Концепция Семантического Web
&
Интеллектуальные агенты

Пантелеев М.Г.

Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”

Центр новых информационных технологий

Лаборатория Интеллектуальных технологий (“ИНТЕЛТЕХ”)

M_Pantel@eltech.ru

MGPanteleev@mail.eltech.ru

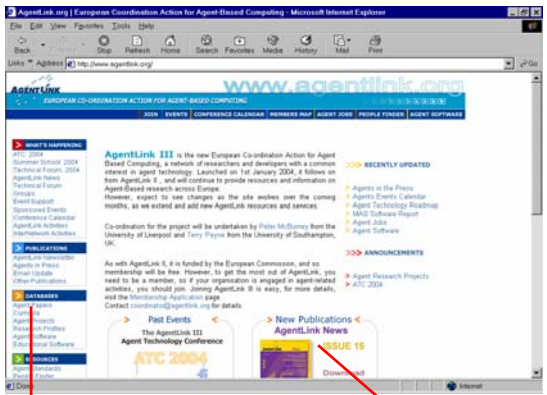
11.2004

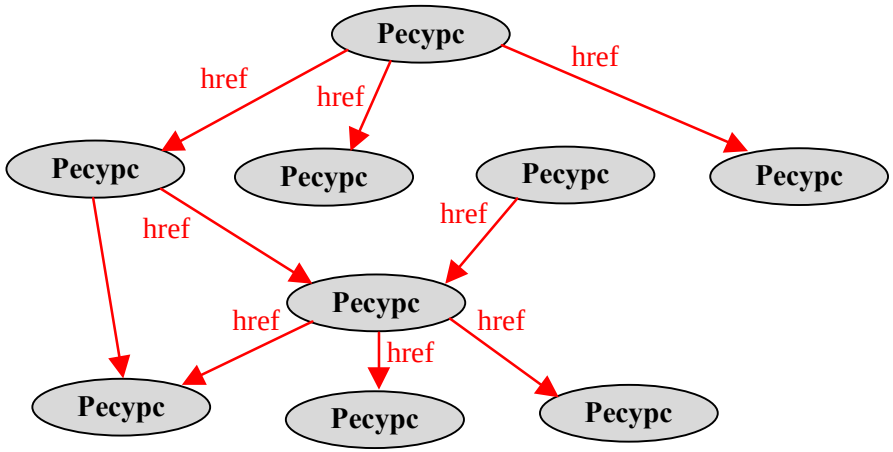
Содержание

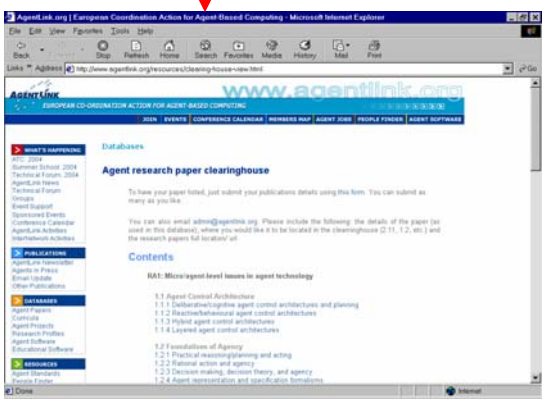
- Недостатки традиционного (синтаксического) Web
- Что такое Семантический Web
- Язык описания ресурсов RDF
- Язык RDFS (RDF-схема)
- Язык онтологий OWL
- Определение интеллектуального агента (ИА)
- Архитектуры ИА
- Стандартизация в области ИА
- Агентные платформы
- Языки и протоколы общения ИА

Традиционный Web является *синтаксическим*

- Множество HTML-документов (Web-страниц), распределенных в Сети и связанных гипертекстовыми ссылками







Традиционный (синтаксический) Web:

- **Гипермедийная цифровая библиотека:**
  - **библиотека документов (называемых web-страницами), связанных посредством гиперссылок**
- **База данных, платформа для приложений**
  - **Общая точка входа (портал) к приложениям, доступным через web-страницы и представляющим свои результаты как web-страницы**
- **Платформа для мультимедиа**
  - **Всемирная служба BBC Radio 4**
- **Схема именования**
  - **уникальные идентификаторы для документов (ресурсов)**

Глобальная информационная среда, в которой *компьютеры* (*легко*) выполняют *представление информации*, а *люди* (с *трудом*) выполняют ее *связывание* и *интерпретацию*.

Цель: *передать компьютеру больше трудной работы.*

Традиционный (синтаксический) Web не позволяет:

- Обрабатывать сложные запросы, подразумевающие базовые знания:
  - “Найти информацию о животных, использующих звуковую локацию, но не являющихся ни летучей мышью ни дельфином»”
- Находить информацию в репозитариях данных:
  - Туристические запросы;
  - Цены на товары и услуги;
  - Результаты исследований генома человека
- Находить и использовать “Web-сервисы”:
  - Визуализация поверхностей взаимодействия между двумя протеинами
- Делегировать решение сложных задач “Web-агентам”:
  - “Закажи мне на следующий выходной отдых, где-нибудь в теплом месте, не очень далеко и где говорят на английском или французском языках”

В чем проблема?

- **Разметка состоит из :**
  - **информации, относящейся к визуализации (например, размер и цвет шрифта);**
  - **гипер-ссылок на связанное содержание.**
- **Семантика содержания доступно людям, но не компьютерам**

Какую информацию мы можем видеть…

WWW2002

The eleventh international world wide web conference

Sheraton waikiki hotel

Honolulu, hawaii, USA

7-11 may 2002

1 location 5 days learn interact

Registered participants coming from

australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire

Register now

On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event …

Speakers confirmed

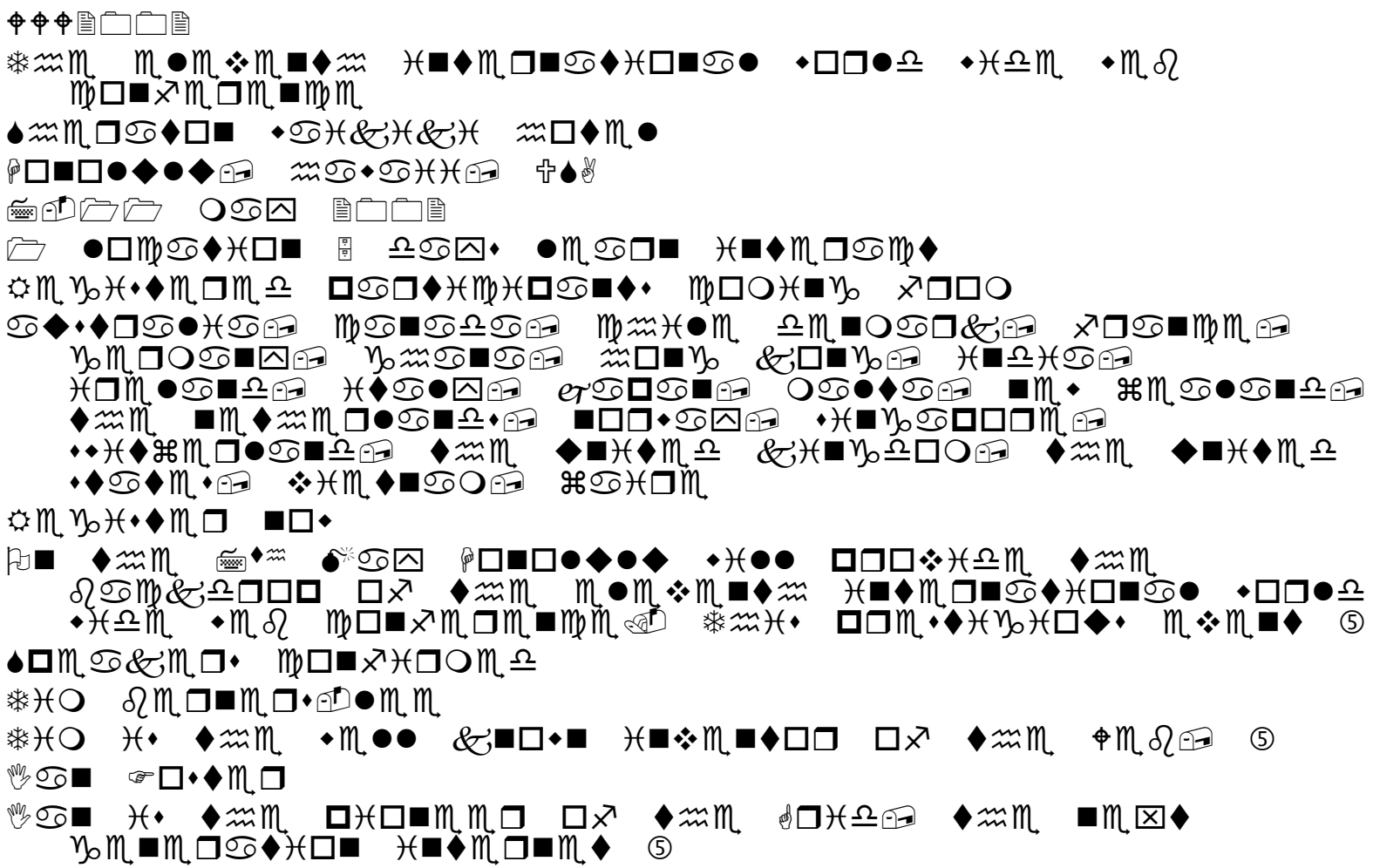
Tim berners-lee

Tim is the well known inventor of the Web, …

Ian Foster

Ian is the pioneer of the Grid, the next generation internet …

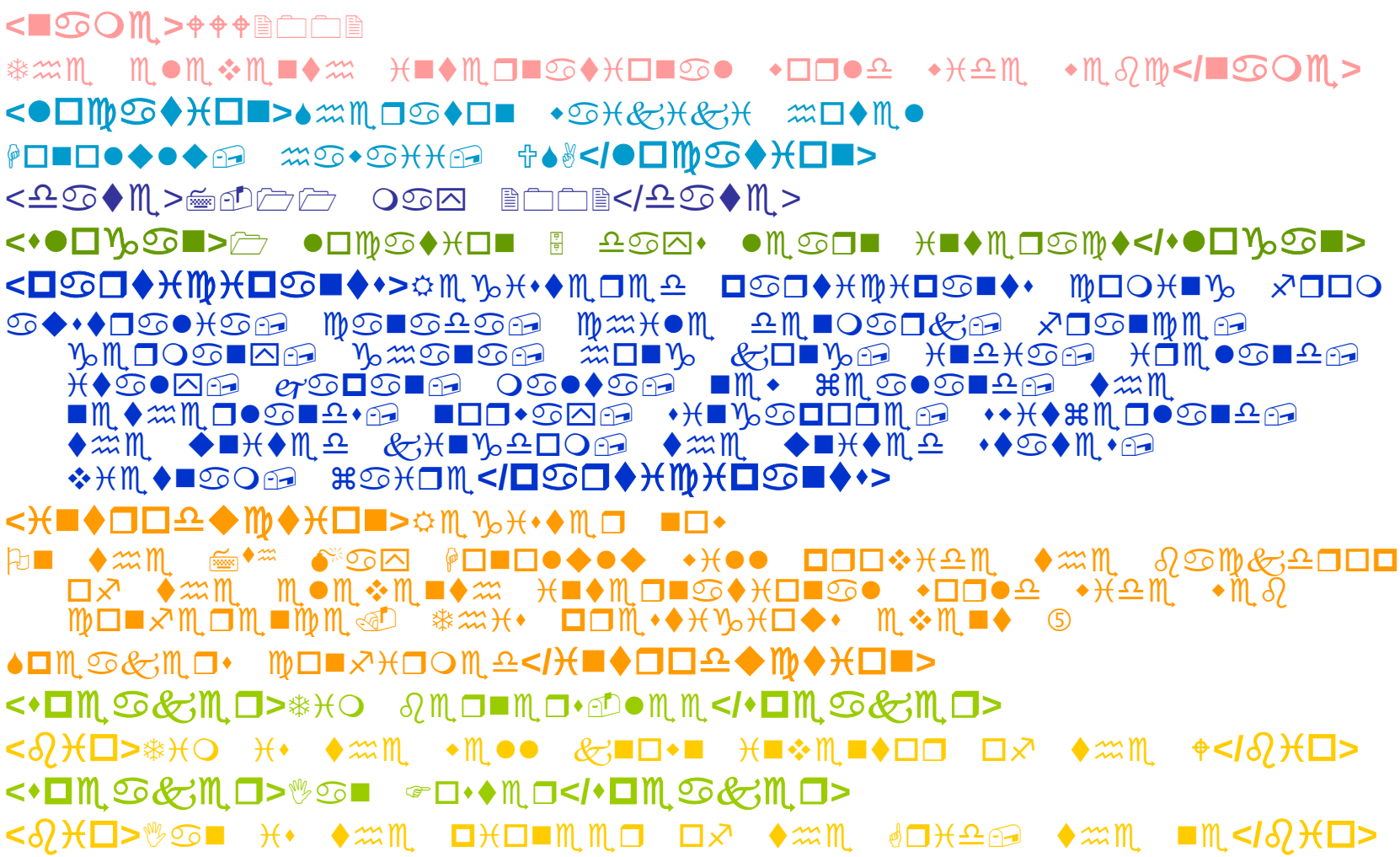
Какую информацию «видит» компьютер



Возможное решение: XML-разметка со “смысловыми” тегами

<name>♑♑♑📄🗀🗀📄
❄♒♏ ♏●♏❖♏■♦♒ ♓■♦♏❒■♋♦♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌♍□■</name>
<location>💧♒♏❒♋♦□■ ⬥♋♓&♓&♓ ♒□♦♏● 🔊□■□●◆●◆🖃 ♒♋⬥♋♓♓🖃 ☥💧✌</location>
<date>📻📬🗁🗁 ❍♋🖂 📄🗀🗀📄</date>
<slogan>🗁 ●□♍♋♦♓□■ 🎚 ♎♋🖂⬥ ●♏♋❒■ ♓■♦♏❒♋♍♦</slogan>
<participants>☼♏♑♓⬥♦♏❒♏♎ ◘♋❒♦♓♍♓◘♋■♦⬥ ♍□❍♓■♑ ↗❒□❍
♋◆⬥♦❒♋●♓♋🖃 ♍♋■♋♎♋🖃 ♍♒♓●♏ ♎♏■❍♋❒&🖃 ↗❒♋■♍♏🖃
♑♏❒❍♋■🖂🖃 ♑♒♋■♋🖃 ♒□■♑ &□■♑🖃 ♓■♎♓♋🖃 ♓❒♏●♋■♎🖃
♓♦♋●🖂🖃 ♂♋◘♋■🖃 ❍♋●♦♋🖃 ■♏⬥ ⌘♏♋●♋■♎🖃 ♦♒♏
■♏♦♒♏❒●♋■♎⬥🖃 ■□❒⬥♋🖂🖃 ⬥♓■♑♋◘□□❒♏🖃 ⬥⬥♓♦⌘♏❒●♋■♎🖃 ♦♒♏
◆■♓♦♏♎ &♓■♑♎□❍🖃 ♦♒♏ ◆■♓♦♏♎ ⬥♦♋♦♏⬥🖃 ❖♓♏♦■♋❍🖃
⌘♋♓❒♏</participants>
<introduction>☼♏♑♓⬥♦♏❒ ■□⬥
🏳■ ♦♒♏ 📻♦♒ 💣♋🖂 🔊□■□●◆●◆ ⬥♓●● ◘❒□❖♓♎♏ ♦♒♏ ♌♋♍&♎❒□◘
□↗ ♦♒♏ ♏●♏❖♏■♦♒ ♓■♦♏❒■♋♦♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌
♍□■↗♏❒♏■♍♏📬 ❄♒♓⬥ ◘❒♏⬥♦♓♑♓□◆⬥ ♏❖♏■♦ ⑤
💧□♏♋&♏❒⬥ ♍□■↗♓❒❍♏♎</introduction>
<speaker>❄♓❍ ♌♏❒■♏❒⬥🖃●♏♏</speaker>
<bio>❄♓❍ ♓⬥ ♦♒♏ ⬥♏●● &■□⬥■ ♓■❖♏■♦□❒ □↗ ♦♒♏ ♑♏♌🖃</bio>…

Машина видит:



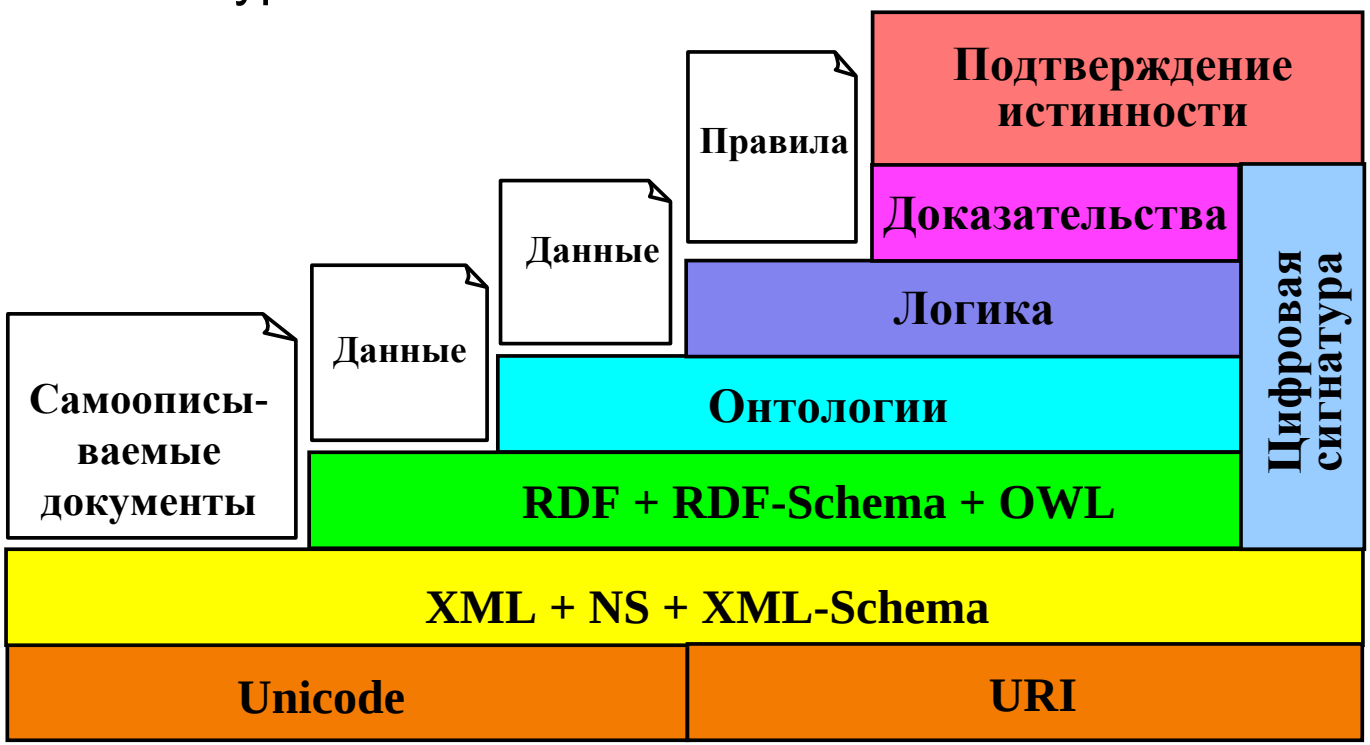
Необходимость добавления семантики

- Внешние соглашения о смысле аннотаций
  - Например, Dublin Core
    - Соглашение о смысле множества аннотирующих тегов;
  - Недостатки подхода:
    - отсутствие гибкости;
    - может быть выражено ограниченное число вещей.

