчикам правильно в дальнейшем реализовывать алгоритмы и уменьшает проблемы интеграции. Словарь данных является дополнением к модели предметной области, в которой определяются термины предметной области или бизнес-термины приложения, сокращения и акронимы.

Изучите типы элементов данных [1, с.295] и составьте словарь данных для вашего текущего проекта на основании примера словаря данных [1, с.296, рис. 13-4], обратите внимание на возможность наличия в словаре структур данных и повторяющихся групп.

Отчетность

Отчет должен содержать:

- описание исходных данных проекта;
- перечень требований и соответствующие им задачи разработчика согласно табл.1;
- диаграмму вариантов использования;
- текстовые описания (сценарии) вариантов использования
- модель предметной области в любом виде;
- системную диаграмму последовательностей;
- выводы.

В выводах необходимо определить, правильно ли реализована начальная фаза процесса разработки ПО, в этом случае она должна устанавливать высокоуровневые требования для желаемой и осуществимой системы. Неадекватная фаза делает систему далекой от желаемой, плохо описанной и слишком дорогой, в результате чего она никогда не будет реализована.

Литература к занятию

1. *Вигерс Карл* Разработка требований к программному обеспечению/Пер. с англ. — М.: Издательство-торговый дом «Русская Редакция», 2012. — 576 с.
2. *Алистер Коберн* Современные методы описания функциональных требований к системам /Пер. с англ. — М.: Издательство Лори, 2002. — 288 с.
3. *Ларман Крэг*. Применение UML 2.0 и шаблонов проектирования. 3-е издание.: Пер. с англ. — М. : Вильямс, 2013. – 736 с.
4. *Буч Гради, Максимчук Роберт А., Энгл Майкл У, Янг Бобби Дж., Коналлен Джим, Хьюстон Келли А*. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. - М.: Вильямс, 2008. - 720 с.
5. Как писать качественные пользовательские истории. [Электронный ресурс]. Режим доступа: https://netology.ru/blog/history-klient
6. Как писать User Story. [Электронный ресурс]. Режим доступа: https://medium.com/@alexandertvar/%D0%BA%D0%B0%D0%BA-%D0%BF%D0%B8%D1%81%D0%B0%D1%82%D1%8C-user-story-2410093b23c2

Практическое занятие 2. Проектирование алгоритма решения задачи согласно процедурной парадигме

Цель

Уяснить принципы разработки программ согласно процедурной парадигме.

Порядок выполнения

Для полученных в первой работе вариантов использования требуется описать возможность использования программы с помощью функций.

Выполнить проектирование требуемых функций – разработать функциональную модель согласно стандарту IDEF0.

Модель должна быть реализована в виде следующих диаграмм:

- контекстная диаграмма (диаграмма верхнего уровня);
- диаграмма декомпозиции 1-го уровня;
- две диаграммы декомпозиции 2-го уровня для двух наиболее интересных блоков с диаграммы декомпозиции 1-го.

Для каждой функции продумать передаваемые, внутренние и выходные параметры.

Обосновать выбранные типы данных.

Отчетность

Отчет должен содержать:

– структурную схему, соответствующую потокам событий передачи параметров в функции с соответствующими пояснениями;
– описание выбранных типов данных;
– выводы.

Литература к занятию

1. РД IDEF0 - 2000. Методология функционального моделирования IDEF0. [Электронный ресурс] Режим доступа: idef0_Стандарт.pdf.

Реализация объектно-ориентированной программы

Практическое занятие 3. Проектирование объектной модели

Цель

Уяснить принципы разработки диаграмм взаимодействия (в частности последовательностей), соответствующие потокам событий вариантов использования.

Порядок выполнения

Для полученных в первой работе вариантов использования требуется описать сценарий использования с помощью диаграммы последовательности.

Рекомендации по разработке

Диаграмма взаимодействий акцентирует внимание на временной упорядоченности сообщений. Графически такая диаграмма представляет собой таблицу, объекты в которой располагаются вдоль оси X, а сообщения в порядке возрастания времени – вдоль оси Y.

Диаграммы взаимодействий бывают двух видов: диаграмма последовательностей (sequence diagram) и диаграмма коммуникации

(communication diagram). В данном задании рассматривается диаграмма последовательностей.

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования, и демонстрирует взаимодействие объектов, упорядоченное по времени.

Объект является представлением сущности либо из реального мира, либо концептуальной модели. Объект может представлять нечто конкретное, например, грузовик или компьютер, или концепцию, например химический процесс, банковскую транзакцию, чек на покупку.

Класс описывает группу объектов с общими свойствами (атрибутами), общим поведением (операциями), общими связями с другими объектами и общей семантикой (шаблон для создания объекта).

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами.

Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

При этом диаграмма последовательности имеет как бы два измерения. Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии.

Второе измерение диаграммы последовательности – вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим.

Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни. Внутри прямоугольника записываются имя объекта и имя класса, разделенные двоеточием. При этом вся запись подчеркивается, что является признаком объекта, который, как известно, представляет собой экземпляр класса.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия.

Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме

последовательности образуют некоторый порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Отчетность

Отчет должен содержать:

– диаграммы взаимодействия, соответствующие потокам событий вариантов использования с соответствующими пояснениями;
– выводы.

Инструменты для создания диаграмм

1. https://sequencediagram.org/
2. https://online.visual-paradigm.com/diagrams/templates/sequence-diagram/

Практическое занятие 4. Проектирование объектно-ориентированной программы

Цель

Разработать диаграммы классов, состояний и деятельности.

Порядок выполнения

На основе диаграммы последовательности разработать диаграмму классов. Для каждого класса продумать атрибуты, конструкторы и методы.

Обратите особое внимание на отношения между классами, подпишите тип отношения (является, содержит или реализует) над каждой стрелкой на диаграмме.

Реализуйте, где это необходимо, механизм композиции. В этом случае поведение не наследуется, а предоставляется правильно выбранным объектом.

На основе выбранного класса разработать диаграмму состояний, в которых может находиться объект класса, а затем соответствующую диаграмме состояний диаграмму деятельности.

Отчетность

Отчет должен содержать:

– диаграммы классов, состояний и деятельности;
– выводы.

Инструменты для создания диаграмм

1. https://online.visual-paradigm.com/diagrams/templates/class-diagram/
2. https://online.visual-paradigm.com/diagrams/templates/state-machine-diagram/
3. https://online.visual-paradigm.com/diagrams/templates/activity-diagram/