Анализ и проектирование на UML

Новиков Федор Александрович

fedornovikov@rambler.ru

Курс подготовлен по заказу
ООО Сан Майкросистемс СПб

Часть 7

Курс подготовлен при поддержке Sun Microsystems

План лекций

- Введение в UML
- Обзор языка
- Моделирование использования
- Моделирование структуры
- Моделирование поведения
- Управление моделями
- ✓ Тенденции развития языка
- UML и процесс разработки

7. Тенденции развития UML

- 7.1. Средства визуального моделирования и спецификации
- 7.2. Изменение контекста применения
- 7.3. Модельно центрированная разработка (MDA)
- 7.4. Новые средства в UML 2.0

7.1. Средства визуального моделирования и спецификации

- Визуальное конструирование и спецификация начались не с UML и на нем не закончатся
- Методология структурного анализа Росса (SADT) и семейство стандартов IDEF
- Моделирование данных, модель «сущность-связь» Чена и диаграммы ERD
- Поведение встроенных систем, диаграммы Харела и язык SDL

Методология структурного анализа

- SADT (Structured Analysis and Design Technique) одна из самых известных и широко используемых методологий структурного анализа и проектирования сложных систем
- Разработана Дугласом Т. Россом (D. T. Ross) в конце 1960х - начале 1970х годов изначально для аэрокосмической и военной промышленности
- Имеет широчайшее распространение во многих областях

Основные принципы SADT

- Система — совокупность взаимодействующих компонентов и взаимосвязей между ними
- Моделирование — процесс создания точного описания системы
- Модель (SADT-модель) — полное и точное описание системы с помощью SADT, которое может быть использовано для получения ответов на вопросы относительно системы с заданной точностью

Способ описания модели

- Фиксация цели и точки зрения
- Функциональные блоки
- Вход (слева), выход (справа), механизм (снизу), управление (сверху)
- Структурная декомпозиция

- Неформально
- Один блок – одна функция (квадратик)
- Выходы блока – стрелки на входы, механизмы и управление других блоков
- Каждый блок может быть раскрыт в своей диаграмме

Цель и точка зрения модели

Вопросы:
Каковы обязанности архитектора?
Каковы обязанности программиста?
Каковы обязанности тестера?
Кто контролирует выдачу и прохождение заданий?
Как обеспечивается управление конфигурацией?
На каких стадиях требуются диаграммы?
Как производится обработка дефектов?

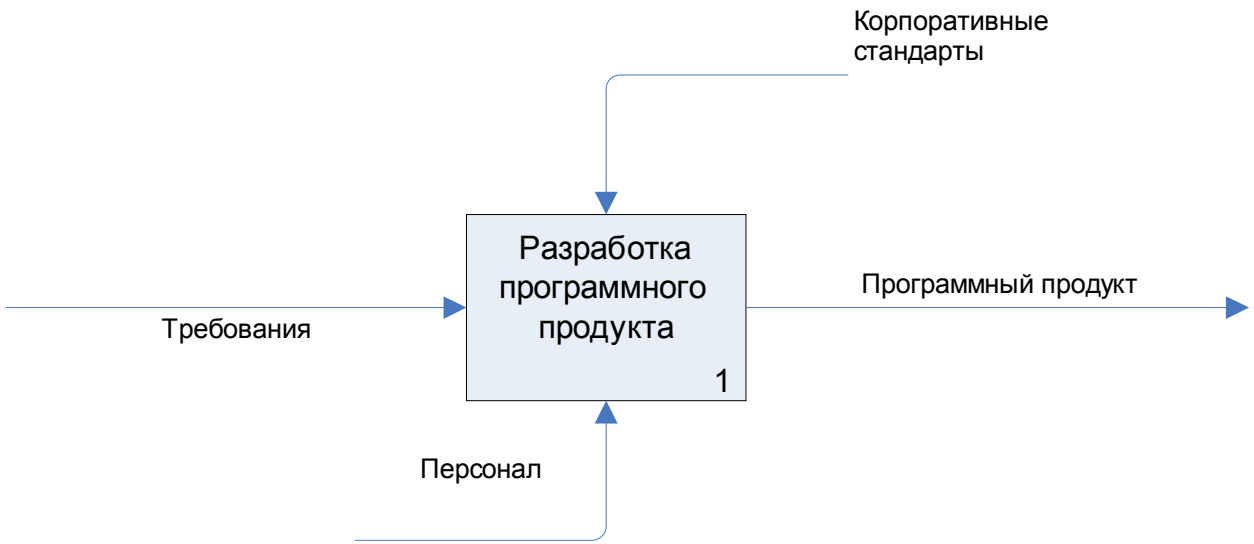
Цель:
Определить обязанности каждого сотрудника лаборатории АВР и понять, как эти обязанности связаны между собой, с тем чтобы написать должностные инструкции

Претенденты:
Главный архитектор
Технический лидер
Уполномоченный по качеству
Заведующий лабораторией

