Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Tehnologii CASE”

Аттестация

Вариант 1

Преподаватель: Гладей Анатолий

Студент: 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.

**3. Какова внутренняя организация CASE-систем?**

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 | | | |

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

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 нему впоследствии 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.

**6. Диаграммы UML: диаграммы классов, объектов, пакетов, компонентов и т.д.**

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.