Раздел 3. Объектно-ориентированное проектирование

Тема 7. Принципы объектно-ориентированного проектирования

1. Представление вариантов использования
2. Проектирование структуры системы
3. Проектирование взаимодействий

Краткое описание разрабатываемой системы

Автоматизированная система записи на курсы дистанционного обучения

1. Методист составляет и занимается поддержкой учебного плана

2. Студент должен выбрать 4 обязательных и 2 курса по выбору

3. Как только студент зарегистрировался на курс, система оплаты должна предоставить счёт на оплату

4. Студент должен иметь возможность добавить/удалить курс из своего плана в течение определённого времени после регистрации

5. Профессора используют систему для предоставления информации о предстоящем курсе, а также для получения списка студентов, зарегистрированных курс

6. Один и тот же курс может читаться несколько раз, в разное время, в разных местах

7. Все пользователи системы получают к ней доступ, используя логин/пароль

Выделяем актёров

1. Методист составляет учебный план используя алгоритм расписания

2. Студент должен выбрать 4 обязательных и 2 курса по выбору, заполнив форму регистрации

3. Как только студент зарегистрировался на курс, система оплаты должна предоставить счёт на оплату

4. Студент должен иметь возможность добавить/удалить курс из своего плана в течение определённого времени после регистрации

5. Профессора используют систему для предоставления информации о предстоящем курсе, а также для получения списка студентов, зарегистрированных курс

6. Один и тот же курс может читаться несколько раз, в разное время, в разных местах

7. Все пользователи системы получают к ней доступ, используя логин/пароль

Структурные диаграммы: Выделяем классы

1. Методист составляет учебный план используя алгоритм расписания

2. Студент должен выбрать 4 обязательных и 2 курса по выбору, заполнив форму регистрации

3. Как только студент зарегистрировался на курс, системы оплаты должна предоставить счёт на оплату

4. Студент должен иметь возможность добавить/удалить курс из своего плана в течение определённого времени после регистрации

5. Профессора используют систему для предоставления информации о предстоящем курсе, а также для получения списка студентов, зарегистрированных курс

6. Один и тот же курс может читаться несколько раз, в разное время, в разных местах

7. Все пользователи системы получают к ней доступ, используя логин/пароль

Структура класса: атрибуты и операции

- ☐ определение класса
- ☐ изучение требований
- ☐ анализ предметной области

Отношения между классами

- [ ] зависимость - x использует y
- [ ] ассоциация / агрегация - x содержит y
- [ ] обобщение - x является y

Примеры отношений

- ☐ RegistrationManager использует SchedulingAlgorithm
- ☐ RegistrationManager использует Student
- ☐ Student использует RegistrationManager
- ☐ Student регистрируется на CourseOffering
- ☐ Student содержит CourseOffering
- ☐ Course состоит из CourseOffering
- ☐ CourseOffering является Course
- ☐ Student является RegistrationUser
- ☐ Professor является RegistrationUser

Путь к хорошей диаграмме классов

- [ ] понимание проблемы
- [ ] хорошие названия классов
- [ ] концентрация на ЧТО, а не КАК
- [ ] итеративность

Диаграмма компонентов

Component Diagram

- **Вершины - Узел (node) - компонент (набор классов с хорошо определённым интерфейсом)**
- **Рёбра - Грань (edge) - отношение “использует”**