Точка зрения:
Заведующий лабораторией

NODE: АВРА-0	TITLE: Цель и точка зрения модели АВР	NO.: ФАН 001

Функция верхнего уровня

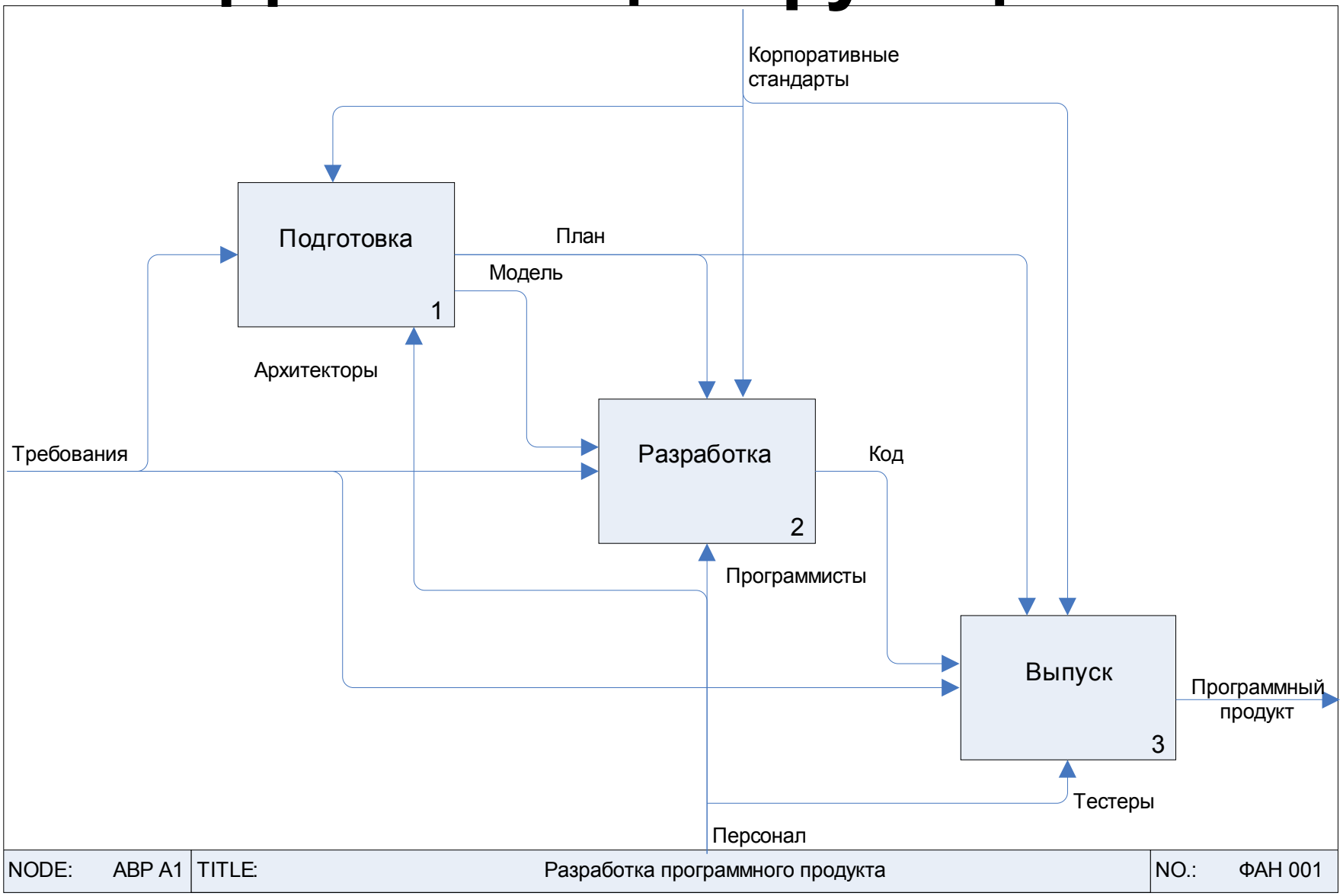


Цель:
Понять, какие функции должны быть включены в процесс изготовления программного продукта и как эти функции взаимосвязаны между собой с тем чтобы написать должностные инструкции

Точка зрения заведующий лабораторией

NODE: ABP A0	TITLE: Разработка программного продукта	NO.: ФАН 001

Детализация функции



Общие замечания

- На основе идей структурного анализа разработано семейство стандартов IDEF
- IDEF0 – функциональная декомпозиция
  - Предыдущие три слайда
  - + множество полезных мелких деталей
- IDEF3 – взаимодействие процессов
  - Process Flow Description Diagrams
  - Object State Transition Network
- ≈ Диаграммы деятельности UML
  - Без потери семантики
  - С потерей структуры и прагматики

Модель «сущность-связь»

- Моделирование данных
- Модель Чена и ее вариации
- Диаграммы «сущность-связь»
  - В оригинале: Entity Relation Diagram (ERD)
  - Традиционно: «диаграмма сущность-связь»
  - Правильно: «диаграмма отношений сущностей»
- Выражение модели «сущность-связь» средствами UML

Моделирование данных (1)

- Реальный мир
- Модель данных
- Схема базы

- **Моделирование данных – процесс создания логического представления пользовательских требований к данным**
- **Модель данных в СУБД ≈ Модель использования в UML (≠ детальная модель структуры!)**