- Использование Онтологий для специфицирования смысла аннотаций
  - Онтологии предоставляют словарь терминов;
  - Новые термины могут формироваться путем комбинирования существуюших;
  - Смысл (семантика) таких терминов формально специфицирован;
  - Кроме того, могут быть специфицированы отношения между терминами во многих онтологиях.

Что же такое Семантический Web?

Семантический Web - расширение существующего Web, в котором информации придается точно определенный смысл, что позволяет компьютерам «понимать» и обрабатывать ее на семантическом уровне



Многоуровневое представление Семантического Web по T.Berners-Lee

Семантический Web — Основные идеи

- Сделать web-ресурсы более доступными для автоматических процессов
- Расширить существующую разметку, предназначенную для визуализации семантической разметкой
  - **Аннотации с помощью метаданных, описывающие содержание/функции доступных через web ресурсов**
- Использование онтологий, предоставляющих словарь для аннотаций
  - **“Формальная спецификация” доступная машинам**
- Предпосылкой является стандартный язык web онтологий
  - **Необходимость согласия об общем синтаксисе, прежде чем совместно использовать семантику**
  - **(Синтаксический web основан на таких стандартах как HTTP и HTML)**

RDF

Что такое RDF

RDF (*Resource Description Framework*) - язык для *описания ресурсов способом, "понятным" компьютеру на семантическом уровне*.

Ресурс - ***любая*** (физическая или абстрактная) ***сущность***, ***имеющая уникальный идентификатор*** - **URI**:

- *доступные по сети ресурсы*: электронные документы, изображения, сервисы или группы ресурсов;
- *недоступные* непосредственно *по сети объекты*: люди, корпорации, книги в переплетах;
- не существующие в физическом мире *абстрактные понятия*: "создатель" ("creator").

RDF :

- модель + графический формализм +
- XML-синтаксис +
- семантика для представления метаданных

RDF - официальная рекомендация консорциума W3C с **10.02.2004** (http://www.w3.org/RDF)

RDF: Модель данных

- ***Утверждения***, задаются *тройками:*
  < *субъект*, *предикат*, *объект* >

- *Субъект* - некоторый ресурс (идентификатор ресурса):
- *Предикат* - свойство ресурса (или отношение с другим ресурсом)
- *Объект* - значение свойства ресурса (или отношения)

Пример:
“Ora Lassila is the *creator* of the resource http://www.w3.org/Home/Lassila”

Субъект **(ресурс) - http://www.w3.org/Home/Lassila**
Предикат **(свойство) - *creator***
Объект **- (значение) - Ora Lassila**

RDF: Графическое представление

- Свойства также являются ресурсами и имеют URI;
- Имена свойств берутся из пространств имен, которые указываются в виде префикса.

Если свойство Creator взято из пространства имен dc: (Dublin Core):

XML-синтаксис для RDF (RDF/XML)

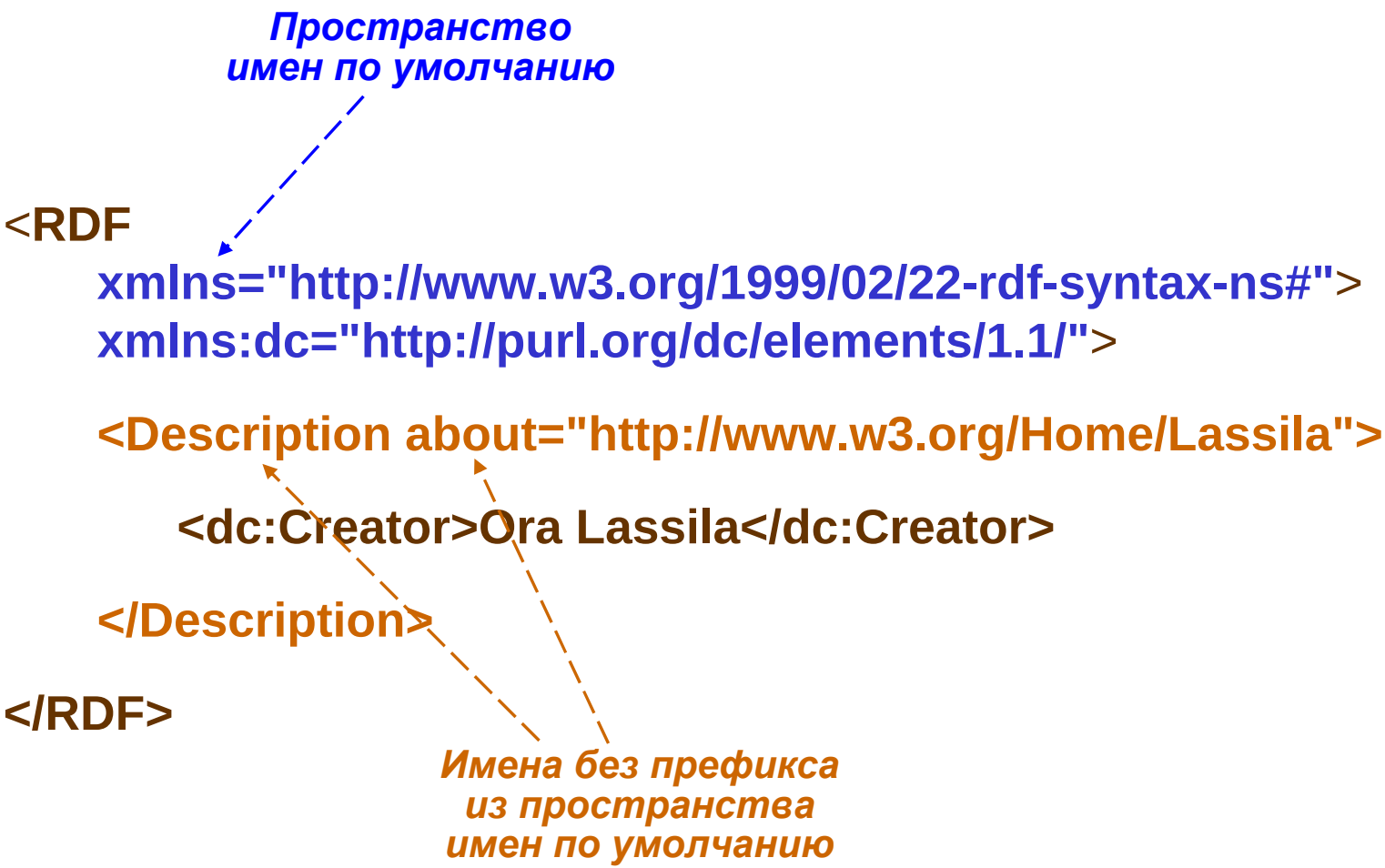
- *RDF-модель представляется XML-элементом* RDF, который заключается в тег <rdf:RDF>).
- Элемент RDF должен содержать *объявление пространства имен* RDF:

```
<?xml version="1.0”?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      /* RDF-модель */
</rdf:RDF>
```

- В *RDF-модели*, как правило, *описывается* сразу *несколько* ресурсов и у каждого ресурса *описывается несколько свойств* (т.е делается несколько утверждений);
- Множество утверждений, относящихся к одному ресурсу, группируются с помощью элемента Description из пространства имен rdf.
- Идентификатор описываемого ресурса (URI) указывается в атрибуте about элемента Description.

RDF/XML: Простой пример

RDF/XML: Пространство имен по умолчанию



RDF/XML: свойство как атрибут

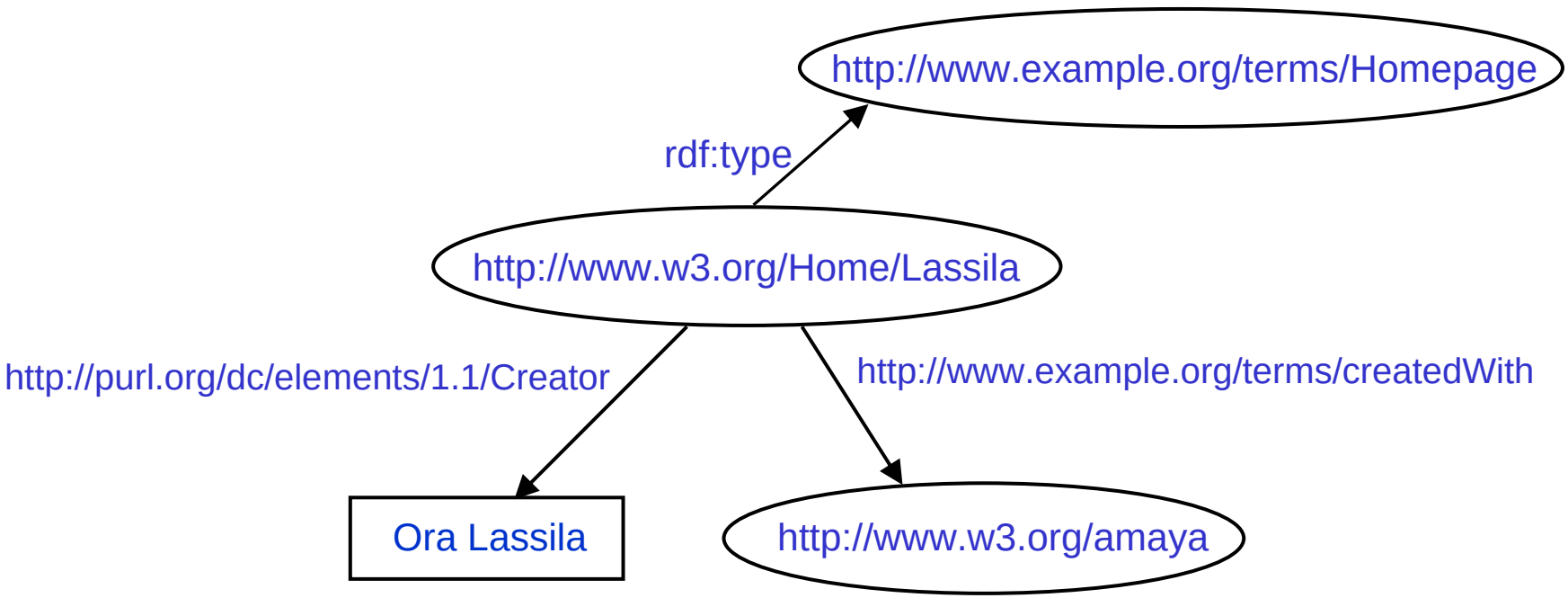
- Свойства могут записываться как атрибуты XML-элемента Description:

```
<rdf:RDF
    <rdf:Description rdf:about="http://www.w3.org/Home/Lassila">
                    dc:Creator="Ora Lassila"/>
 </rdf:RDF>
```

- Т. к. XML- элемент **Description** в данном случае не имеет другого содержания, кроме записанного в форме атрибута свойства **Creator**, то используется синтаксис пустого XML-элемента XML (бег концевого тега)

RDF-модель: значение свойства - ресурс

- Объект утверждения (значение свойства) может не только *литералом* , но и *ресурсом*. В последнем случае он, в свою очередь, может быть субъектом другого утверждения.
- Совокупности утверждений формируют направленный помеченный граф: *ресурсы* представляются овалами, *литералы* - прямоугольниками.



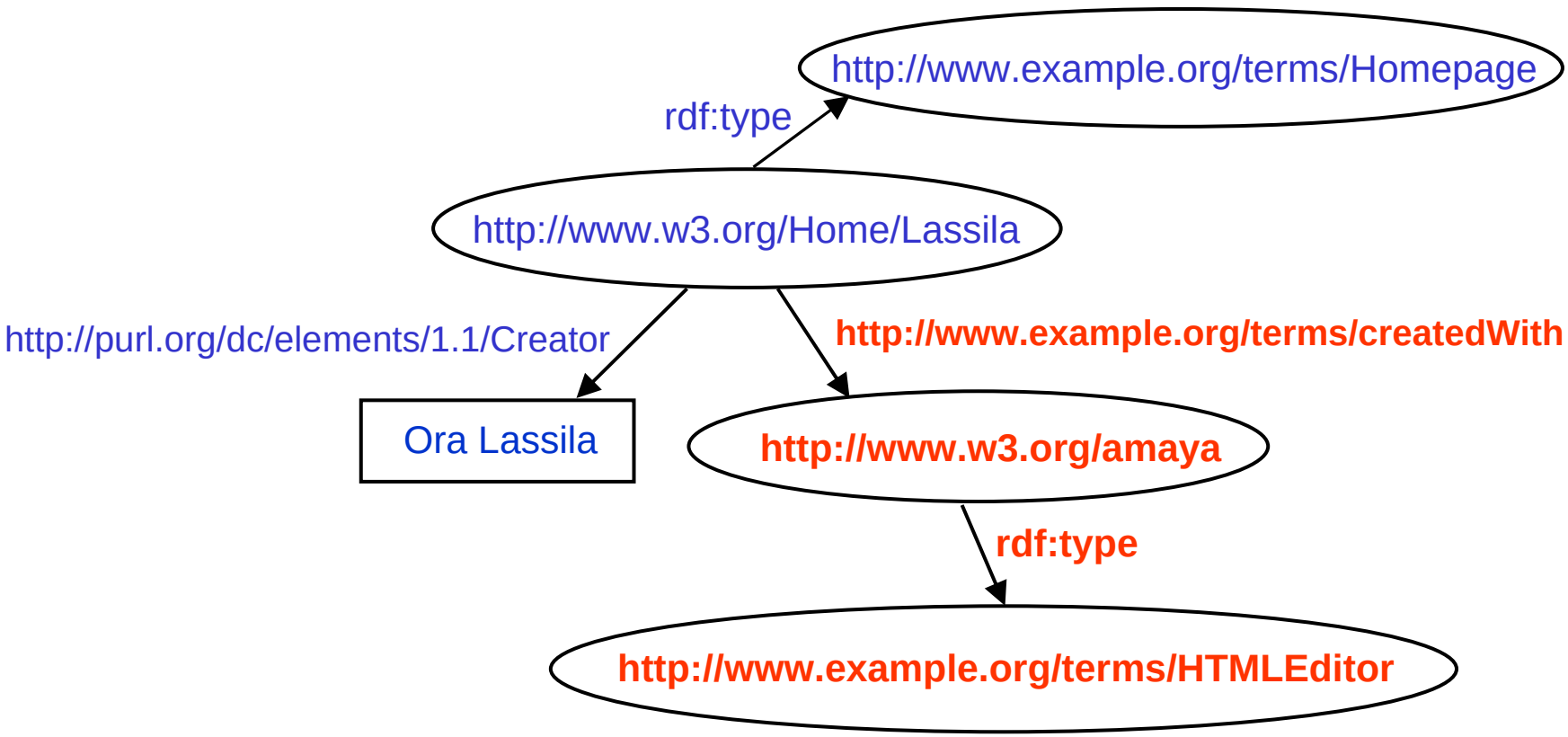
RDF/XML: значение свойства - ресурс

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#»
    xmlns:dc="http://purl.org/dc/elements/1.1/»
    xmlns:ex="http://www.example.org/terms/»>