Пример диаграммы компонентов

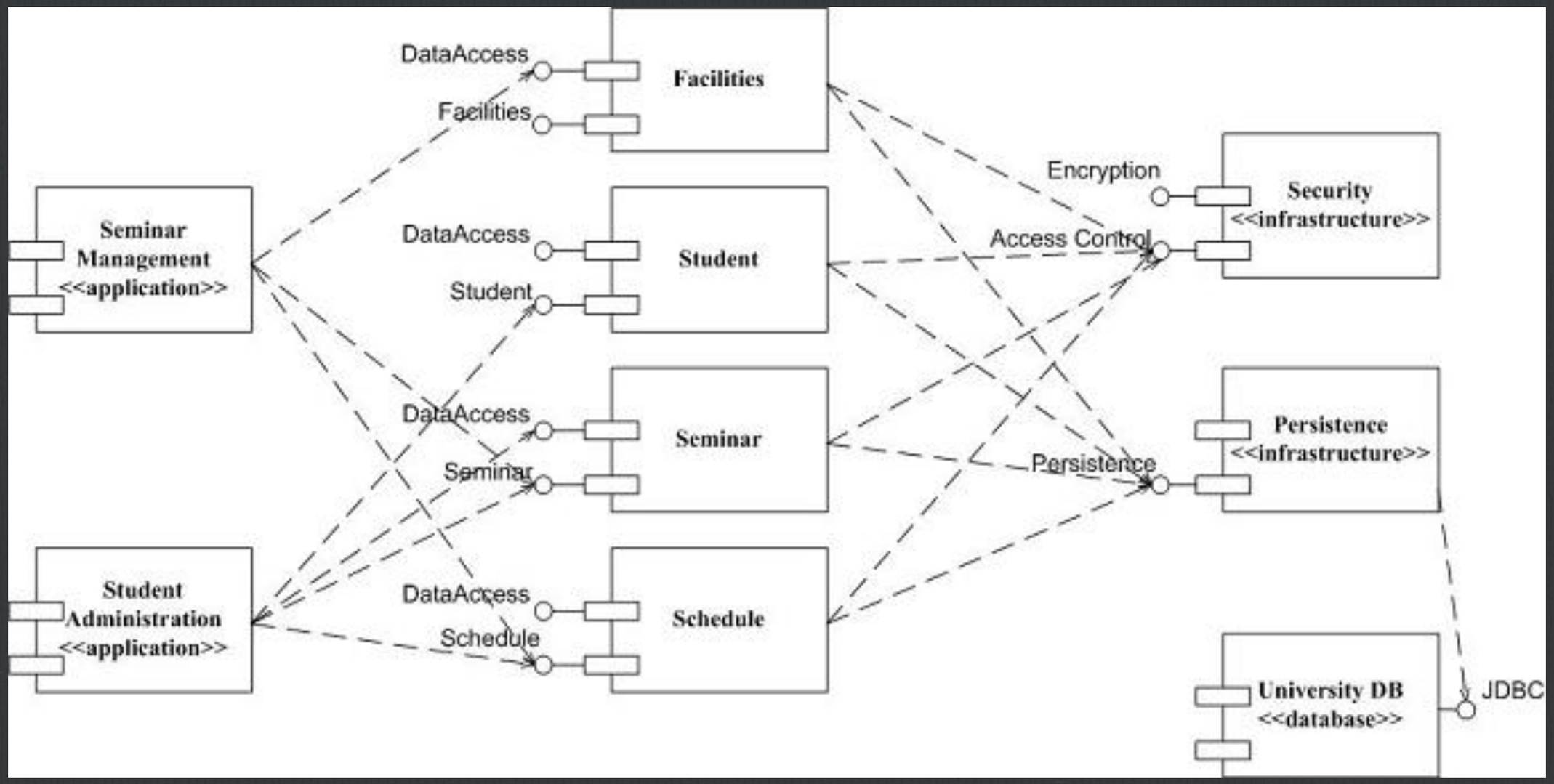