Моделирование данных (2)

- …
- Модель «сущность-связь»
  - Peter Chen 1976
    - E.F. Codd 1970: A relational Model of Data for Large Shared Databanks
- Семантическая объектная модель
  - E.F. Codd 1976: Extending Relational Model to Capture More Meaning
- …

Модель «сущность-связь»

- Сущность (entity)
- Атрибут (attribute ≈ property)
- Идентификатор (identifier)
- Связь (relationship)

Сущность

- Объект, видимый пользователю
- Класс сущностей, обладающих одинаковой структурой
- Экземпляр сущности – набор значений

Атрибут

- Характеристика сущности
- Все экземпляры имеют одинаковые атрибуты
- Составные атрибуты (struct) и многозначные атрибуты (array)

Идентификатор

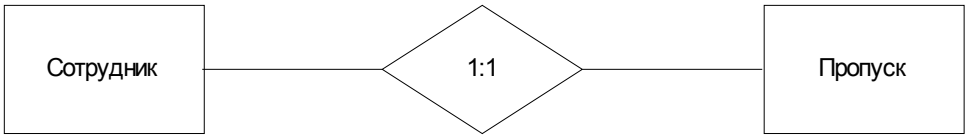
- **Идентификатор = атрибут (ы), именующий (е) экземпляр сущности**
- **Идентификатор, состоящий из нескольких атрибутов, называется *составным* (composite)**
- **Уникальный (unique) идентификатор именует один экземпляр**
- **Неуникальный идентификатор именует несколько экземпляров**

Связь

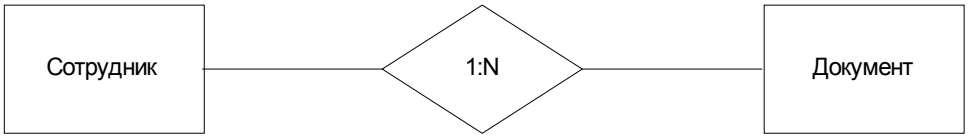
- Класс связей – отношение между классами сущностей
- Экземпляр связи – отношение между экземплярами сущностей
- Связи могут иметь атрибуты
- Связи могут связывать несколько сущностей

Типы бинарных связей

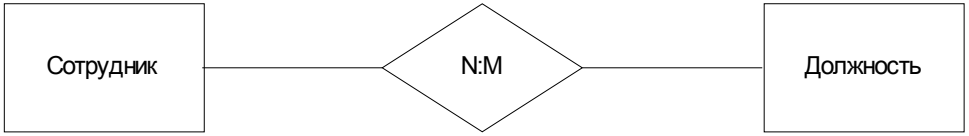
- **Один к одному**



- Один ко многим



- Многие ко многим

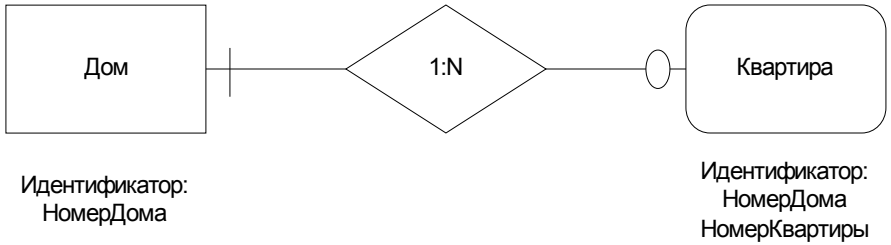


Диаграммы «сущность связь» (Entity Relation Diagram - ERD)

- Сущность – прямоугольник
- Связь – ромб
- Имя связи над или под ромбом
- Нижняя граница кратности (0 - 1) на полюсе возле сущности
- Атрибуты – овалы (привязаны к сущности или связи) или отдельный прямоугольник

