Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Tehnologii CASE”

Аттестация

Вариант 1

Проверил: Гладей Анатолий

Выполнил: Маруневич Николай

Кишинев 2022

1. **Дайте определение концепции CASE-системы. Типовые компоненты  
   CASE-систем. (15 p.)**

CASE (Computer Aided Software Engineering) – набор инструментов и методов [программной инженерии](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F) для [проектирования программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F).

CASE состоит из 4 основных компонентов:

* Средства централизованного хранения всей информации о  
  проекте (репозиторий)
* Средства ввода
* Средства анализа и разработки
* Средства вывода

1. **Что такое визуальное моделирование? (10 p.)**

Визуальное моделирование - применение для фиксирования эскизов программного обеспечения нотаций с развитой семантикой, графикой и текстовым содержанием.

1. **Какова внутренняя организация CASE-систем? (5 p.)**

Внутренняя организация, представляет собой взаимосвязи образующих ее компонентов, объединенных в две большие группы: опорную технологию и базу знаний. Предметная область, Функциональные процессы, Модели предметной области, Информационная технология, Опорная технология, База знаний, Аппаратные, Системные и База, Пользовательский средства инструм. данных интерфейс средства Информационно-технологические процессы

|  |  |  |  |
| --- | --- | --- | --- |
| Предметная область | | | |
| Функциональные процессы | | | |
| Модели предметной области | | | |
| Информационная технология | | | |
| Опорная технология | | База знаний | |
| Аппаратные средства | Системные и инструм. средства | База данных | Пользовательский интерфейс |
| Информационно-технологические процессы | | | |

1. **Методологии предшественников UML: сети Петри, диаграммы Харела, DFD, метод Буча**

Методологии предшественников UML:

1. Сеть Петри - это двудольный граф, состоящий из вершин типов “место” (или “позиция”) (place) и “переход” (transition). Позиция может содержать токены исполнения, переход забирает токены из состояний, с которыми он связан входящими связями, и добавляет токены в позиции, с которыми он связан исходящими связями (при этом важно именно не количество токенов, а связи, то есть переход с одной входящей связью и тремя исходящими требует для срабатывания один токен, а после срабатывания создаёт три). Переход возможен только когда на всех его входящих связях есть токен.

Пример:

Diagram, schematic

Description automatically generated

1. Диаграммы Харела – это формализм, который позволяет их визуализировать. Формализм достаточно продвинутый, который позволяет не просто рисовать состояния и переходы, но и определять параллельные машины состояний (подсистемы, работающие практически независимо друг от друга), определять вложенные состояния, состояния с историей и делать много других полезных штук. Пример диаграммы состояний из оригинальной работы приведён на рисунке – модель наручных электронных часов с будильником. Здесь видим два состояния, в которых могут находиться часы – живые и мёртвые, и стрелки, обозначающие переход из одного состояния в другое в зависимости от статуса батарейки. Над стрелкой пишется условие, при выполнении которого выполняется переход по этой стрелке. Состояние “живой” разделено на

несколько работающих параллельно подсистем – дисплей, будильники, звонок, подсветка дисплея. Они, в свою очередь, состоят из машин состояний, которые могут иметь сложные состояния, и т.д. Такие диаграммы оказываются чрезвычайно удобны при спецификации работы каких-либо реактивных устройств (которые имеют какие-то состояния и могут обрабатывать какие-то события, причём по-разному, в зависимости от состояния, в котором они сейчас находятся). Часы, микроволновка, сетевой сокет, лексический анализатор – примеры таких штук, которые очень удобно представлять в виде автоматов. При этом такие диаграммы обладают довольно простой и чётко специфицированной семантикой, что делает возможным их непосредственное исполнение или генерацию кода по ним.

Diagram

Description automatically generated

1. Метод включает в себя шесть типов диаграмм – классов, состояний и переходов, объектов, взаимодействий, модулей и процессов. Диаграммы состояний в методе Буча позаимствованы из statechart-ов Харела, как и в UML.

Diagram

Description automatically generated

Классы рисуются облачком с пунктирной границей, объекты – с границей обычной линией. Классы могут иметь модификаторы (абстрактный, статический, виртуальный), которые рисуются соответствующей буквой в перевёрнутом треугольничке у облака. Классы могут быть шаблонными (формальные параметрытипы рисуются в прямоугольнике с пунктирной границей, фактические параметрытипы инстанциированного шаблона - в прямоугольнике с обычной границей). Бывают категории классов (модули), рисуются квадратиками с именем категории.

1. **Для чего используется UML. Какие еще подобные языки существуют?**

**(10 p.)**

Unified Modeling Language (UML) — унифицированный язык моделирования. Расшифруем: modeling подразумевает создание модели, описывающей объект. Unified (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. UML описывает объект в едином заданном синтаксисе, поэтому где бы вы не нарисовали диаграмму, ее правила будут понятны для всех, кто знаком с этим графическим языком — даже в другой стране.