    <rdf:Description rdf:about="http://www.w3.org/Home/Lassila">
      <rdf:type rdf:resource="http://www.example.org/terms/Homepage"/>
      <dc:Creator>Ora Lassila</dc:Creator>
      <ex:createdWith rdf:resource="http://www.w3.org/amaya"/>
    </rdf:Description>
</rdf:RDF>
```

RDF-модель: значение - ресурс со свойствами

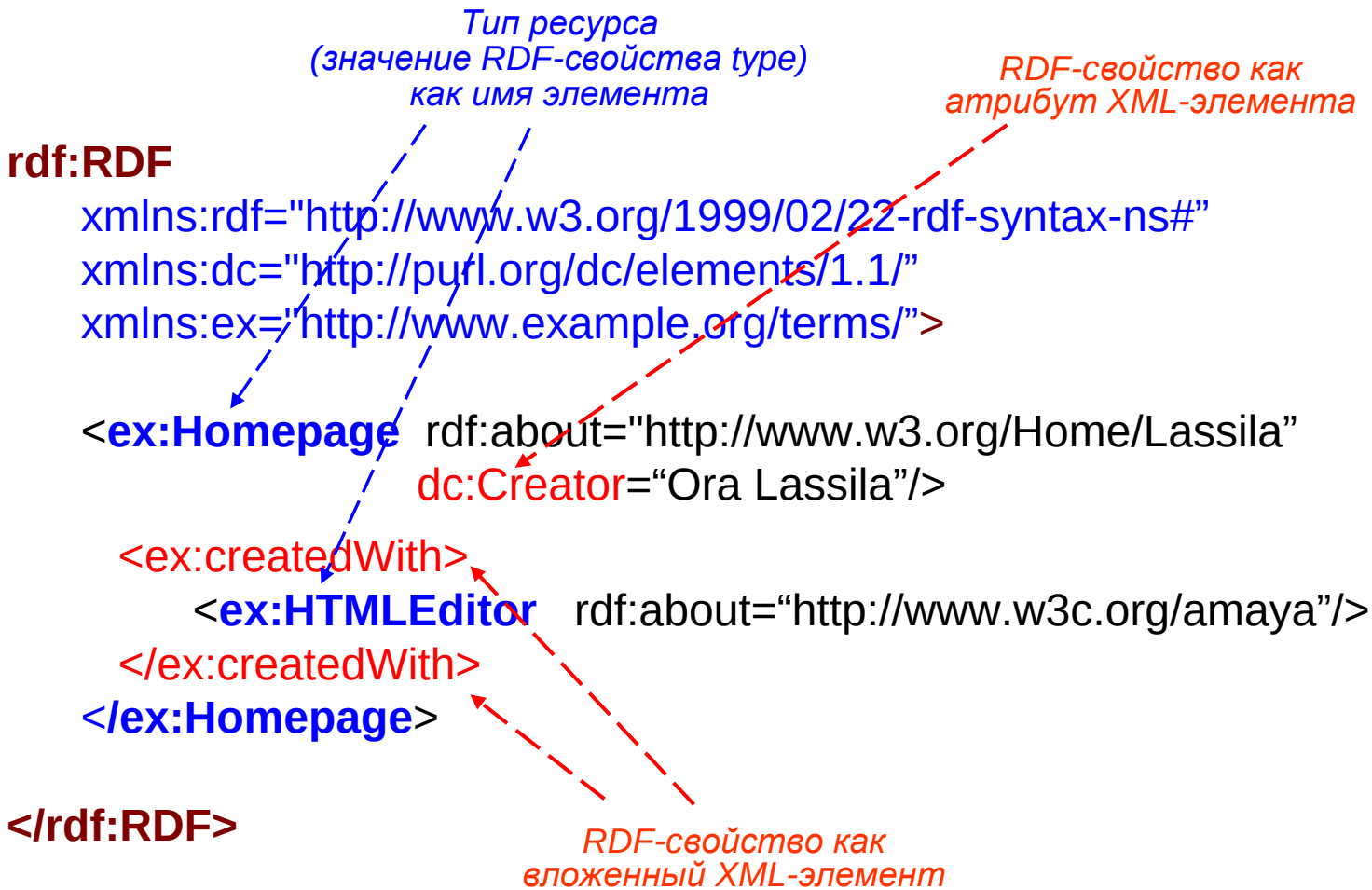
1. Ресурс http://www.w3.org/Home/Lassila является домашней страницей
2. Ресурс http://www.w3.org/Home/Lassila имеет создателя Ора Лассила.
3. Ресурс http://www.w3.org/Home/Lassila создан с помощью http://www.w3.org/amaya.
4. Ресурс http://www.w3.org/amaya является HTML-редактором.



RDF/XML: значение - ресурс со свойствами

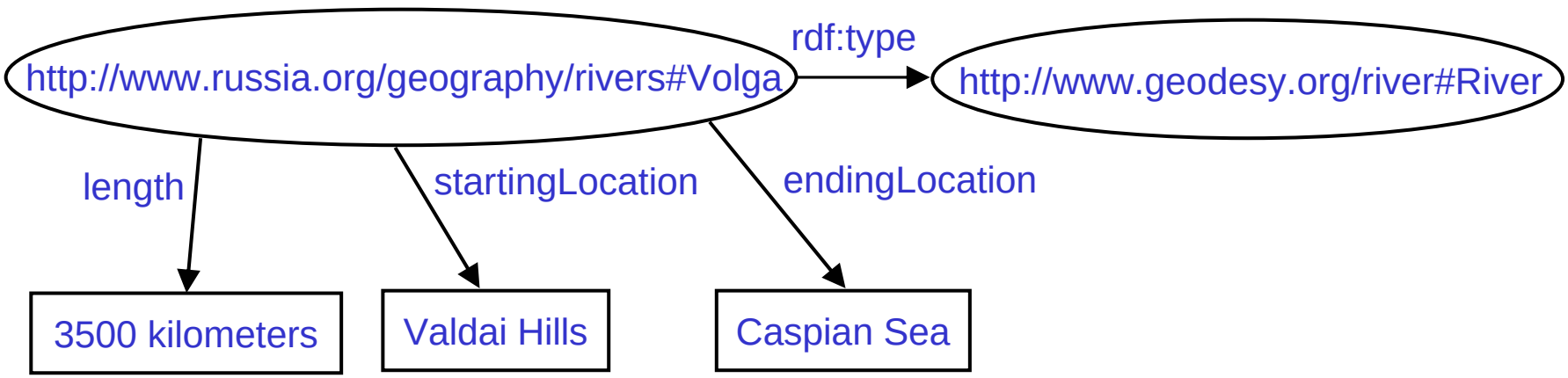
```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#»
   xmlns:dc="http://purl.org/dc/elements/1.1/»
   xmlns:ex="http://www.example.org/terms/»>
   <rdf:Description rdf:about="http://www.w3.org/Home/Lassila">
     <rdf:type rdf:resource=http://www.example.org/terms/Homepage"/>
     <dc:Creator>Ora Lassila</dc:Creator>
     <ex:createdWith>
      <rdf:Description  rdf:about="http://www.w3.org/amaya">
         <rdf:type rdf:resource="http://www.example.org/terms/HTMLEditor"/>
      </rdf:Description>
     </ex:createdWith>
   </rdf:Description>
</rdf:RDF>
```

RDF/XML: тип как имя элемента

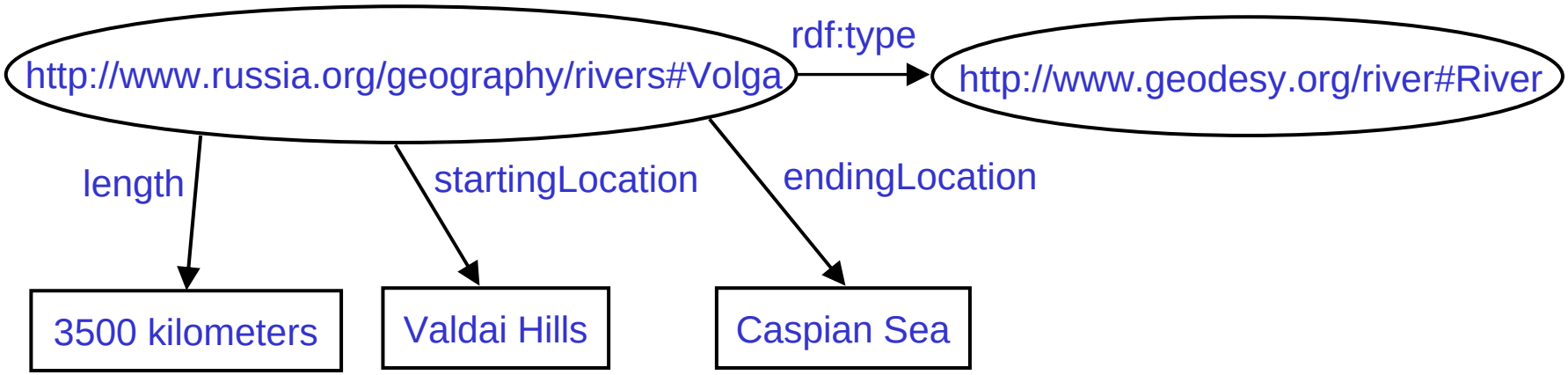


RDF-модель: Еще пример . . .

- Множество утверждений:
  1. Волга - это река
  2. Длина Волги равна 3500 км.
  3. Исток Волги находится на Валдайской Возвышенности.
  4. Волга впадает в Каспийское море.



RDF/XML: Еще пример. . .




```
<?xml version="1.0"?>
<rdf:Description
    rdf:about="http://www.russia.org/geography/rivers#Volga"
                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns="http://www.geodesy.org/river#">
    <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
    <length>3500 kilometers</length>
    <startingLocation>Valdai Hills</startingLocation>
    <endingLocation>Caspian Sea</endingLocation>
</rdf:Description>
```

RDFS

RDF Схема (RDFS)

- RDF дает формализм для аннотирования ресурсов с помощью метаданных и способ его записи в XML, но не дает никакого *конкретного смысла* элементам словаря, таким как subClassOf или type
  - Они интерпретируются как произвольные бинарные отношение

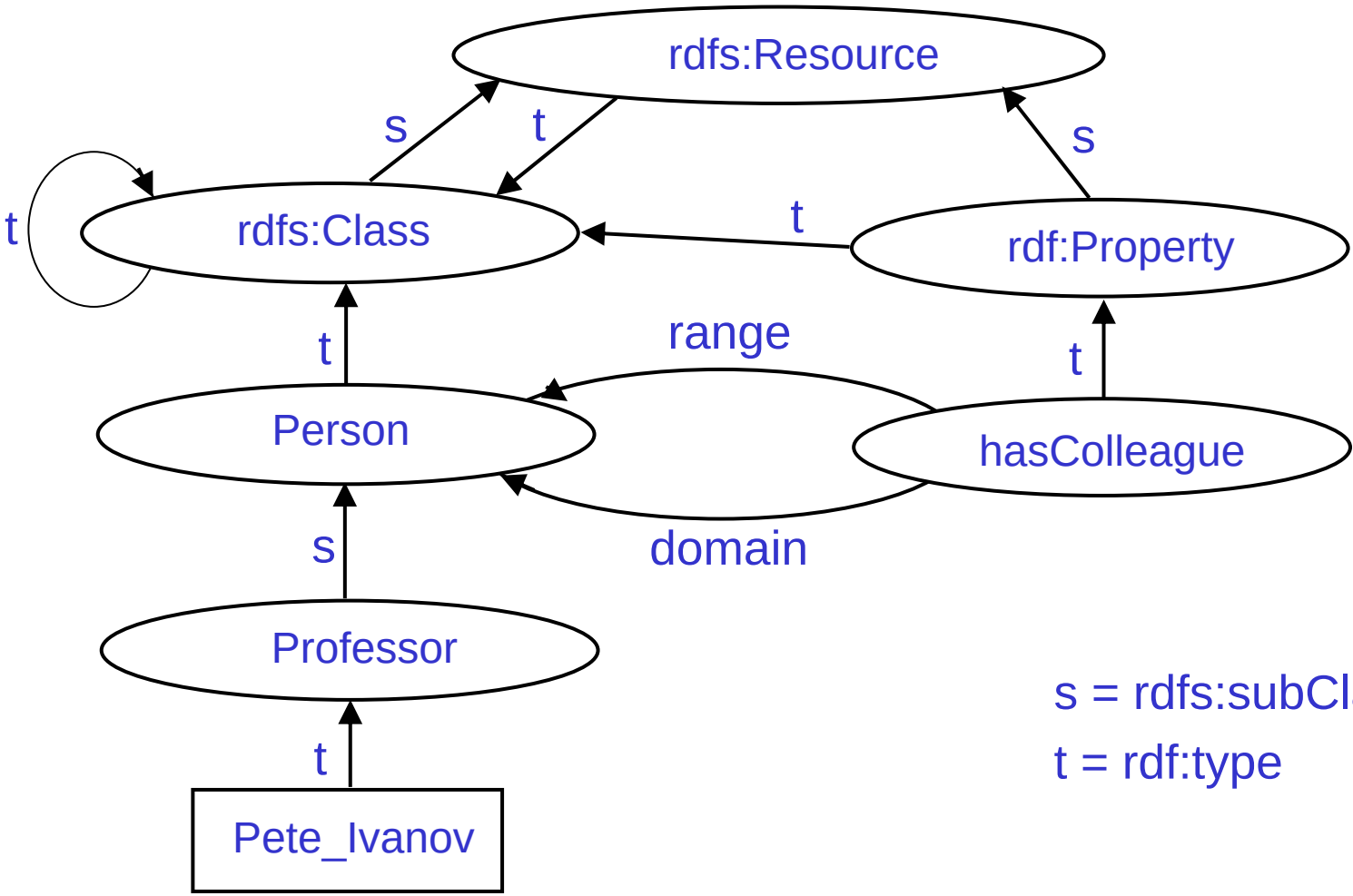
- RDF Схема (RDFS) позволяет *определить словарь терминов* и отношения между этими терминами
  - придает конкретным RDF предикатам и ресурсам “дополнительный смысл”
  - этот “дополнительный смысл ” (семантика) специфицирует, как термин должен интерпретироваться

RDFS

- Термины RDF-схемы:
  - rdfs:Class (Класс);
  - rdf:Property (Свойство);
  - rdf:type (тип);
  - rdfs:subClassOf (подкласс);
  - rdfs:subPropertyOf;
  - rdfs:range (область значений);
  - rdfs:domain (область определения)
- Эти термины используются как строительные блоки RDF-схемы (конструкторы) используемыми для создания словарей:

```
<Person,type,Class>
<hasColleague,type,Property>
<Professor,subClassOf,Person>
<Pete_Ivanov,type,Professor>
<hasColleague,range,Person>
<hasColleague,domain,Person>
```

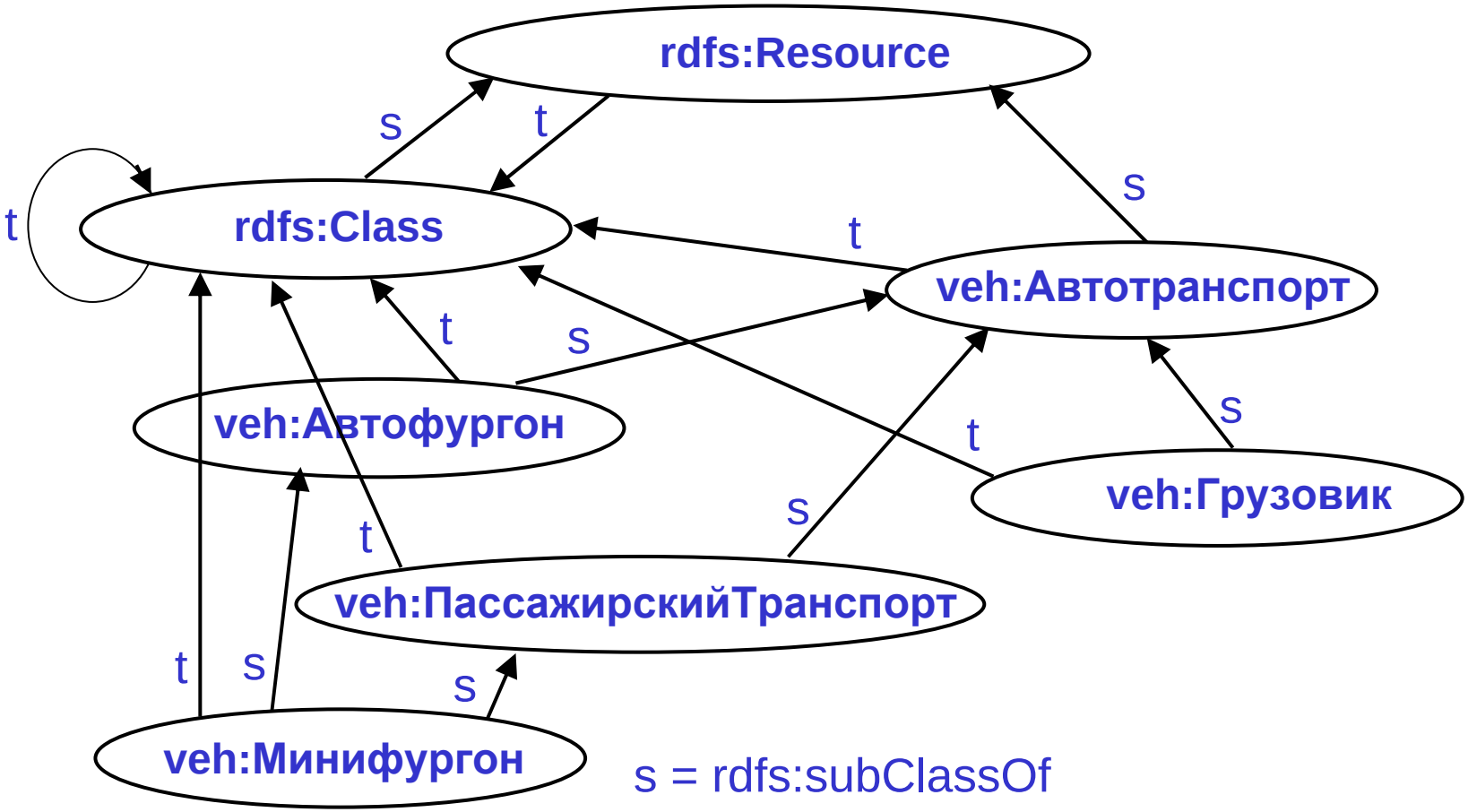
RDFS: пример 1



s = rdfs:subClassOf

t = rdf:type

RDFS: пример 2



s = rdfs:subClassOf

t = rdf:type

veh: - пространство имен, относящееся к транспортным средствам

Особенности RDF(S)

- Нет разницы между классами и экземплярами (индивидуами):

  <Species,type,Class>

  <Lion,type,Species>

  <Chandor,type,Lion>

- Свойства сами могут иметь свойства:

  <hasDaughter,subPropertyOf,hasChild>

  <hasDaughter,type,familyProperty>

- Нет различия между конструкторами языка и онтологическим словарем, поэтому конструкторы могут применяться к самим себе и друг к другу:

  <type,range,Class>

  <Property,type,Class>

  <type,subPropertyOf,subClassOf>

Недостатки RDFS

- RDFS слишком беден, чтобы достаточно детально описывать ресурсы:
  - Отсутствуют ограничения, локализующие область определения и область значений
    - Нельзя сказать, что областью значений свойства hasChild, когда оно применяется к людям, является человек, а когда оно применяется к слонам - слон;
  - Отсутствуют ограничения существования/кардинальности
    - Нельзя сказать, что все *экземпляры* класса людей имеют мать, которая также относится к этому классу или что человек имеет в точности двух родителей
  - Отсутствуют транзитивные, инверсные или симметричные свойства
    - Нельзя сказать, что isPartOf является транзитивным свойством, что hasPart является инверсным к isPartOf или что касание симметрично
- Трудно обеспечить поддержку рассуждений:
  - Отсутствуют “органически присущие” машины логического вывода для нестандартной семантики
  - Возможно для рассуждений может быть использована аксиоматизация логики первого порядка

Языки онтологий для Семантического Web

OWL

Онтологии в Computer Science

“Явная спецификация концептуализации” [Gruber, 1993]

- Онтология является искусственно созданным (инженерным) объектом:
  - **Состоит из конкретного словаря, используемого для описания определенной реальности и**
  - **Множества явных допущений, относящихся к подразумеваемому значению словаря.**
- Онтология описывает формальную спецификацию определенной предметной области:
  - **Общее (разделяемое) понимание рассматриваемой предметной области**
  - **Формальная, допускающая машинную обработку модель рассматриваемой предметной области**

Структура онтологии

Онтологии обычно имеют два компонента:

- Имена понятий, важных для рассматриваемой предметной области:
  - **Слон – класс (понятие), член которого является видом животных**
  - **Травоядные – понятие, членами которого являются в точности те животные, которые едят только растения или части растений**
  - **Взрослый_Слон – понятие, членами которого являются в точности те слоны, чей возраст больше 20 лет**
- Базовые знания и ограничения предметной области:
  - **Взрослые_Слоны имеет вес не менее 2 000 кг**
  - **Все Слоны являются либо Африканскими_Слонами либо Индийскими_Слонами**
  - **Никакой индивидуум не может быть одновременно Травоядным и Плотоядным**

Языки онтологий

- **Большое разнообразие языков для “явной спецификации”**
  - **Графические нотации:**
    - **Семантические сети**
    - **Карты понятий (Topic Maps) (http://www.topicmaps.org/)**
    - **UML**
    - **RDF**
  - **Основанные на логике:**
    - **Дескриптивные логики (OIL, DAML+OIL, OWL)**
    - **Правила (RuleML, LP/Prolog)**
    - **Логика первого порядка (KIF)**
    - **Концептуальные графы**
    - **(Синтаксические) логики высших порядков (LBase)**
    - **Неклассические логики (FLogic, немонотонные логики, модальности)**
  - **Вероятностные/нечеткие**
- **Степень формальности широко варьируется**
  - **Возрастающая формальность делает языки в большей степени пригодными для машинной обработки (автоматических рассуждений)**

Многие языки используют “объектно-ориентированную” модель, основанную на:

- Объекты/Экземпляры/Индивидуумы:
  - **элементы области дискурса**
  - **в логике первого порядка - эквивалентны константам**
- Типы/Классы/Понятия
  - **множества объектов, имеющих общие характеристики**
  - **в логике первого порядка - эквивалентны унарным предикатам**
- Отношения/Свойства/Роли
  - **множества пар (троек) объектов**
  - **в логике первого порядка - эквивалентны бинарным предикатам**
- Такие языки:
  - **хорошо понимаемы;**
  - **формально специфицированы;**
  - **(относительно) легки в использовании**
  - **поддаются машинной обработке**

Требования к языку онтологий для Web

Язык Web-онтологий должен:

- Расширять существующие стандарты Web
  - такие как XML, RDF, RDFS
- Быть простым для понимания и использования
  - должен базироваться на хорошо известных способах *представления знаний*
- Быть формально специфицированным
- Обладать “адекватной” выразительной мощностью
- Возможность обеспечить поддержку автоматических рассуждений

От RDF к OWL

- **Чтобы удовлетворить указанным требованиям были разработаны два языка:**
  - **OIL - разработан группой (в основном) Европейских исследователей**
  - **DAML-ONT - разработан группой (в основном) американских исследователей (в рамках выполняемой DARPA программы DAML)**
- **Усилия были объединены для разработки DAML+OIL**
  - **Разработка была выполнена “Объединенным EU/US Комитетом по агентным языкам разметки - Agent Markup Languages”**
  - **Расширяет (DL подмножество) RDF**
- **DAML+OIL был представлен W3C как основа для стандартизации**
  - **была сформирована рабочая группа Web-Ontology (WebOnt)**
  - **WebOnt на основе DAML+OIL разработала язык OWL**
  - **OWL - W3C Recommendation !!!**

Язык OWL

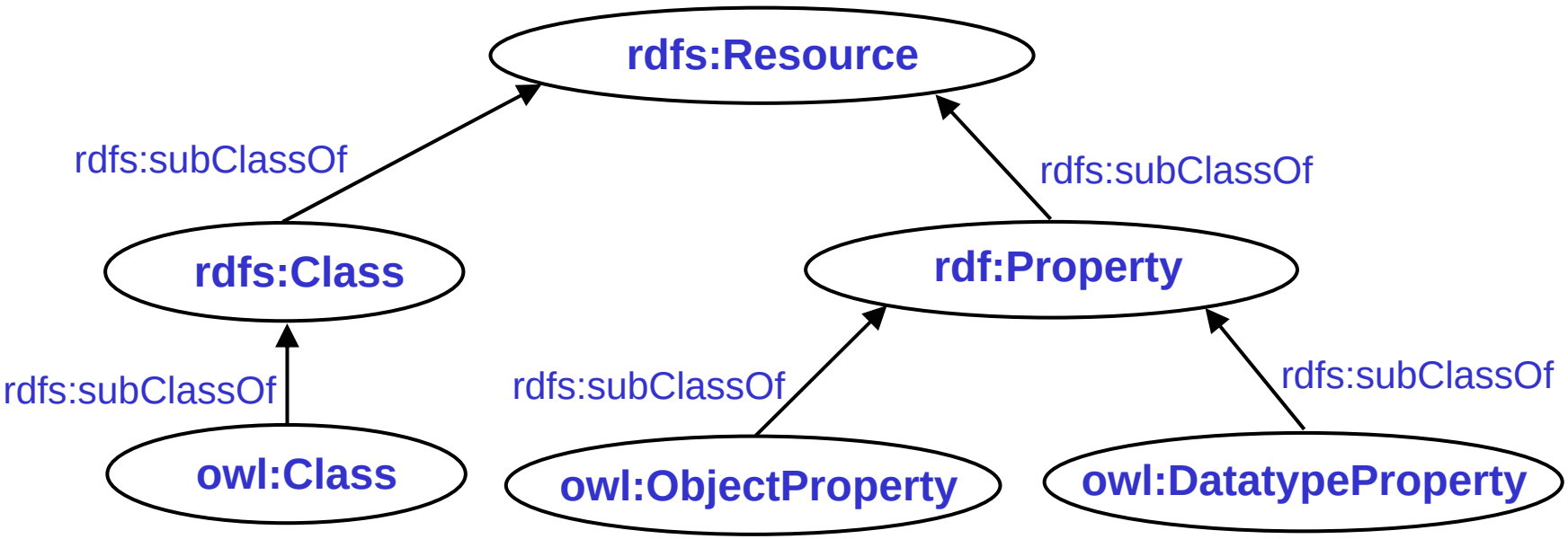
- Три уровня OWL
  - OWL full - объединение OWL-синтаксиса и RDF
  - OWL DL - ограничен фрагментом логики первого порядка (¼ DAML+OIL)
  - OWL Lite - “простое для реализации” подмножество OWL DL
- Уровни семантики:
  - OWL DL составляет ¼ OWL full в рамках фрагмента DL
  - семантика DL официально определена
- OWL DL основан на Дескриптивной Логике SHIQ
  - В действительности он эквивалентен SHOIN(D_n) DL
- Достоинства OWL DL:
  - Хорошо определенная семантика
  - Хорошо осмыслены формальные свойства (сложность, разрешимость)
  - Известны алгоритмы рассуждений
  - Имеются (очень оптимистичные) реализации систем

OWL: Три уровня

Язык OWL имеет три уровня:

- **OWL Lite:**
  – Классификационные иерархии;
  – Простые ограничения.
- **OWL DL:**
  – Максимальная выразительность при сохранении разрешимости;
  – Стандартная формализация.
- **OWL Full:**
  – Очень высокая выразительность (метаклассы, классы как значения);
  – Полная синтаксическая свобода RDF (самомодифицируемость);
  – Потеря разрешимости (алгоритмы рассуждения становяться неэффективными).

Отношения между элементами OWL и RDF(S)



OWL-документы: пространства имен

- *OWL-документы* (обычно называемые OWL-онтологиями) являются *RDF-документами* (т.е. имеют корневой элемент rdf:RDF)
- В этом элементе обычно специфицируются следующие пространства имен:

```
<rdf:RDF
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XLMSchema#">
```

OWL-документы: преамбула

- OWL-онтология может начинаться с множества утверждений для служебных целей. Эти утверждения группируются в элементе **owl:Ontology**, содержащем *комментарии*, *управление версиями* и *включение других онтологий*.

- Например:

```
<owl:Ontology rdf:about="">
   <rdfs:comment>An example OWL ontology</rdfs:comment>
   <owl:priorVersion rdf:resource="http://www.mydomain.org/uni-ns-old"/>
   <owl:imports rdf:resource="http://www.mydomain.org/persons"/>
   <rdfs:label>University Ontology</rdfs:label>
</owl:Ontology>
```

- **owl:imports** - единственное утверждение, имеющее последствия для логического значения онтологии. Перечисляет другие онтологии, содержание которых подразумевается как часть текущей онтологии.

OWL: Элемент owl:Class

- Классы определяются, используя элемент **owl:Class**, который является подклассом **rdfs:Class**.
- Пример:

```
<owl:Class rdf:ID="associateProfessor">
  <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</owl:Class>
```

- Используя элемент **owl:disjointWith** можно сказать, что этот класс не пересекается с классами professor и assistantProfessor. Эти элементы могут включаться в приведенное выше определение или добавляться путем ссылки на id, используя rdf:about. Этот механизм наследуется от RDF:

```
<owl:Class rdf:about="associateProfessor">
    <owl:disjointWith rdf:resource="#professor"/>
    <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>
```

OWL: эквивалентность классов. Классы owl:Thing и owl:Nothing

- Эквивалентность классов может быть определена, используя элемент **owl:equivalentClass**:

```
<owl:Class rdf:ID="faculty">
    <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

- Два заранее определенных класса:
  - **owl:Thing -** наиболее общий класс, содержащий ***все***. (все является вещью);
  - **owl:Nothing** - пустой класс.

- Каждый класс является подклассом **owl:Thing** и суперклассом **owl:Nothing**

OWL: Элементы свойств (Property elements)

- В OWL **два вида свойств**:
  - ***Объектные свойства*** (Object properties) - связывают объекты с другими объектами. Например: **isTaughtBy**, **supervises** и т.д.
  - ***Свойства - типы данных*** (Datatype properties) - связывают объекты со значениями типов данных. Например: phone, title, age и т.д.

- OWL не имеет предопределенных типов данных и способов их определения;
- OWL позволяет использовать типы данных XML Schema.

- Пример свойства - типа данных:

```
<owl:DatatypeProperty rdf:ID="age">
   < rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema#nonNegativeInteger"/ >
</owl:DatatypeProperty >
```

OWL: Пример объектного свойства

- Определяемые пользователем типы данных будут обычно собираться в XML-схему, а затем использоваться в OWL-онтологии.
- Пример объектного свойства:

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
   <owl:domain rdf:resource="#course"/>
   <owl:range rdf:resource="#academicStaffMember"/>
   <rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>
```

OWL: Инверсные свойства

- OWL позволяет задавать «инверсные свойства», например: **isTaughtBy** и **teaches**.

```
<owl:ObjectProperty rdf:ID="teaches">
    <rdfs:domain rdf:resource="#academicStaffMember"/>
    <rdfs:range rdf:resource="#course"/>
    <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

- Области определения и область значений могут наследоваться от инверсного свойства (перестановкой области определения и области значений)

OWL: Эквивалентные свойства

- Эквивалентность свойств может быть определена используя элемент **owl:equivalentProperty**:

```
<owl:ObjectProperty rdf:ID="lecturesIn">
    <owl:equivalentProperty rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

OWL: Ограничения свойств

- Элемент **owl:Restriction** в общем случае содержит элемент **owl:onProperty** и одно или более объявлений ограничений.

- Два типа ограничений:
  - ограничения *на тип значений* свойства (owl:allValuesFrom, owl:hasValue и owl:someValuesFrom);
  - ограничения *кардинальности* (числа значений)

OWL: Ограничения свойств - allValuesFrom

- **rdfs:subClassOf** позволяет специфицировать класс C как подкласс другого класса C'. Объявление класса C, элементы которого удовлетворяют определенным условиям, эквивалентно утверждению, что C является подклассом C', в котором собраны все объекты, удовлетворяющие этим условиям.

- Пример - элемент определяет курсы первого года обучения, читаемые только профессорами:

```
<owl:Class rdf:about="#firstYearCourse">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#isTaughtBy"/>
         <owl:allValuesFrom rdf:resource="#Professor"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

- **owl:allValuesFrom** описывает класс возможных значений, которые может принимать свойство, специфицированное элементом **owl:onProperty** (значением свойства **isTaughtBy** могут быть только профессора).

OWL: Ограничения свойств - hasValue

- Объявление, что курсы по математике читаются профессором, имеющим идентификатор 949352:

```
<owl:Class rdf:about="#mathCourse">
  <rdfs:subClassOf >
      <owl:Restriction >
        <owl:onProperty rdf:resource="#isTaughtBy"/ >
        <owl:hasValue rdf:resource="#949352"/ >
      </owl:Restriction >
   < /rdfs:subClassOf >
</owl:Class >
```

- owl:hasValue утверждает, что свойство, специфицированное элементом owl:onProperty должно иметь конкретное значение.

OWL: Ограничения свойств - someValuesFrom

- Пример:

```
<owl:Class rdf:about="#academicStaffMember">
   <rdfs:subClassOf>
       <owl:Restriction>
       <owl:onProperty rdf:resource="#teaches"/>
       <owl:someValuesFrom rdf:resource="#undergraduateCourse"/>
       </owl:Restriction >
    </rdfs:subClassOf >
</owl:Class >
```

- Преподаватели должны читать по крайней мере один курс уровня undergraduate.

OWL: Ограничения кардинальности свойств

- Пример «Каждый курс читается по крайней мере одним преподавателем»:

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
      1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- Литерал «1» интерпретируется как неотрицательное целое (nonNegativeInteger), а не как строка или еще что-то;
- Используется объявление пространства имен **xsd**, сделанное в элементе-заголовке для ссылки на документ XML-Schema.