Слабые и сильные сущности

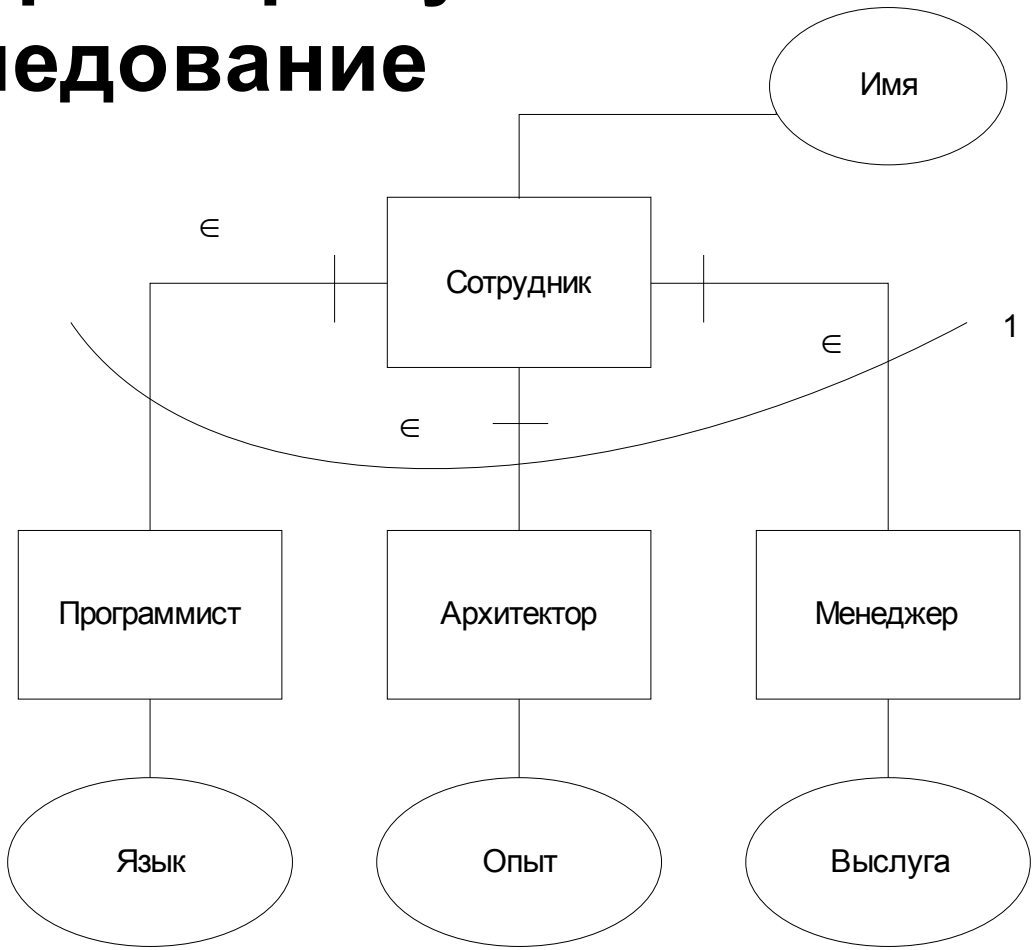
- **Слабая (weak) сущность – может существовать, только если есть другая сущность**



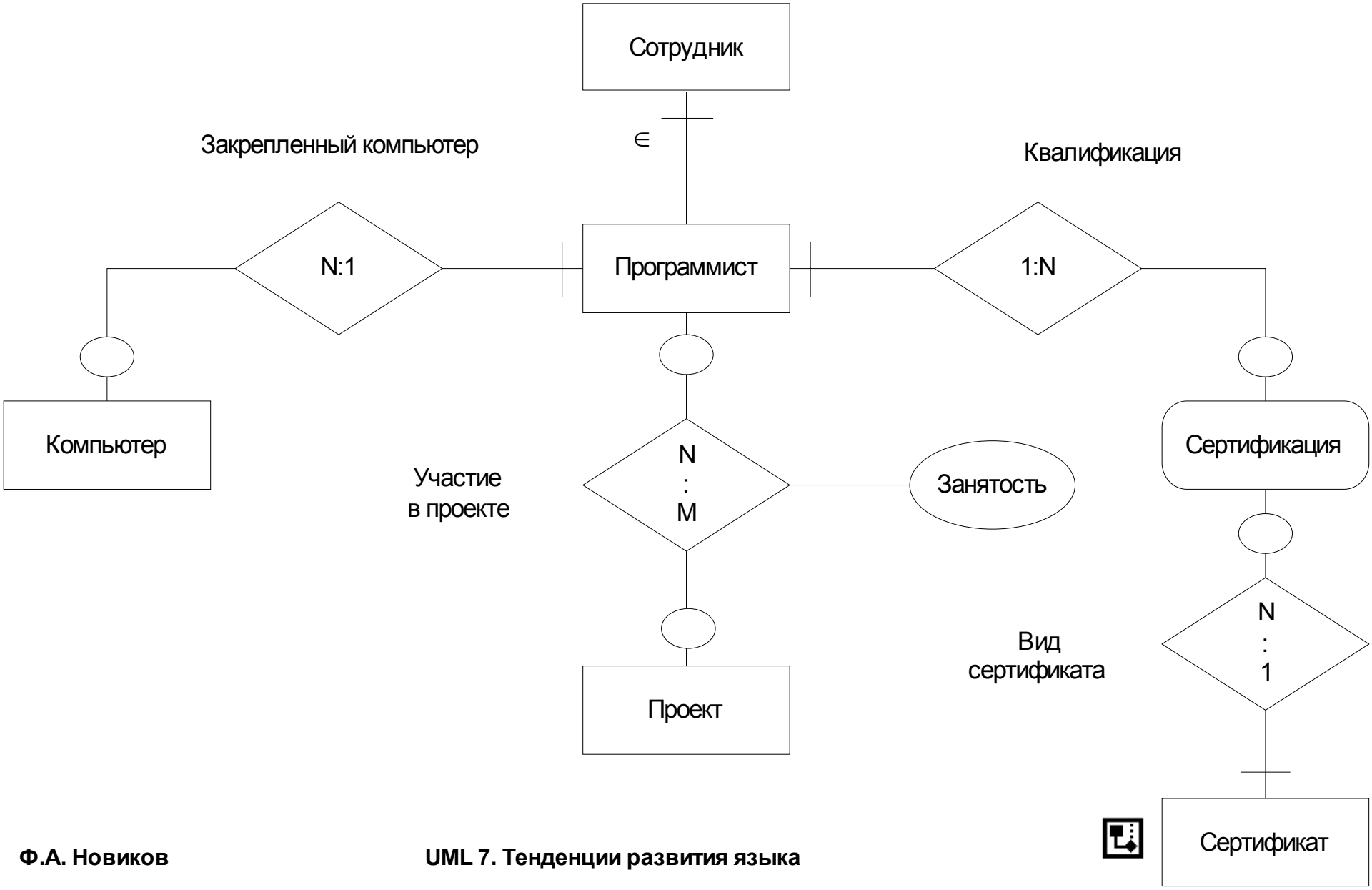
- Идентификационно-зависимая сущность – идентификатор содержит идентификатор другой сущности