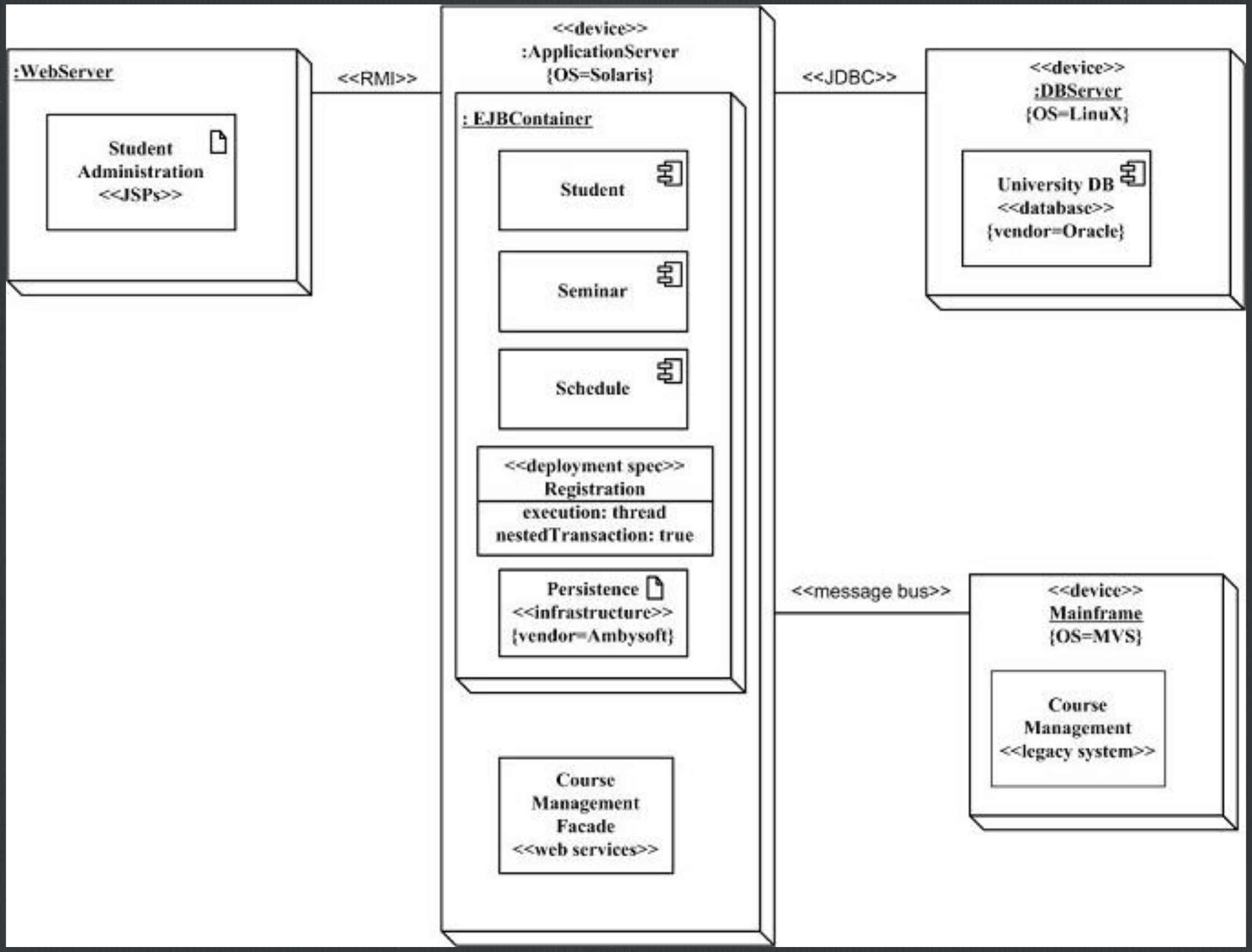