OWL: Ограничения кардинальности свойств

- «Кафедра должна включать от не менее десяти и не более тридцати сотрудников»:

```
<owl:Class rdf:about="#department">
    <rdfs:subClassOf>
       <owl:Restriction>
          <owl:onProperty rdf:resource="#hasMember"/>
          <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
            10
          </owl:minCardinality>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            30
          </owl:maxCardinality>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Интеллектуальные агенты

Что такое Интеллектуальный Агент?
“Слабое” определение:

Система, обладающая следующими свойствами:

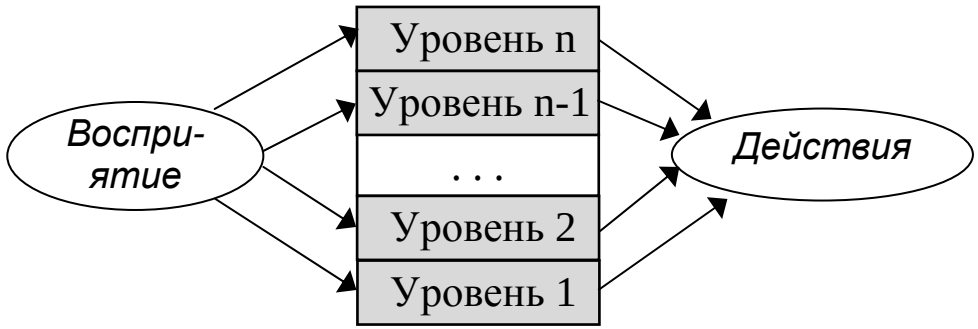
- ***автономность*** - способность функционировать без вмешательства человека, контролируя свои действия и внутреннее состояние;
- ***социальность*** (social ability) - способность функционировать в сообществе агентов и обмениваться с ними сообщениями с помощью некоторого языка коммуникаций;
- ***реактивность*** - способность воспринимать состояние среды и своевременно реагировать на происходящие в ней изменения;
- ***внутренняя активность*** (pro-activity) - способность проявлять инициативу, т.е. не только реагировать на внешние события, но и генерировать цели и действовать рационально для их достижения.

Что такое Интеллектуальный Агент?
“Сильное” определение:

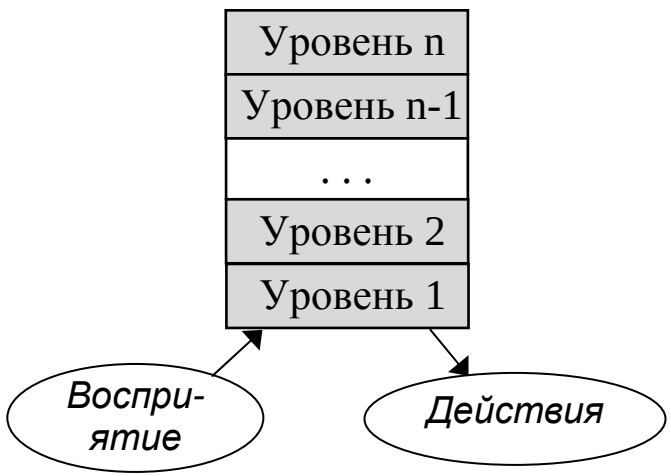
Система, обладающая следующими «ментальными» свойствами (или их подмножеством):

- ***знания* (knowledge)** - постоянные, неизменяемые в процессе функционирования знания агента о себе, среде и других агентах;
- ***убеждения* (beliefs)** - знания агента о среде (в том числе, о других агентах), которые могут стечением времени изменяться и становиться неверными;
- ***желания* (desires)** - состояния, которых агент желает достичь (могут быть противоречивыми), аналогичны *целям*;
- ***намерения* (intentions)** - действия, собирается выполнить вследствие своих желаний или в силу взятых на себя обязательств;
- ***обязательства* (commitments)** - задачи, решение которых агент берет на себя в рамках кооперации с другими агентами по их просьбе или поручению.

Классы архитектур ИА: взаимодействие уровней

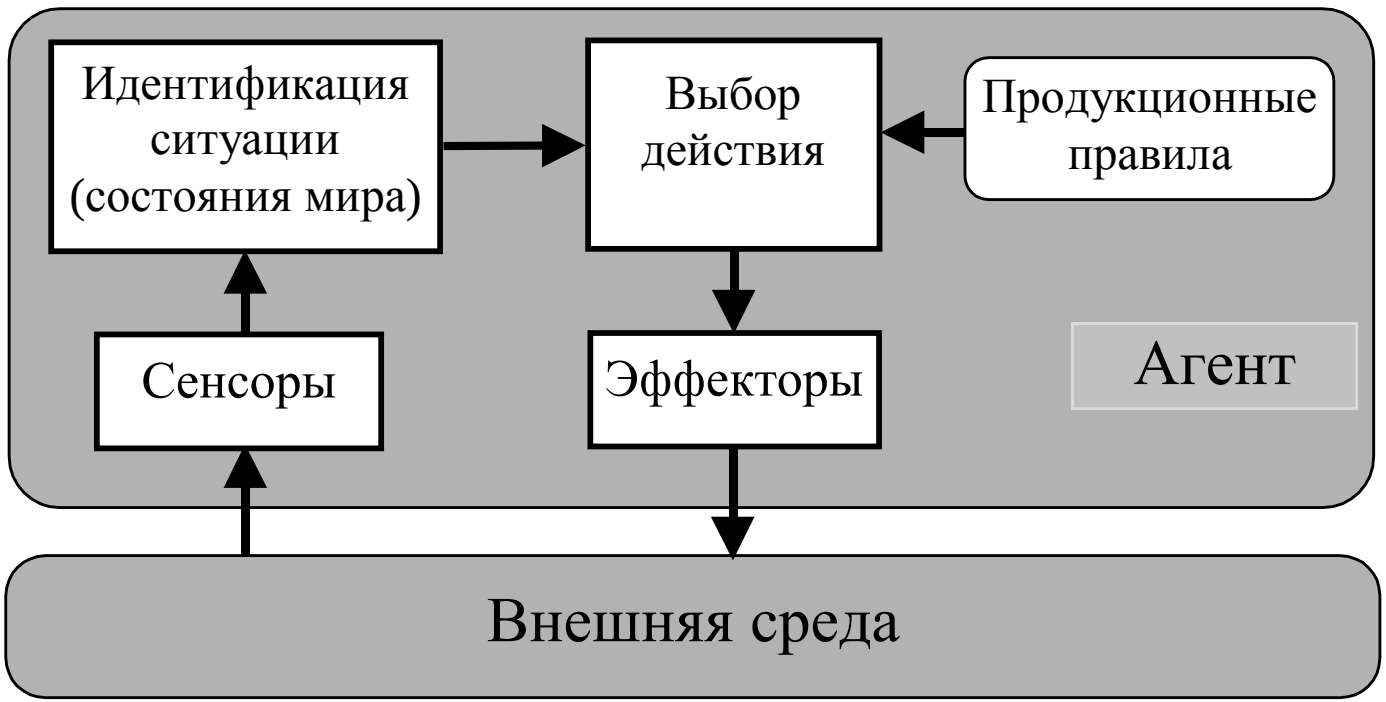


а) Горизонтально-организованная архитектура

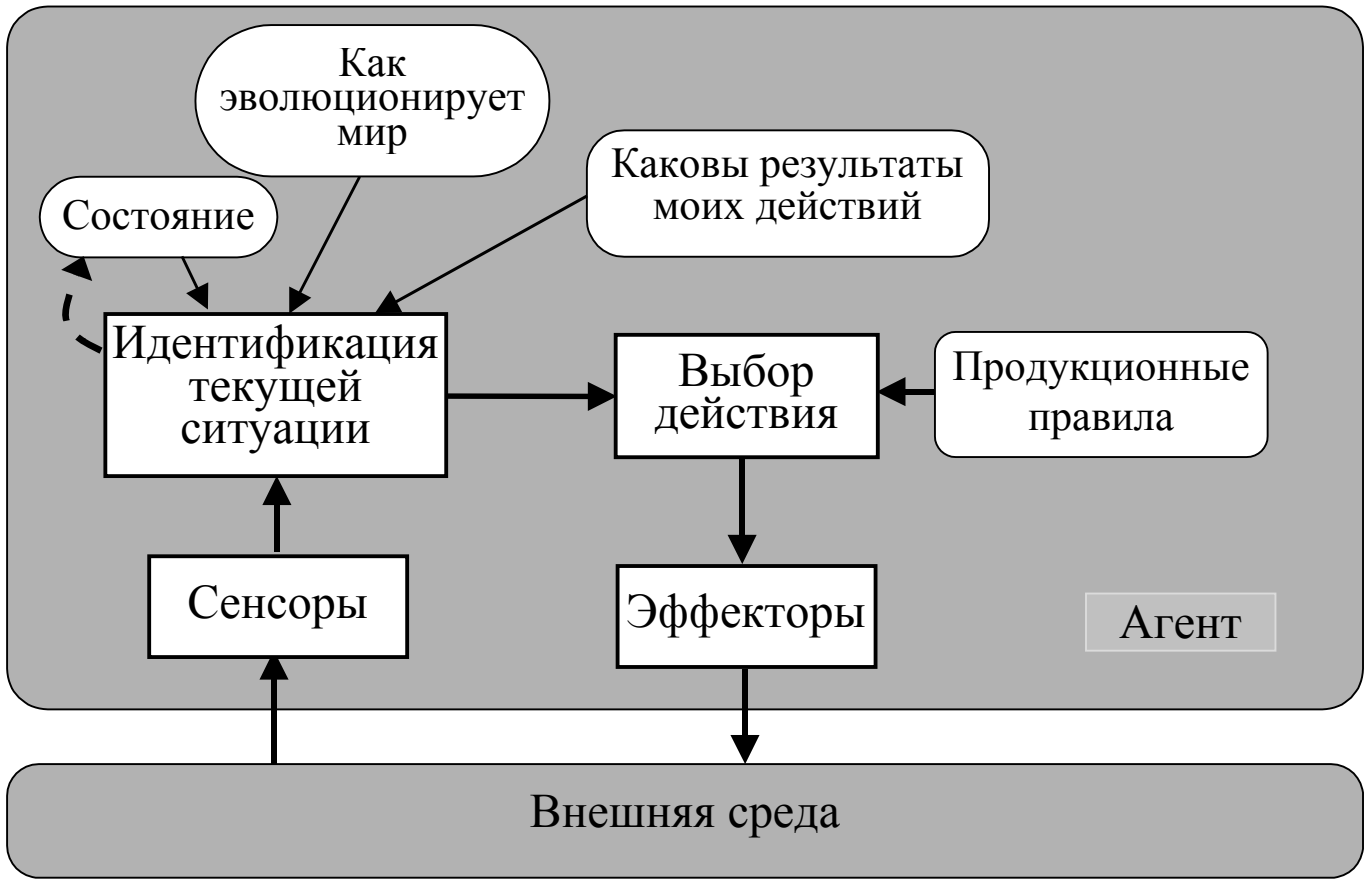


б) Вертикально-организованные архитектуры

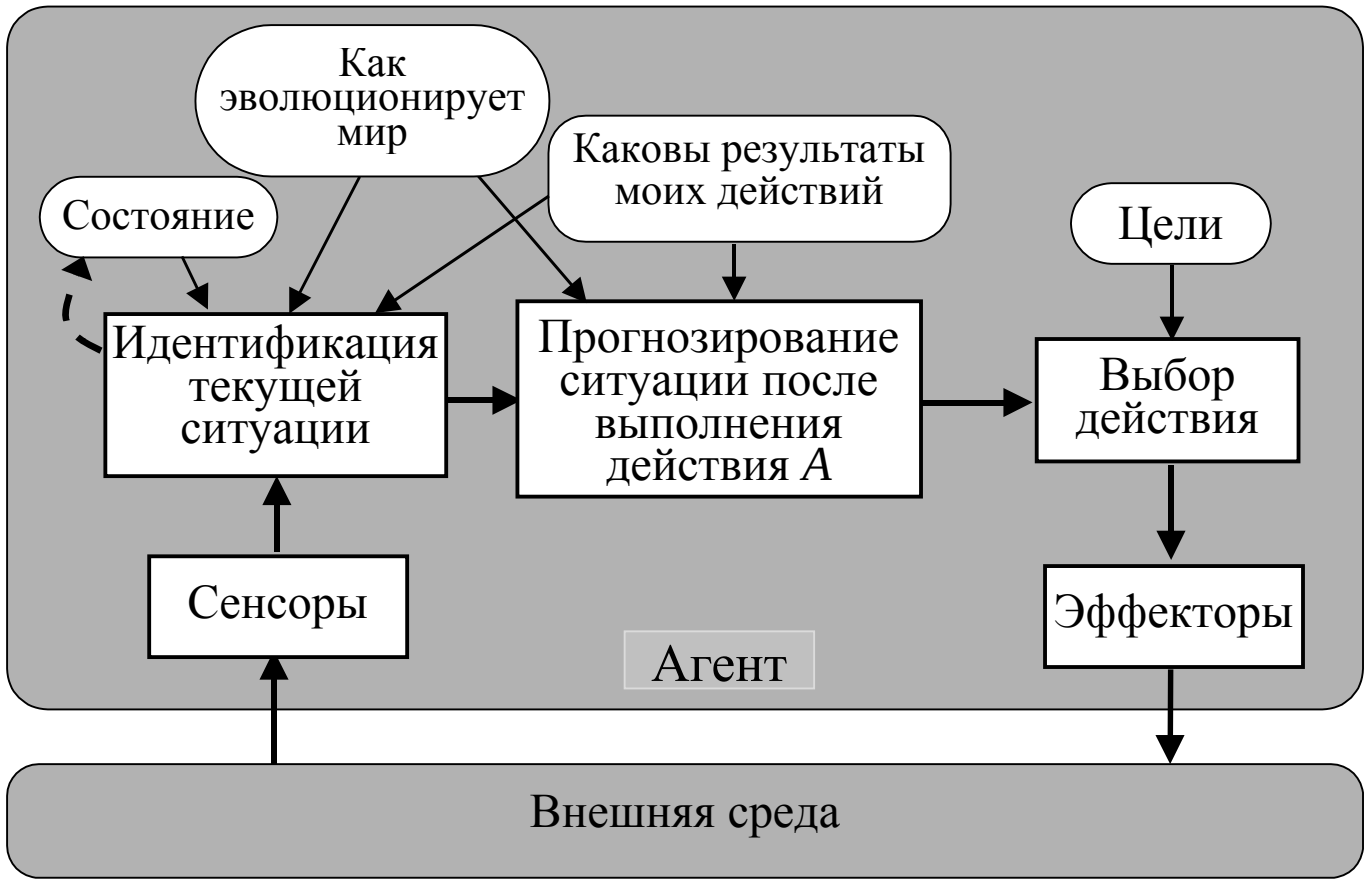
Архитектура простого реактивного агента



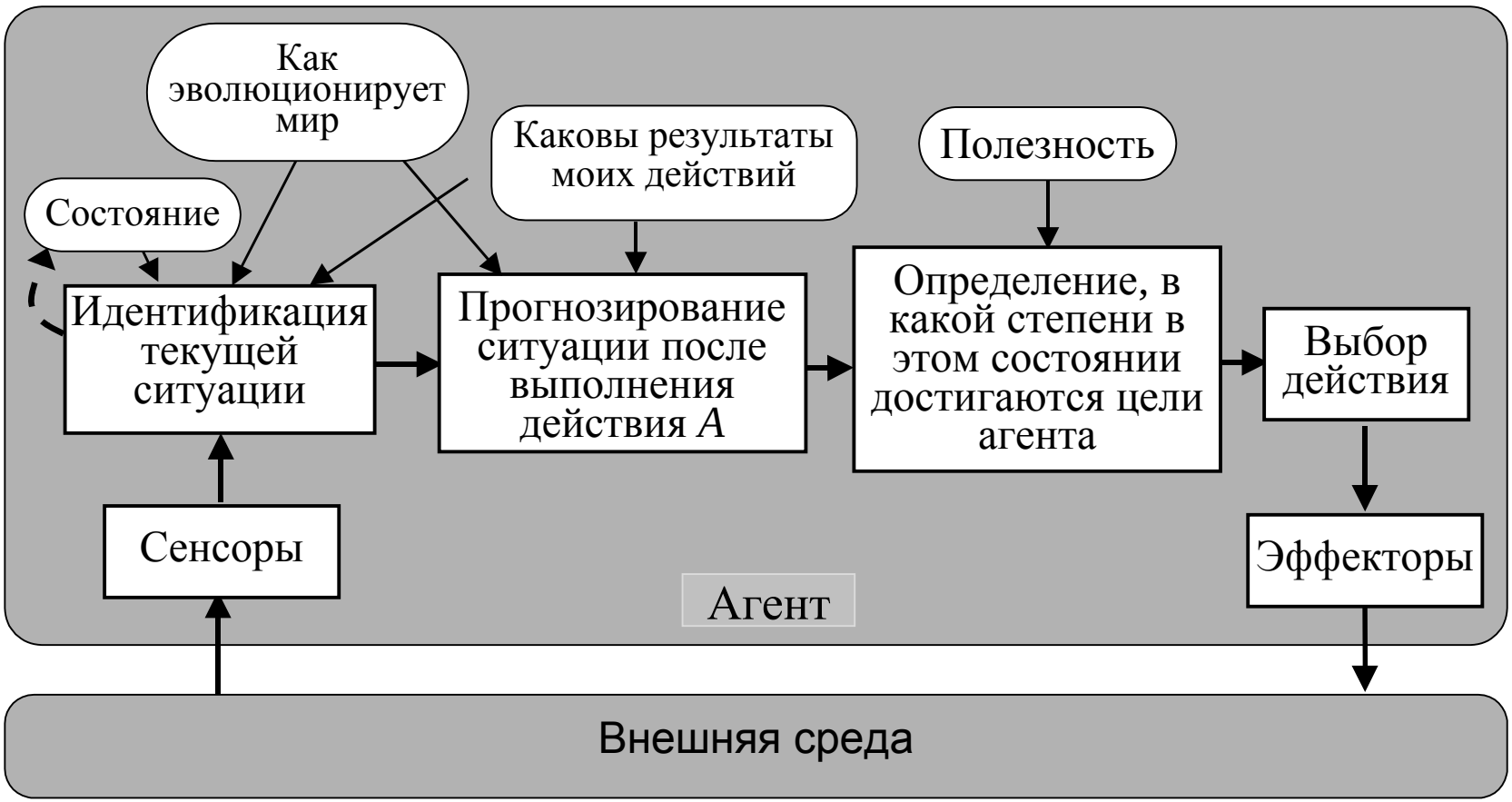
Архитектура основанного на моделях реактивного агента



Архитектура агента, управляемого целями

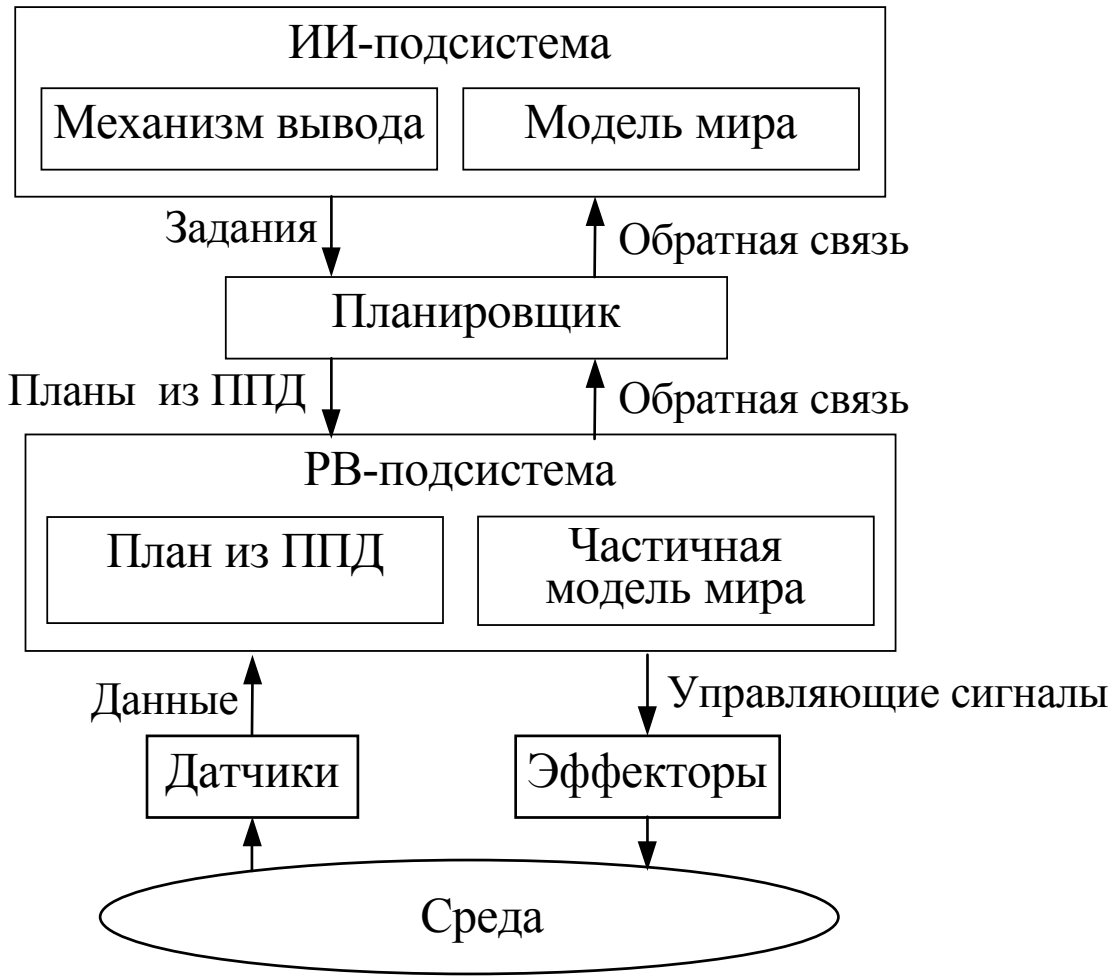


Архитектура агента, учитывающего полезность действий



Архитектуры ИА: InterRap

Архитектуры ИА РВ: CIRCA



Стандартизация в области ИА

FIPA (Foundation for Intelligent Physical Agents) - международная организация, созданная в 1996 г. (http://www.fipa.org/)

Цель – поддержка продвижения коммерческих приложений технологии ИА путем разработки открытых спецификаций, поддерживающих интероперабельность агентов и агентных сервисов.