Подтипы сущностей

- **Альтернативные необязательные наборы атрибутов ≈ наследование**



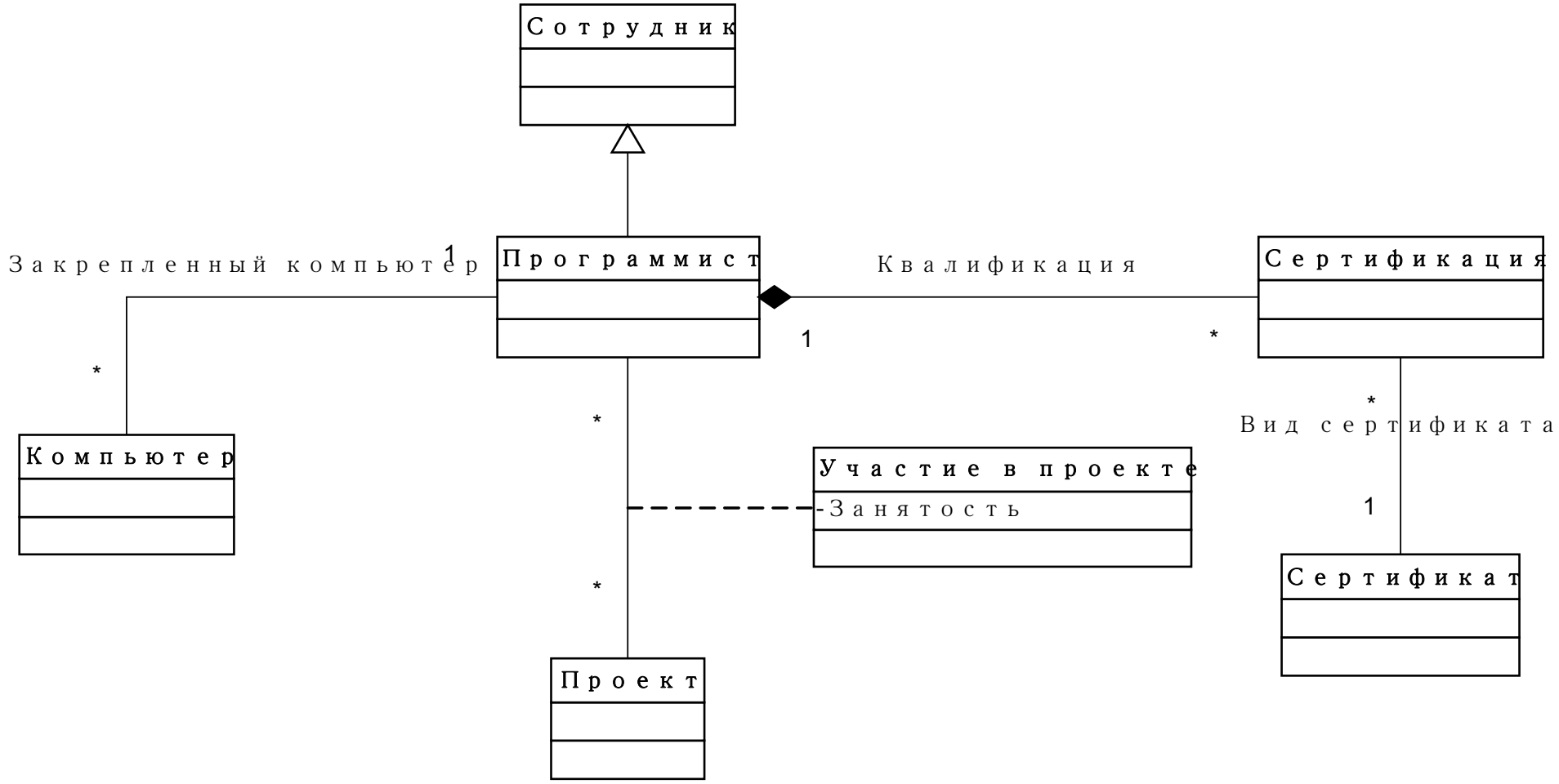
Пример модели ERD



Сущности и связи в UML

- Сущности → классы
  - стереотип «persistent»
- Атрибуты → атрибуты
- Связи → ассоциации
  - Тип (1:1, 1:N, N:M) → кратность полюса
  - Слабая не идентификационная → композиция
  - Слабая идентификационная → ассоциация
  - Атрибуты связи → класс ассоциации
- Подтипы → обобщение

Пример модели UML



Моделирование поведения конечными автоматами

- Спецификация поведения встроенных систем
- Язык SDL
- Язык MSC

Спецификация поведения встроенных систем

- Программируемые микроконтроллеры
- Mainframe 10^3 → PC 10^6 → MC 10^9
- Ограниченные ресурсы
- Реальное время
- Высокая надежность
- Жесткие протоколы
- Международная стандартизация
- → Формализованные спецификации – норма, а не роскошь!