Одна из задач UML — служить средством коммуникации внутри команды и при общении с заказчиком. Давайте рассмотрим возможные варианты использования диаграмм.

Проектирование. UML-диаграммы помогут при [моделировании архитектуры больших проектов](https://evergreens.com.ua/ru/development-services/software-architecture-design.html), в которой можно собрать как крупные, так и более мелкие детали и нарисовать каркас (схему) приложения. По немувпоследствии будет строиться код.

Реверс-инжиниринг — создание UML-модели из существующего кода приложения, обратное построение. Может применяться, например, на проектах поддержки, где есть написанный код, но [документация](https://evergreens.com.ua/ru/development-services/srs-development.html) неполная или отсутствует.

Из моделей можно извлекать текстовую информацию и генерировать относительно удобочитаемые тексты — документировать. Текст и графика будут дополнять друг друга.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы. На общем уровне графическая нотация UML содержит 4 основных типа элементов:

* фигуры;
* линии;
* значки;
* надписи.

UML-нотация является де-факто отраслевым стандартом в области разработки программного обеспечения, ИТ-инфраструктуры и бизнес-систем.

1. **Диаграммы UML: диаграммы классов, объектов, пакетов, компонентов**

**И т.д. (20 p.)**

Часто используемые программы для создания диаграмм

* [Diagrams.net](https://www.diagrams.net/) — удобный сервис для создания блок-схем, UML-диаграмм, моделей бизнес-процессов онлайн. Совместим с большинством популярных инструментов, включая Google Docs, Git, Dropbox, OneDrive и другие.
* [Dbdiagram.io](https://dbdiagram.io/home) — приложение для построения диаграмм связей для баз данных. Хороший инструмент для разработчиков и аналитиков.
* [Google Drawings](https://docs.google.com/drawings) — бесплатный инструмент для создания блок-схем и диаграмм в составе Google Drive (менее удобный по сравнению с diagrams.net);
* [xmind.net](https://www.xmind.net/) — программа для построения интеллектуальных карт (mind map), логических схем, сложных структур, проведения брейнсторма и не только.

Виды UML-диаграмм

В языке UML есть 12 типов диаграмм:

* 4 типа диаграмм представляют статическую структуру приложения;
* 5 типов представляют поведенческие аспекты системы;
* 3 представляют физические аспекты функционирования системы (диаграммы реализации).

Некоторые из видов диаграмм специфичны для определенной системы и приложения. Самыми доступными из них являются:

* Диаграмма прецедентов (Use-case diagram);
* Диаграмма классов (Class diagram);
* Диаграмма активностей (Activity diagram);
* Диаграмма последовательности (Sequence diagram);
* Диаграмма развёртывания (Deployment diagram);
* Диаграмма сотрудничества (Collaboration diagram);
* Диаграмма объектов (Object diagram);
* Диаграмма состояний (Statechart diagram).

Диаграмма прецедентов — Use-case diagram

Диаграмма прецедентов использует 2 основных элемента:

1) Actor (участник) — множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек, роль человека в системе или другая система, подсистема или класс, которые представляют нечто вне сущности.

2) Use case (прецедент) — описание отдельного аспекта поведения системы с точки зрения пользователя. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.

Рассмотрим классический студенческий пример, в котором есть 2 участника: студент и библиотекарь. Прецеденты для студента: ищет в каталоге, заказывает, работает в читальном зале. Роль библиотекаря: выдача заказа, консультации (рекомендации книг по теме, обучение использованию поисковой системы и заполнению бланков заказа).

Второй пример немного сложнее. Видим, что одно и то же лицо может выступать в нескольких ролях. Например, product manager у нас работает над стратегией и больше ничем не занимается, архитектор работает над стратегией и занимается внедрением, build master занимается тремя вещами одновременно, и так далее. По такой схеме мы можем проследить, какая из ролей связана с какими прецедентами.

1. **Охарактеризуйте структуру визуальных языков и их свойства (5 p.)**

Визуальное программирование — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста. Некоторые авторы представляют визуальное программирование как следующий этап развития языков программирования (следующее поколение). В настоящее время визуальному программированию стали уделять больше внимания, в том числе в связи с развитием мобильных сенсорных устройств (КПК, планшеты). Визуальное программирование может применяться для создания программ с графическим интерфейсом, но не только. Существуют, например, и успешно используются в промышленности, средства графического программирования встроенных приложений для микроконтроллеров. Есть и среды визуального программирования, позволяющие создавать Веб-приложения для браузеров.