Согласно FIPA агент «обладает способностью предоставлять в рамках унифицированной и интегрированной исполнительной модели один или более сервисов, которые могут включать доступ к внешнему программному обеспечению, пользователям и коммуникационным возможностям»

Категории спецификаций FIPA

Приложения

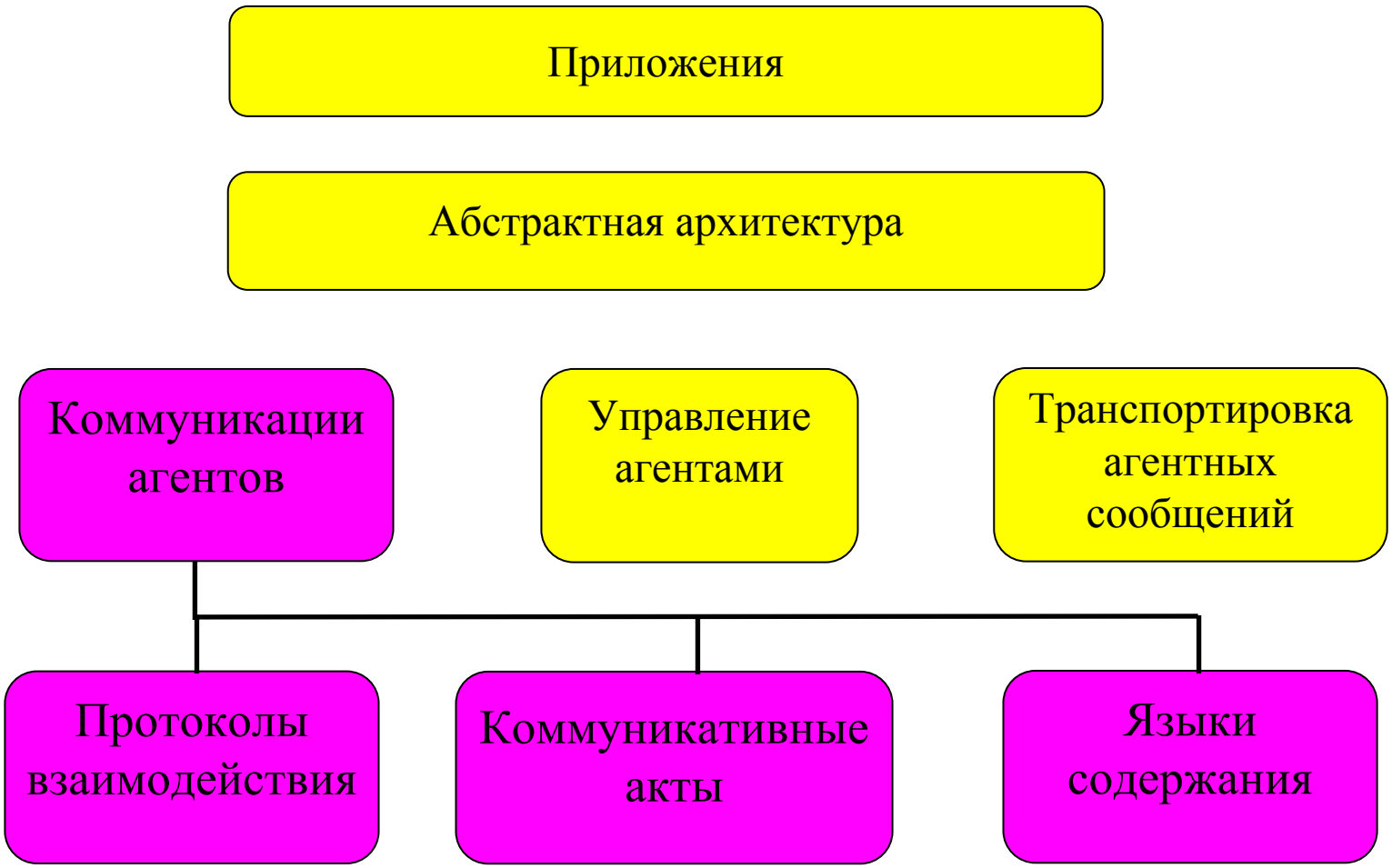
Абстрактная архитектура

Коммуникации агентов

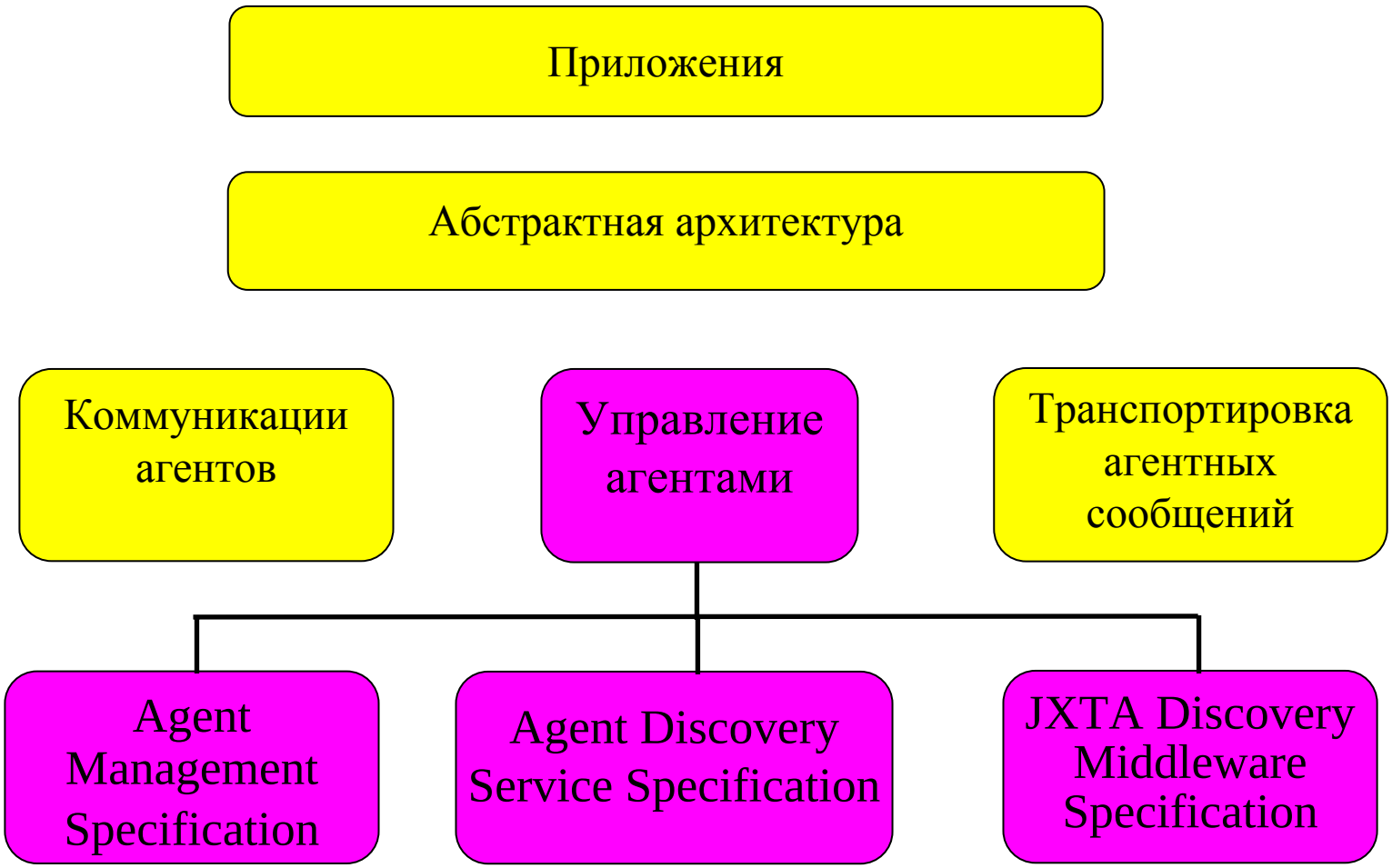
Управление агентами

Транспортировка агентных сообщений

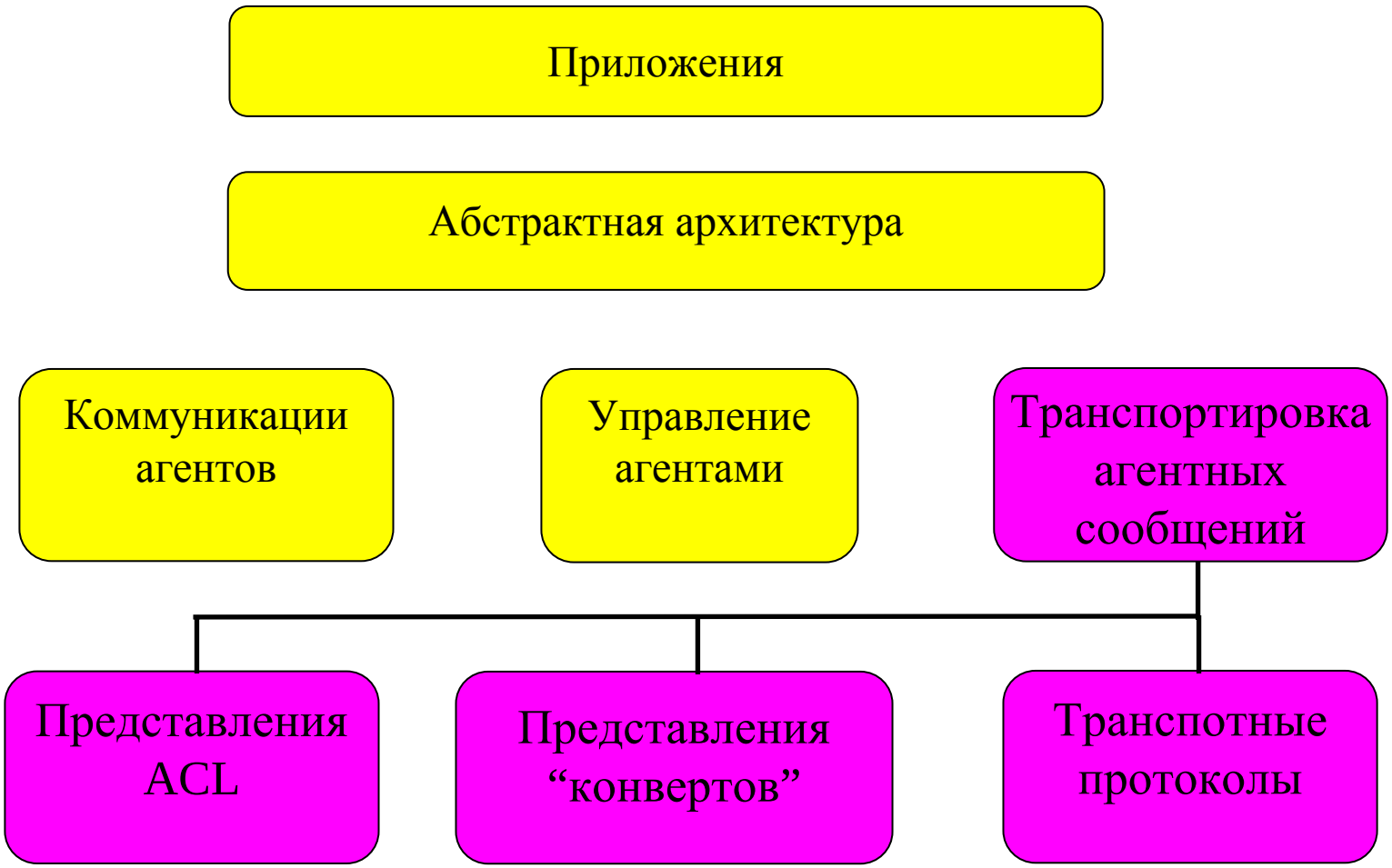
FIPA: коммуникации агентов



FIPA: управление агентами



FIPA: транспортировка сообщений



Ключевые спецификации FIPA:

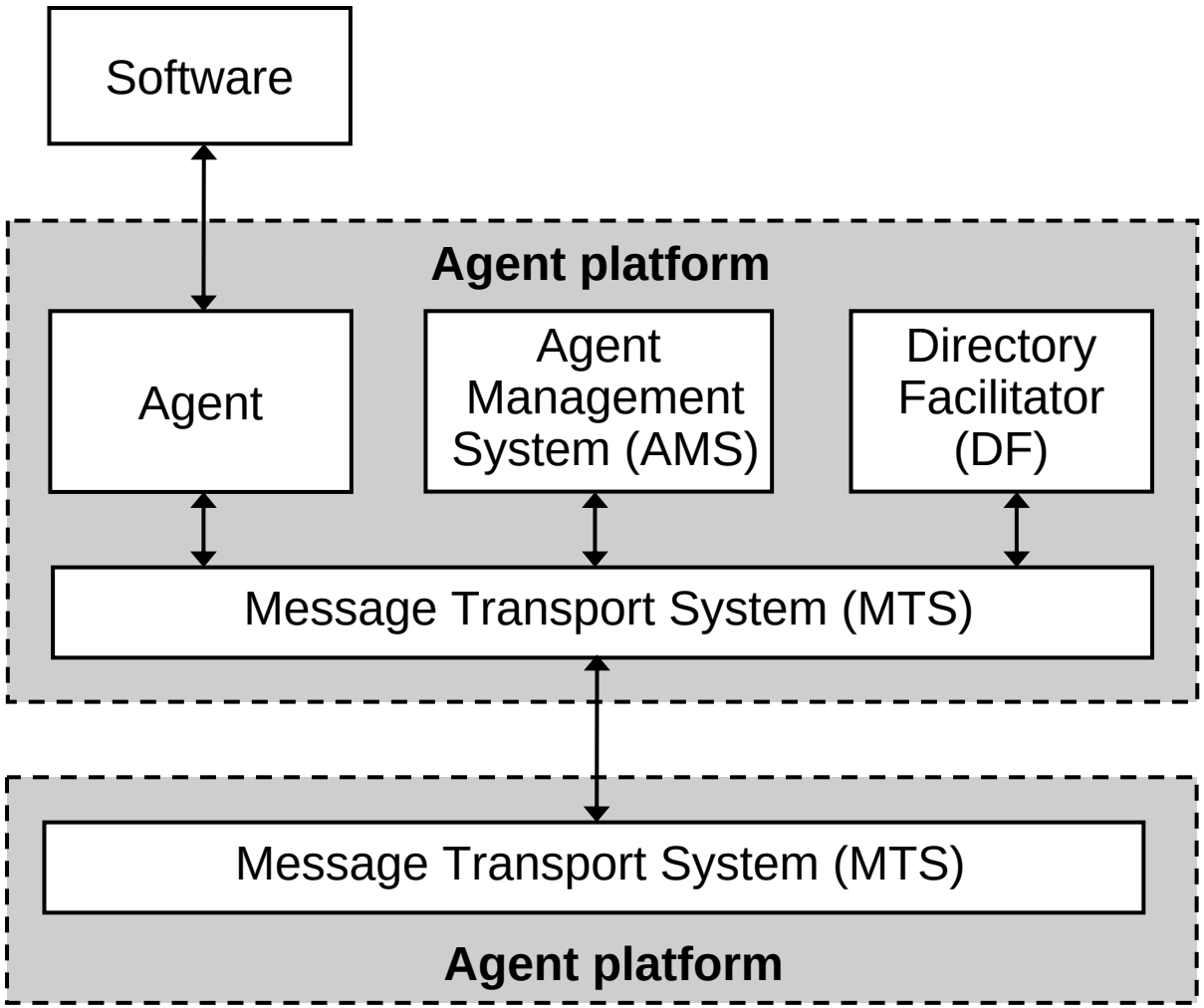
00001 – Abstract Architecture Specification
00007 – Content Languages Specification
00008 – SL Content Language Specification
00023 – Agent Management Specification
00024 – Agent Message Transport Specification
00025 – Interaction Protocol Library Specification
00037 – Communicative Act Library Specification
00061 – ACL Message Structure Specification
00063 – Control Agent Specification
00082 – Network Management and Provisioning Specification
00084 – Agent Message Transport Protocol for HTTP Specification
00086 – Ontology Service Specification
00087 – Agent Management Support for Mobility Specification
00094 – Quality of Service Specification

В настоящее время (ноябрь 2004 г.) разработаны и находятся в разработке *94* спецификации

Понятие агентной платформы по FIPA

- *Эталонная модель управления агентами* - множество логических компонентов, каждый из которых предоставляет множество возможностей, которые могут комбинироваться в физических реализациях агентных платформ.
- *Агентная платформа* (АП) предоставляет физическую инфраструктуру, в которой могут быть размещены агенты.
- АП состоит из компьютера(ов), операционной системы, поддерживающего агентов программного обеспечения, *компонентов управления агентами по FIPA* (DF, AMS и MTS) и агентов.
- Внутренняя конструкция АП определяется разработчиками агентных систем и не является предметом стандартизации FIPA.
- FIPA рассматривает только способ реализации коммуникаций между «родными» для данной АП агентами и внешними или динамически регистрирующимися на АП агентами.

Структура агентной платформы по FIPA



Компоненты агентной платформы: Агенты

- *Агент* – основное «действующее лицо» на АП:
  - Сочетает одну или более сервисных возможностей *в унифицированной и интегрированной исполнительной модели*, которая может включать доступ к внешнему программному обеспечению, людям-пользователям и коммуникационные возможности;
  - Должен иметь как минимум одного владельца (например, на основе принадлежности к организации или человеку-владельцу);
  - Может поддерживать несколько понятий идентичности;
  - *Идентификатор агента* (Agent Identifier - AID) помечает агента так, что его можно *однозначно выделить* внутри Агентного Универсума;
  - Может регистрироваться с множеством транспортных адресов, по которым с ним можно контактировать;
  - Может иметь возможности брокера ресурсов для доступа к программному обеспечению;

Компоненты агентной платформы: AMS & DF