Общие сведения

- SDL = Specification and Description Language
- MSC = Message Sequence Charts
- Самые старые, заслуженные, полезные и толковые языки графической спецификации (после блок-схем и структурного анализа)
- Активно поддерживаются CCITT (бывшей) и ITU-T (International Telecommunication Union – Telecommunication Standardization Sector)

История SDL

- Первые публикации идей – начало 80-х
- 1988 – стандарт
- 1993 – Recommendation Z100 – текущий стандарт.
- Все время выходят дополнения к стандарту, в которых учитываются новые веяния, но они в форме исправлений, а не в форме новой версии

Три «кита» SDL

- Структурная декомпозиция
  - Вирт, ~ 1970
  - Remake в UML 2.0
- Взаимодействующие параллельные процессы
  - Дейкстра, 1969
- Абстрактные типы данных
  - Гуттаг, 1978
  - Ветвь Парнас – Лисков породила ООП, это другая ветвь

Цитата из Z100: 1.1.1 Objective

The general objectives when defining SDL have been to provide a language that:

a) is easy to learn, use and interpret;

b) provides unambiguous specification for ordering and tendering;

c) may be extended to cover new developments;

d) is able to support several methodologies of system specification and design, without assuming any particular methodology.

Способы описания

- ## Абстрактная грамматика

  System :: System-name

  Block-set …

- ## Текстовая грамматика

  <System> ::= system **<system name>**

  <block> + …

  endsystem **<system name>**

- ## Графическая грамматика

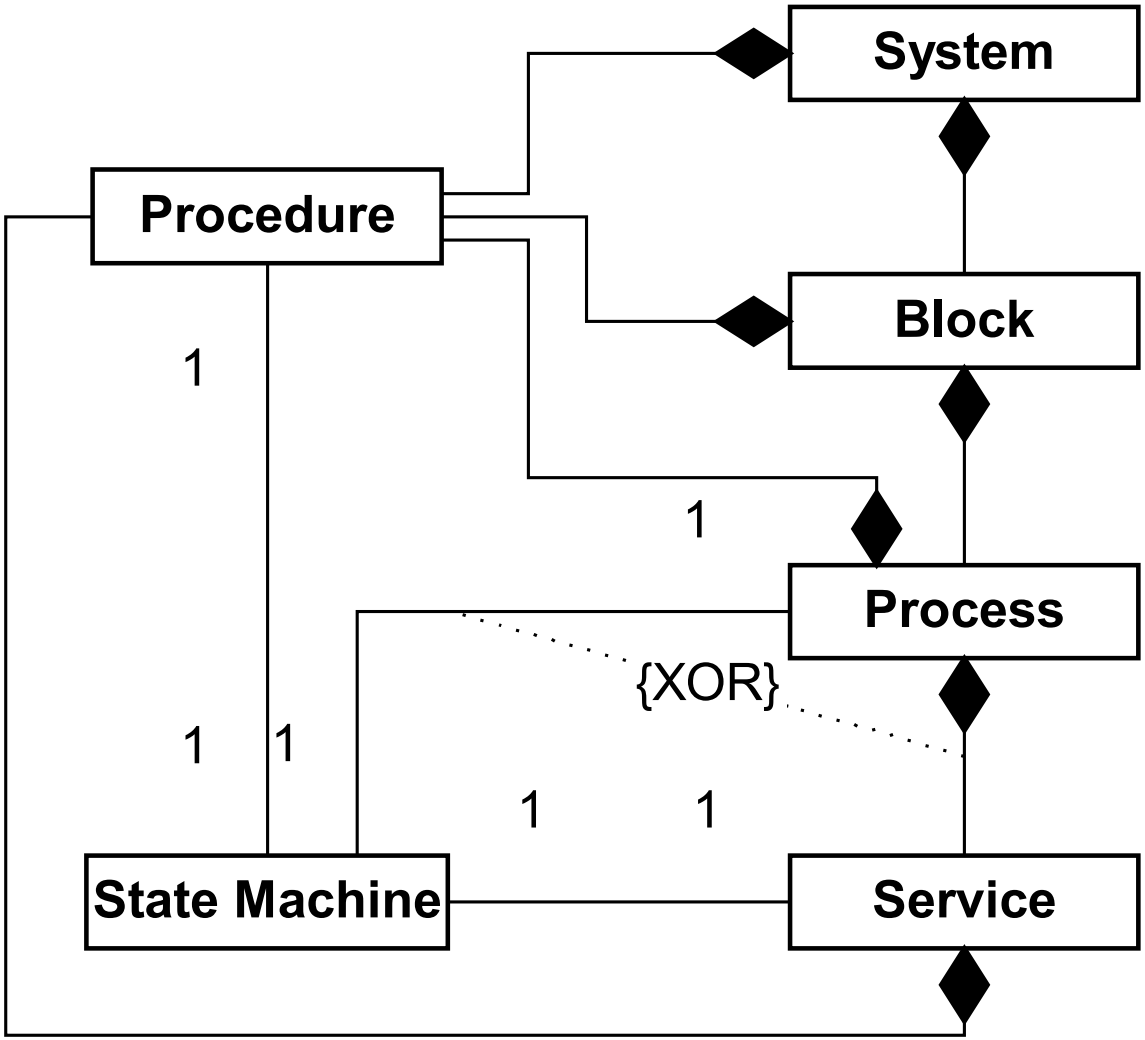
  <System diagram> ::= is associated with