* Визуальные языки программирования:  
  [App Inventor](https://ru.wikipedia.org/wiki/App_Inventor) — Cреда визуальной разработки android-приложений, требующая от пользователя минимальных знаний программирования.
* Sketchware — Среда визуальной разработки приложений для Android.
* [Дракон](https://ru.wikipedia.org/wiki/%D0%94%D0%A0%D0%90%D0%9A%D0%9E%D0%9D_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))  — графический язык программирования, имеющий корни в программировании ракетно-космической техники («[Буран](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%80%D0%B0%D0%BD_(%D0%BA%D0%BE%D1%81%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D1%80%D0%B0%D0%B1%D0%BB%D1%8C))», «[Морской старт](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%80%D1%81%D0%BA%D0%BE%D0%B9_%D1%81%D1%82%D0%B0%D1%80%D1%82) »). Существуют Дракон-редакторы, включая бесплатные.
* Язык последовательных функциональных схем [SFC](https://ru.wikipedia.org/wiki/SFC) (Sequential Function Chart) — [графический язык программирования](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) широко используется для программирования промышленных логических контроллеров [PLC](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80).
* [HiAsm](https://ru.wikipedia.org/wiki/HiAsm) — это язык и среда разработки приложений, которая позволяет создавать приложения, управляя их моделью с помощью интуитивно понятного графического интерфейса [HiAsm](http://hiasm.com/).
* В SFC программа описывается в виде схематической последовательности шагов, объединённых переходами.
* [LD](https://ru.wikipedia.org/wiki/Ladder_Diagram) — язык релейно-контактных схем.
* [FBD](https://ru.wikipedia.org/wiki/FBD) — язык Функциональных блоковых диаграмм.
* Язык CFC ([Continuous Flow Chart](https://ru.wikipedia.org/w/index.php?title=Continuous_Flow_Chart&action=edit&redlink=1" \o "Continuous Flow Chart (страница отсутствует))) — ещё один высокоуровневый язык графического программирования. CFC — это дальнейшее развития языка FBD. CFC был специально создан для проектирования систем управления непрерывными технологическими процессами.
* Язык «G» системы [LabVIEW](https://ru.wikipedia.org/wiki/LabVIEW" \o "LabVIEW) — один из самых распространенных языков разработки программ, работающих с некомпьютерным оборудованием.
* [VisSim](https://ru.wikipedia.org/wiki/VisSim) — это визуальный язык программирования предназначенный для моделирования [динамических систем](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), а также проектирования, базирующегося на моделях, для встроенных [микропроцессоров](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80).
* [Блокли](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA%D0%BB%D0%B8) — это библиотека для создания среды визуального программирования, которая может быть встроена в произвольное веб-приложение.
* [Кибор](http://kibor-bot.com/stat/visual-programming.php) — Интегрированная среда создания бот программ автоматизации. Обладает визуальным инструментом для построения программ

1. Перечислите некоторые известные вам CASE-системы и их основные  
   особенности. (10р.)  
     
   Примеры комплексов CASE-средств обеспечивающих поддержку полного ЖЦ ПО. Здесь хотелось бы еще раз отметить нецелесообразность сравнения отдельно взятых CASE-средств, поскольку ни одно из них не решает в целом все проблемы создания и сопровождения ПО. Это подтверждается также полным набором критериев оценки и выбора, которые затрагивают все этапы ЖЦ ПО. Сравниваться могут комплексы методологически и технологически согласованных инструментальных средств, поддерживающие полный ЖЦ ПО и обеспеченные необходимой технической и методической поддержкой со стороны фирм-поставщиков. По мнению автора, на сегодняшний день наиболее развитым из всех поставляемых в России комплексов такого рода является комплекс технологий и инструментальных средств создания ИС, основанный на методологии и технологии DATARUN. В состав комплекса входят следующие инструментальные средства:

* CASE-средство Silverrun;
* средство разработки приложений JAM;
* мост Silverrun-RDM JAM;
* комплекс средств тестирования QA;
* менеджер транзакций Tuxedo;
* комплекс средств планирования и управления проектом SE Companion;
* комплекс средств конфигурационного управления PVCS;
* объектно-ориентированное CASE-средство Rational Rose;
* средство документирования SoDA.

Примерами других подобных комплексов являются:

* Vantage Team Builder for Uniface + Uniface (фирмы "DataX/Florin" и "ЛАНИТ");
* комплекс средств, поставляемых и используемых фирмой "ФОРС":
* CASE-средства Designer/2000 (основное), ERwin, Bpwin и Oowin (альтернатив-  
  ные);
* средства разработки приложений Developer/2000, ORACLE Power Objects (ос-  
  новные) и Usoft Developer (альтернативное);
* средство настройки и оптимизации ExplainSQL (Platinum);
* cредства администрирования и сопровождения SQLWatch, DBVision, SQL Spy, TSReorg и др. (Platinum);
* средство документирования ORACLE Book.
* комплекс средств на основе продуктов фирмы CENTURA:
* CASE-средства ERwin, Bpwin и Oowin (объектно-ориентированный анализ);
* средства разработки приложений SQLWindows и TeamWindows;
* средство тестирования и оптимизации приложений "клиент-сервер" SQLBench (ARC);
* cредства эксплуатации и сопровождения Quest и Crystal Reports.