- *Agent Management System (AMS)* – *Система Управления Агентами:*
  - Обязательный компонент агентной платформы;
  - Выполняет супервизорное управление доступом к агентной платформе и ее использованием;
  - На одной агентной платформе может существовать только одна AMS;
  - Поддерживает каталог идентификаторов агентов (AID), содержащих (кроме прочего) транспортные адреса агентов, зарегистрированных на данной АП;
  - Предоставляет другим агентам сервис «белых страниц»;
  - Каждый агент должен зарегистрироваться у AMS для того, чтобы получить корректный AID.
- *Directory Facilitator (DF) – Служба Каталога:*
  - обязательный компонент агентной платформы
  - предоставляет агентам сервис «желтых страниц» - агенты могут регистрировать свои сервисы и/или выполнять поиск в службе каталога с целью найти агента, предоставляющего заданные сервисы;
  - на одной агентной платформе может существовать несколько служб каталогов и они могут объединяться в федерации.

Компоненты агентной платформы: MTS & Software

- *Message Transport Service (MTS)* – *Служба Транспортировки Сообщений:*
  - поддерживает заданный по умолчанию метод коммуникации между агентами на различных АП.

- *Software - стороннее (неагентное) программное обеспечение:*
  - совокупность не-агентного исполняемого кода, доступного *через* агента;
  - *не относится к агентной платформе;*
  - агенты могут обращаться к стороннему программному обеспечению, например, чтобы:
    - добавить новые сервисы;
    - приобрести новые коммуникационные протоколы;
    - приобрести новые протоколы/алгоритмы безопасности;
    - приобрести новые протоколы переговоров;
    - обратиться к инструментам поддерживающим миграцию;
    - . . .

Именование агентов - 1

- *Идентификатор агента (AID)* – расширяемая совокупность пар «параметр-значение», определяющих:
  - *Имя агента* (обязательный параметр);
  - *Транспортные адрес(а) агента* (необязательный параметр);
  - *Адрес(а) сервиса разрешения имен* (необязательный параметр).

(agent-identifier

:name	**<**символическое имя агента**>**
:addresses	**<**упорядоченная последовательность транспортных адресов, по которым можно контактировать с агентом**>**
:resolvers	**<**упорядоченная последовательность AID, по которым можно контактировать с сервисами разрешения имен для агента**>**

)

Порядок перечисления адресов и AID в параметрах :addresses и :resolvers определяет предпочтение их использования.

Именование агентов - 2

- Транспортный адрес обычно специфичен для Протокола Транспортировки Сообщений (Message Transport Protocol).
- Агент может поддерживать *множество методов коммуникации* - в параметре `:addresses` указывается множество значений транспортных адресов.
- Сервис разрешения имен предоставляется AMS через функцию поиска. Параметр `:resolvers` в AID содержит последовательность AID, по которым AID агента в конечном счете может быть разрешен в транспортный адрес или множество транспортных адресов.

Именование агентов - пример

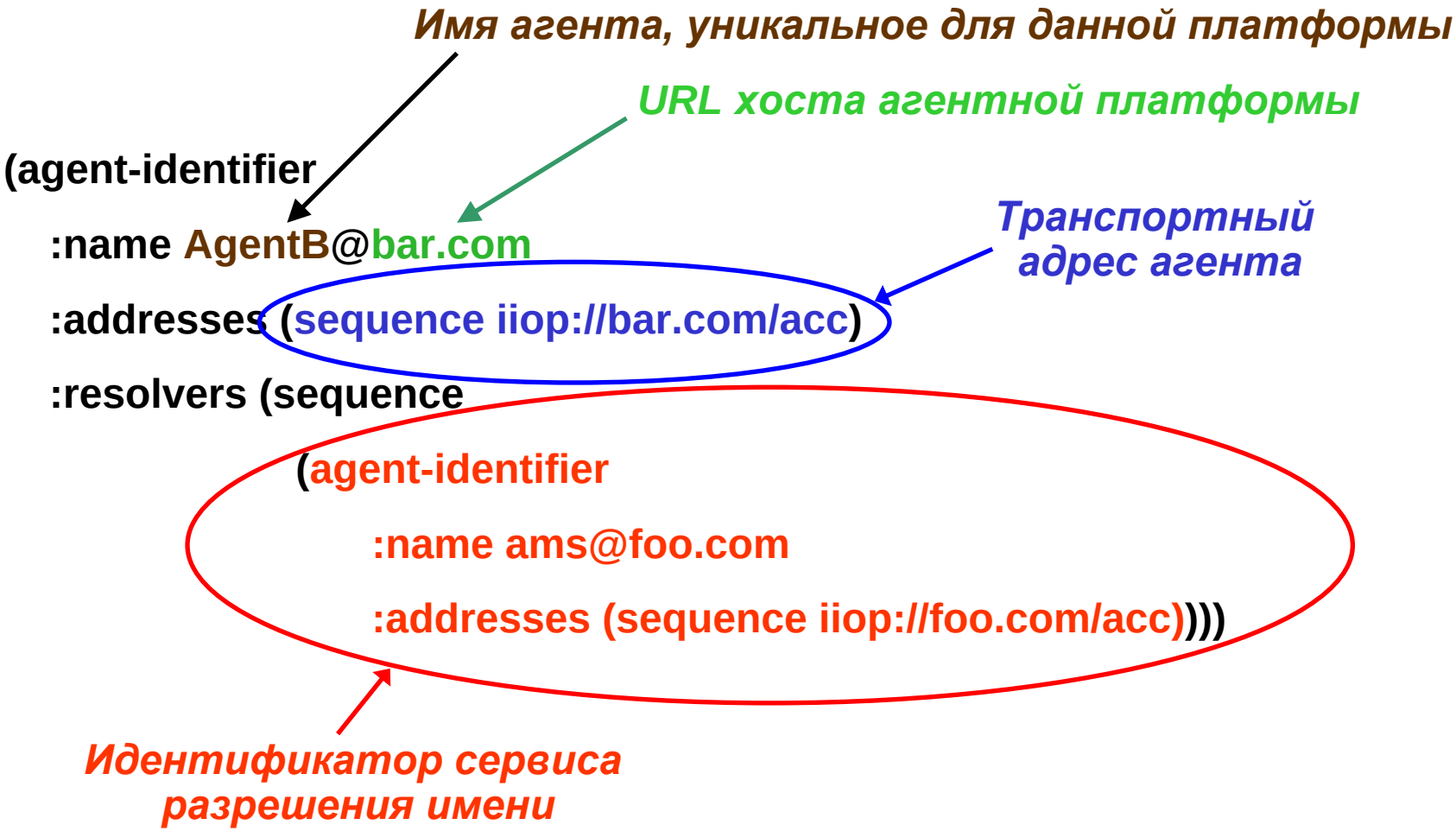
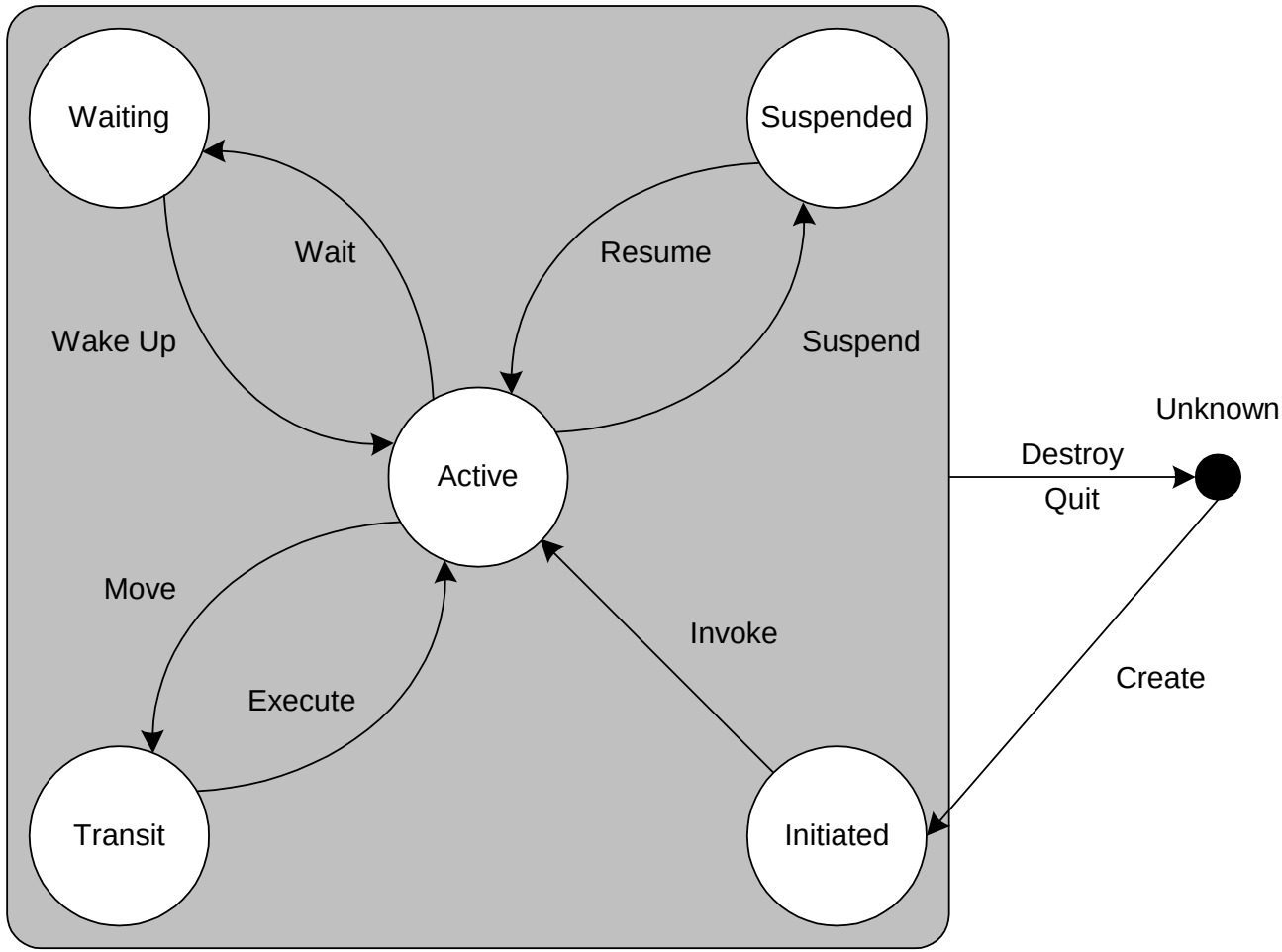


Диаграмма жизненного цикла агента



Формат сообщения на языке FIPA-ACL