  <frame symbol> contains **<system heading>**

  <block interaction area>-set **…**

  <frame symbol> :=

Структурная декомпозиция



Каналы и сигналы

- Между структурными сущностями (процедуры НЕ считаются) могут быть каналы (именованные), по которым передаются сигналы (именованные и с параметрами)
- Канал соединяет
  - либо две структурных сущности одного уровня вложенных в третью
  - либо сущность и границу (именованный gate) объемлющей сущности

Машины состояний

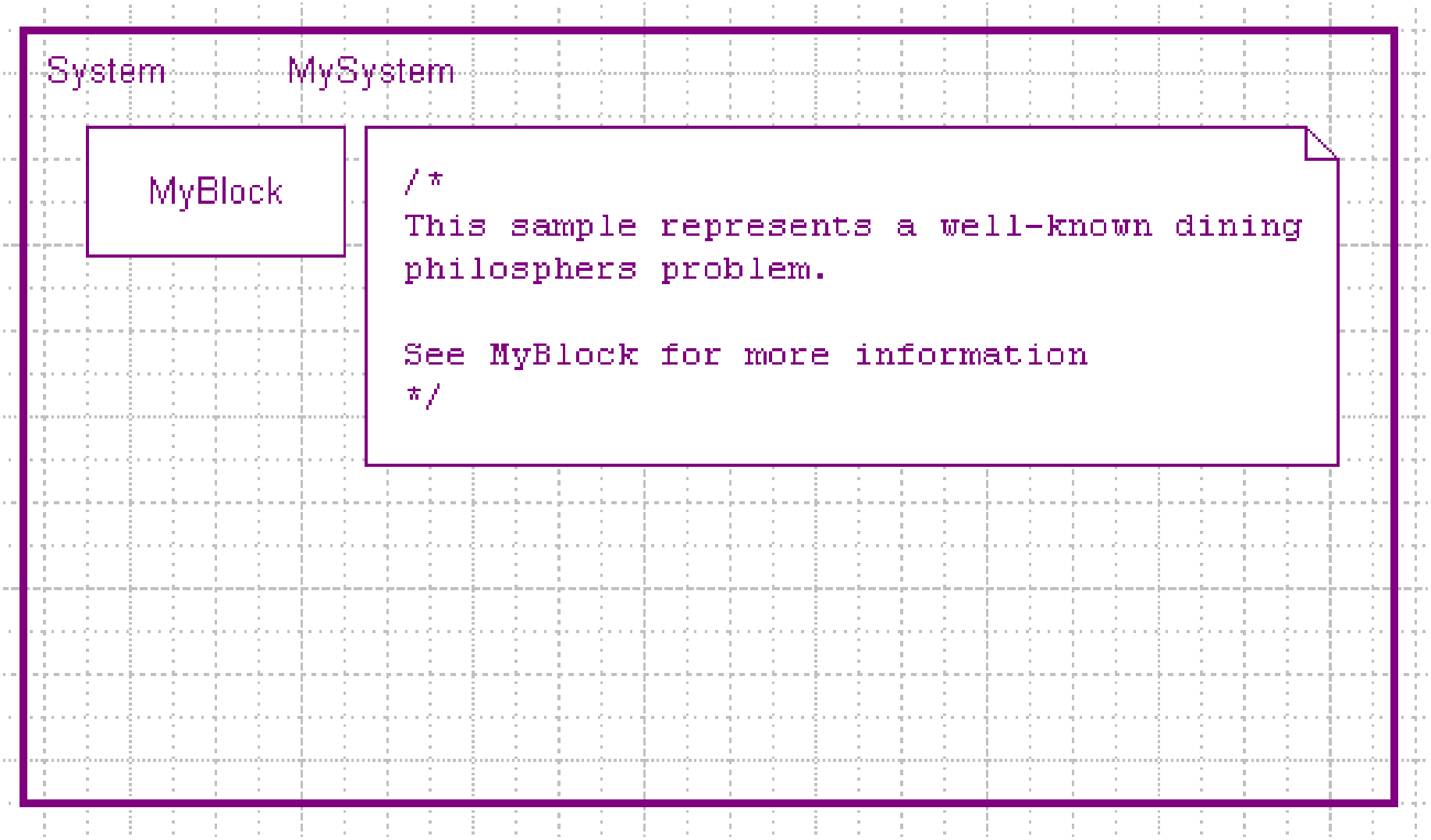
- Состояния совсем простые – внутри ничего нет, кроме имени
- Переходы тоже простые, но длинные:
  - Input (= trigger event UML)
  - Если нет input, то spontaneous transition
  - Enabling Condition (… continuous signal)
  - Action, Action, Action, …
    - Task, в том числе :=
    - Call (процедуры, которая определена выше)
    - Create (экземпляр процесса)
    - Output (= Send UML) (конкретному процессу (-ам))
    - Decision (= сегментированный переход UML)