Диаграмма развёртывания

Deployment Diagram

- Узел - вычислительный узел
- Грань - физическая связь


:WebServer
Student Administration
<<JSPs>>
<<RMI>>
<<device>>
:ApplicationServer
{OS=Solaris}
: EJBContainer
Student
Seminar
Schedule
<<deployment spec>>
Registration
execution: thread
nestedTransaction: true
Persistence
<<infrastructure>>
{vendor=Ambysoft}
Course Management Facade
<<web services>>
<<JDBC>>
<<device>>
:DBServer
{OS=LinuX}
University DB
<<database>>
{vendor=Oracle}
<<message bus>>
<<device>>
Mainframe
{OS=MVS}
Course Management
<<legacy system>>

Диаграммы поведения: Поток событий

Вариант использования Register for Course

1. **Студент заходит в систему используя свои логин/пароль**
2. **Если пароль валиден, студент может заполнить форму регистрации**
3. **Если форма заполнена верно, информация отправляется менеджеру**
4. **Менеджер регистрации проверяет открыта ли регистрация на курс**
5. **Если регистрация открыта, студент добавляется в список зарегистрированных**

Диаграмма последовательности

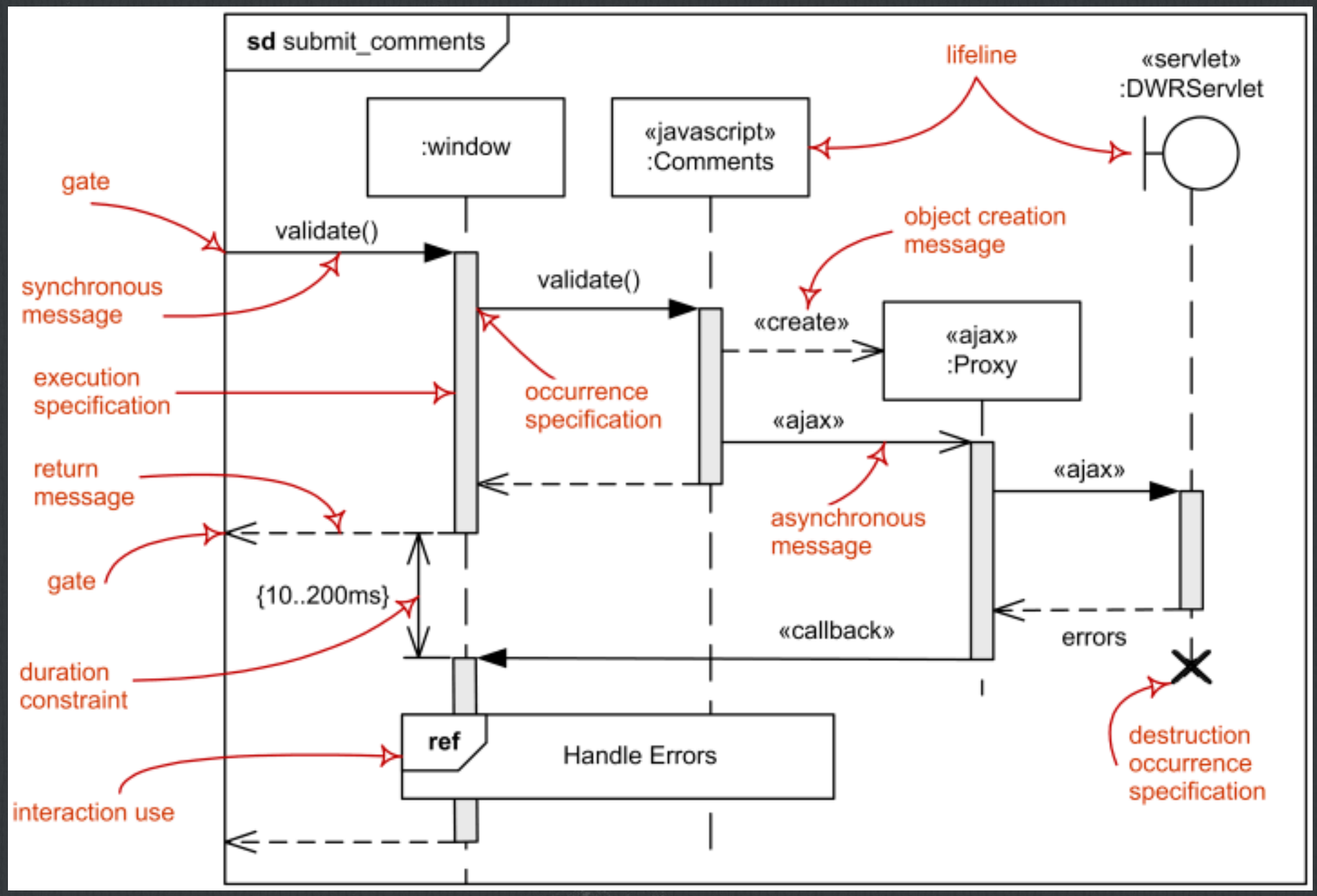
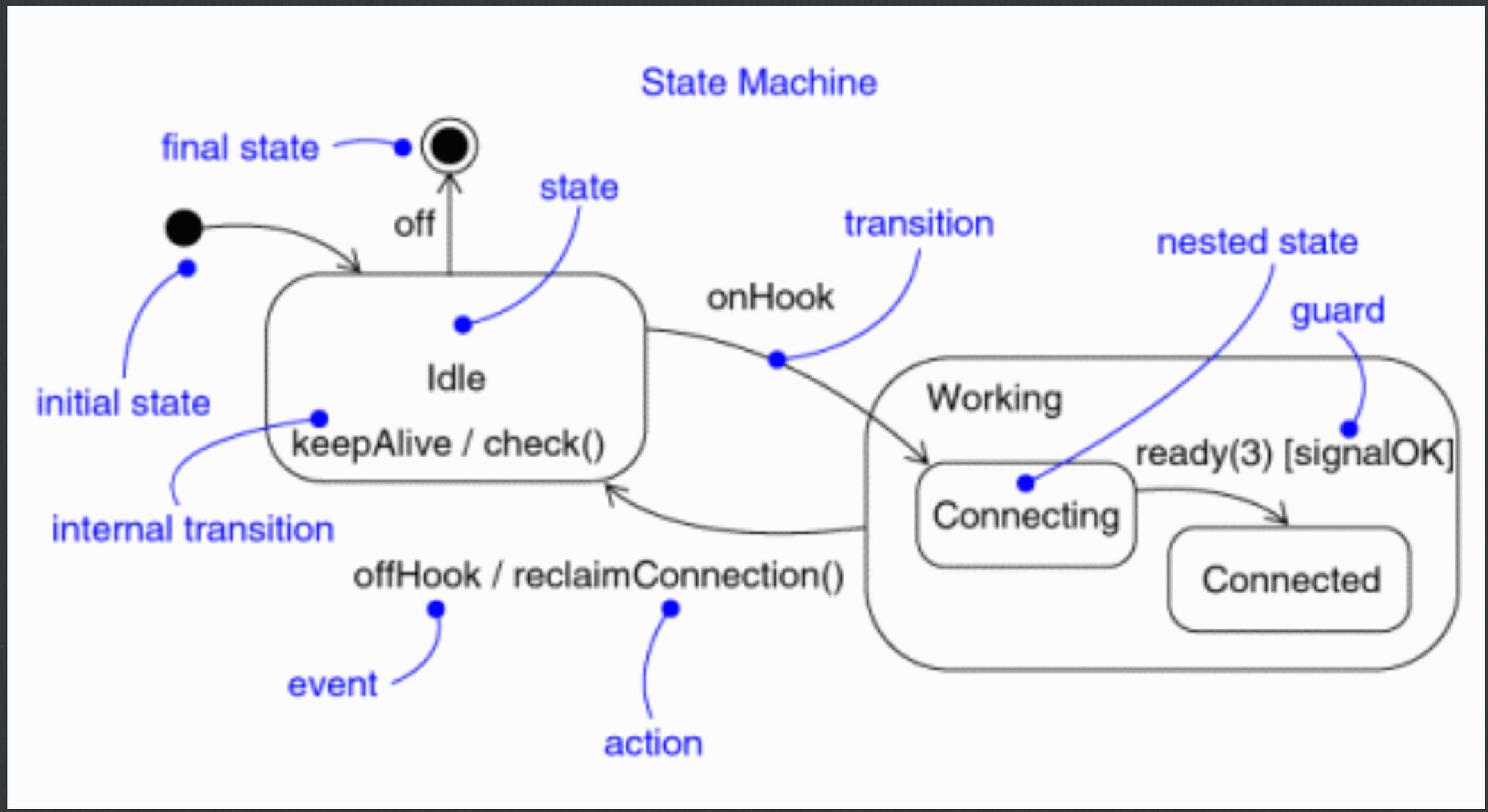


Диаграмма состояний



Замечания по выполнению работ 3 и 4

- ☐ При построении диаграмм нужно использовать нотацию UML
- ☐ Диаграммы Вариантов Использования должны быть понятными
- ☐ Диаграммы активностей должны соответствовать потоку событий
- ☐ Диаграммы состояний могут использовать мокапы и не должны содержать равнозначных безусловных переходов

Замечания по выполнению работ 3 и 4

- [ ] Диаграммы последовательности должны показывать объекты системы, а не саму систему, слишком большие куски системы (GUI, БД) или элементы фреймворка

- [ ] Диаграмма классов должна показывать классы спроектированных выше объектов, с необходимой детализацией

- [ ] Диаграмму компонентов можно объединять с диаграммой развёртывания (для “небольших” систем)

- [ ] Диаграммы должны быть легко доступны в вашем репозитории