```
( <Тип сообщения>
    :sender              // Отправитель сообщения
    :receiver            // Получатель(и) сообщения
    :content             // Содержание сообщения
    :reply-with          // Метка исходящего сообщения
    :in-reply-to         // Ссылка на входящее сообщение
    :replyBy             // Лимит времени на ответ
    :language            // Язык сообщения
    :ontology            // Онтология
    :protocol            // Используемый протокол общения
    :conversation-id     // Идентификатор разговора
)
```

Пример сообщения на языке FIPA-ACL

```
(inform
   :sender agent1
   :receiver agent5
   :content (price good200 150)
   :language sl
   :ontology hpl-auction
)
```

Пример сообщения на языке FIPA-ACL

```
(REQUEST
    :sender (agent-identifier :name SendAg@host:1099/JADE)
    :receiver (set (agent-identifier :name RecvAg@host:1099/JADE))
    :content "((action
                (agent-identifier :name RecvAg@host:1099/JADE)
                (REGISTER :student (STUDENT :name Ivanov
                                            :groupnumber \"8307\")
                                  :course (COURSE :name \"MAS Course\"
                                  :instructor (INSTRUCTOR :name Panteleev
                                                          :dept CS)))))"
    :language  fipa-sl0
    :ontology  eLearning
)
```

FIPA-ACL: “Inform” и “Request”

- “Inform” и “Request” - два базовых речевых акта (performatives) в FIPA ACL.
  - Все остальные являются *макро* определениями, определенными в терминах этих речевых акта

- Значение “Inform” и “Request” определяется в двух частях:
  - предусловия, которые должны быть истинны для того, чтобы речевой акт достигал цели;
  - “рациональный эффект “(“rational effect”) - чего надеется достичь отправитель сообщения.

Речевой акт «inform»

- Содержание представляет собой *утверждение* (*statement*).
- Предусловие заключается в том, что отправитель:
  - считает, что содержание является истинным;
  - имеет намерение, чтобы получатель поверил в это содержание;
  - еще не считает, что получатель осведомлен о том, является ли содержание истинным или нет.

Речевой акт «request»

- Содержание представляет собой *действие* (*action*).
- Предусловие заключается в том, что отправитель:
  - имеет намерение, чтобы было выполнено действие, указанное в содержании;
  - считает, что получатель способен выполнить это действие;
  - не считает, что получатель уже намеревается выполнить действие.

Формальная семантика коммуникативных актов

Семантика коммуникативного акта (КА) *inform*

Агент *i* информирует агента *j* о содержании ϕ :

$< i, inform(j, \phi)>$

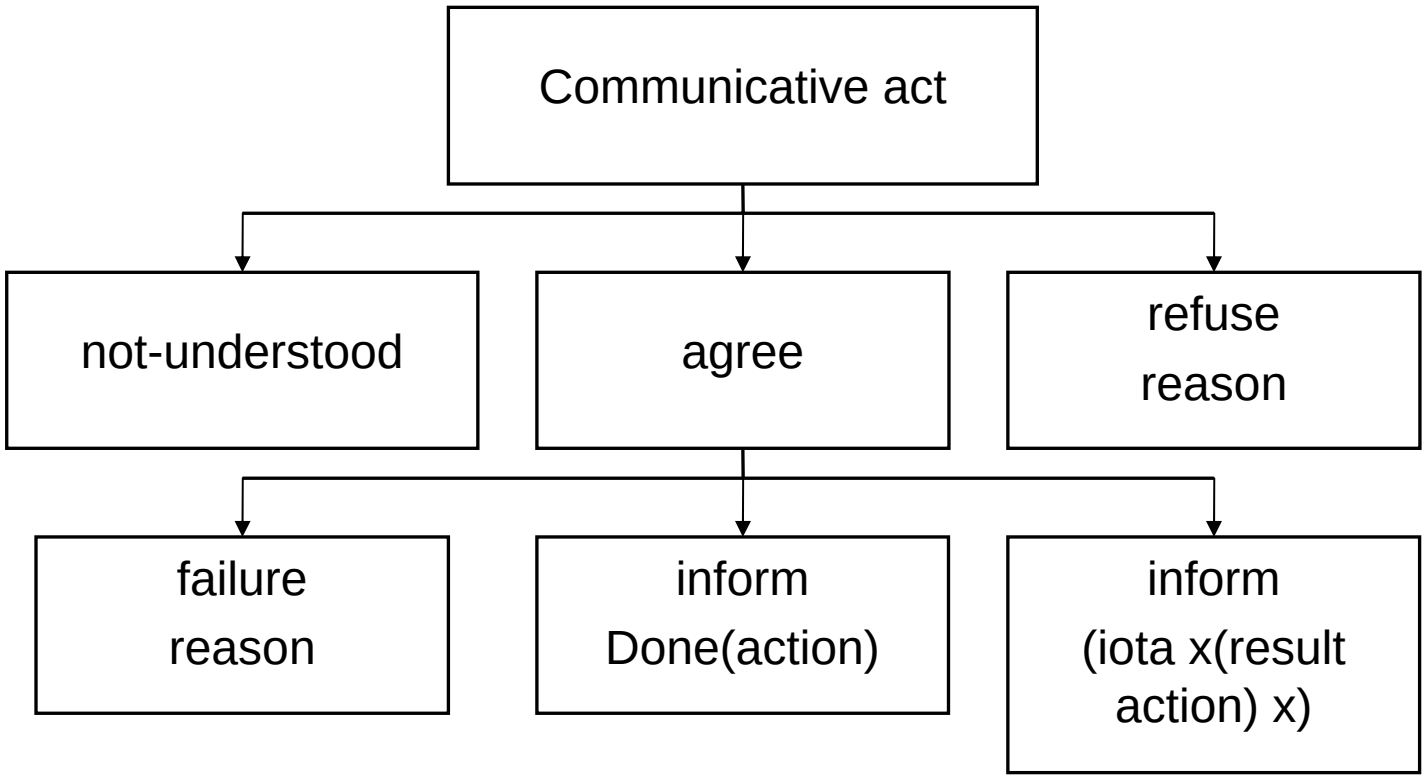
FP: $B_i (\phi) \wedge \neg B_i (B_i f_j (\phi) \vee U_i f_j (\phi))$

RE: $B_j (\phi)$

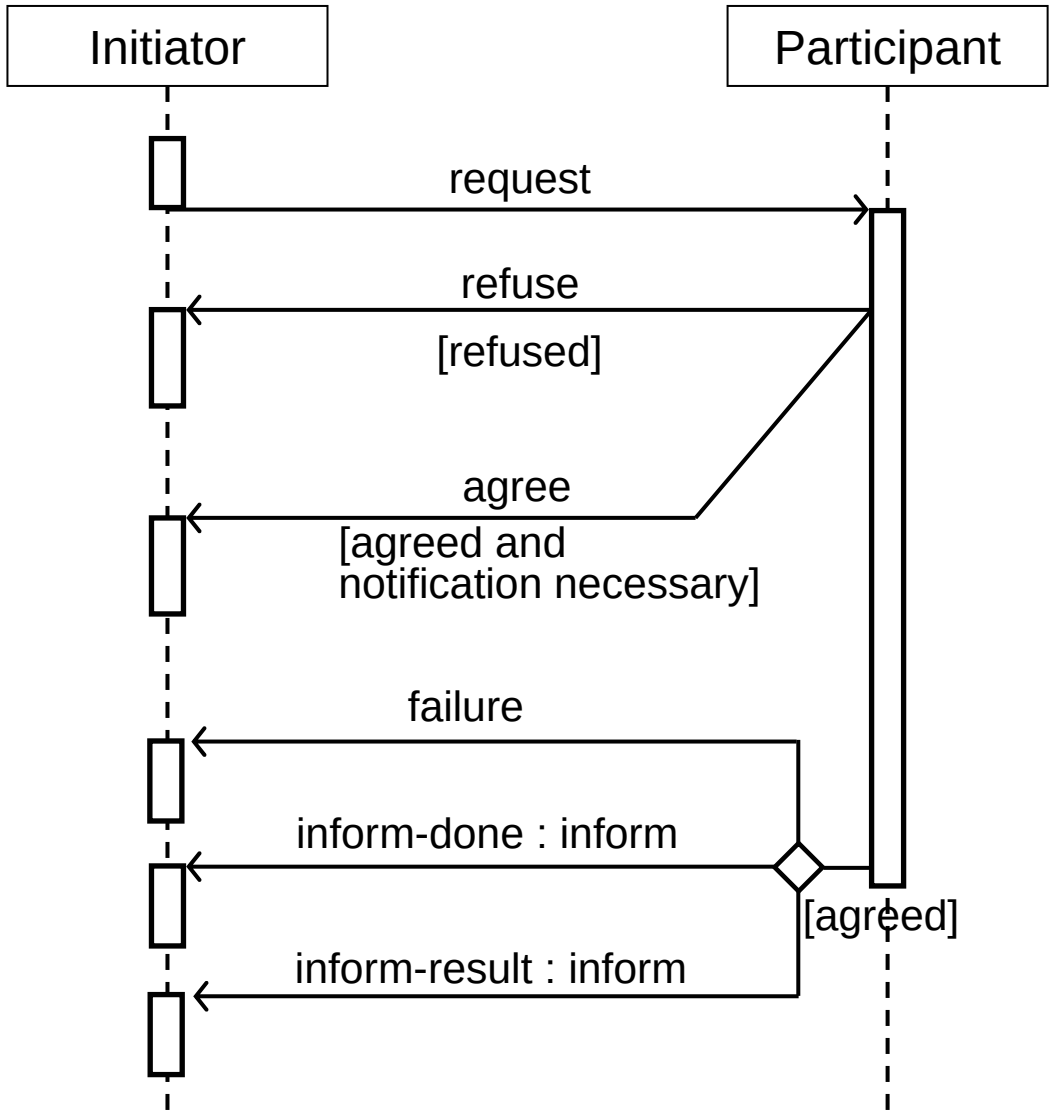
FP (feasibility preconditions) **-** предусловия выполнимости - описывают необходимые условия для отправителя КА.

RE (rational effect) **-** предполагаемый эффект сообщения

Общая структура протоколов взаимодействия



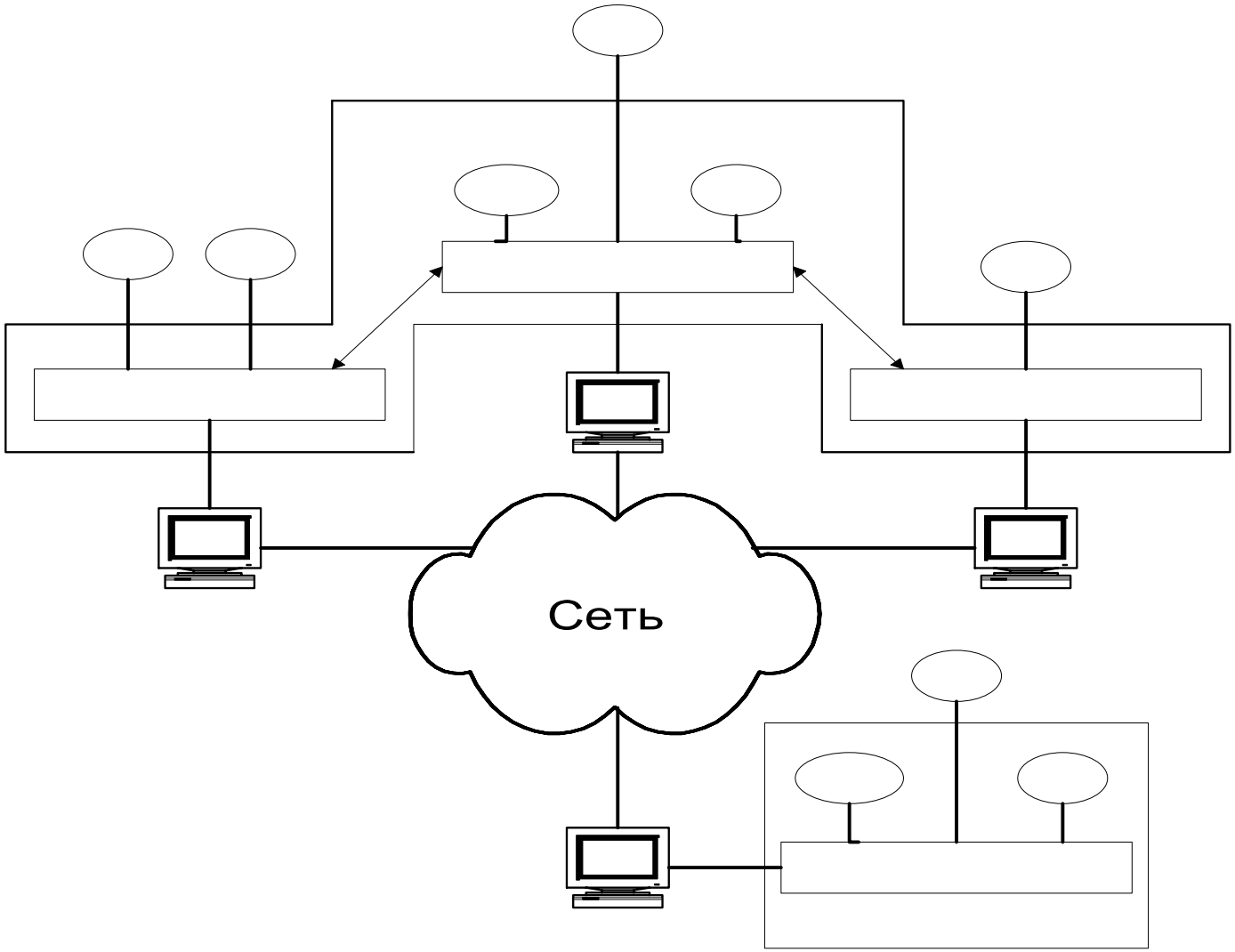
FIPA-Request-Protocol



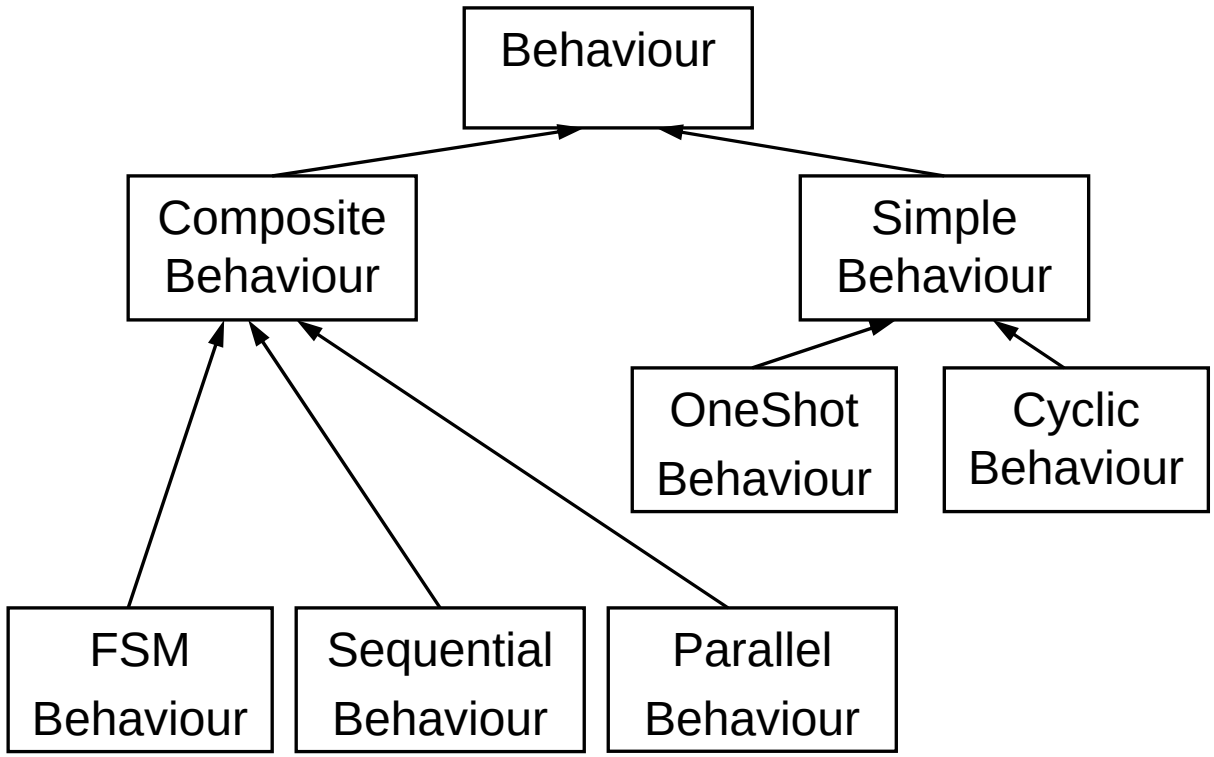
FIPA-совместимые агентные платформы:

- JADE – (http://sharon.cselt.it/projects/jade)
- FIPA-OS – (http://www.nortelnetworks.com/fipa-os)
- Zeus – (http://innovate.bt.com/projects/agents.htm)
- . . .

Платформы и контейнеры в JADE



Иерархия классов Behaviour



Пример создания и отправки сообщения

- Оправка сообщения другому агенту реализуется простым заполнением полей объекта **ACLMessage** и последующим вызовом метода **send()** класса **Agent:**

- Пример:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);

msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));

msg.setLanguage("English");

msg.setOntology("Weather-forecast-ontology");

msg.setContent("Today it's raining");

send(msg);
```

Посылающий агент информирует получателя с именем *Peter,* что *сегодня идет дождь*.