Нотация конечных автоматов

- Состояния рисуют несколько раз (одно и то же определяется по имени)
- Переход – вертикальная линейка блоков
- Переходы слева направо, сверху вниз
- Метка (в кружочке), если не лезет в диаграмму
- Автоматически верстать диаграмму – одно удовольствие

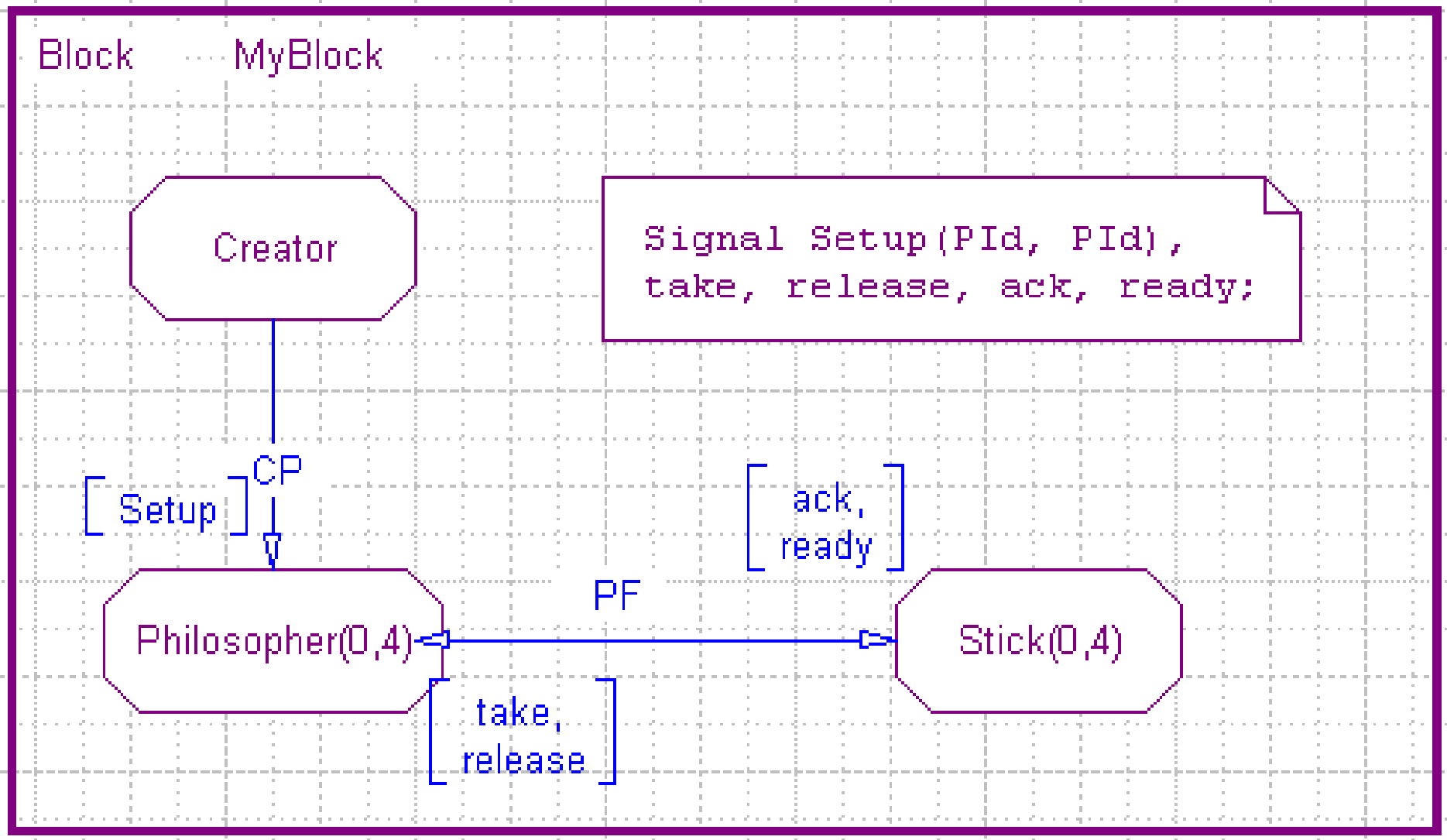
Пример – обедающие философы

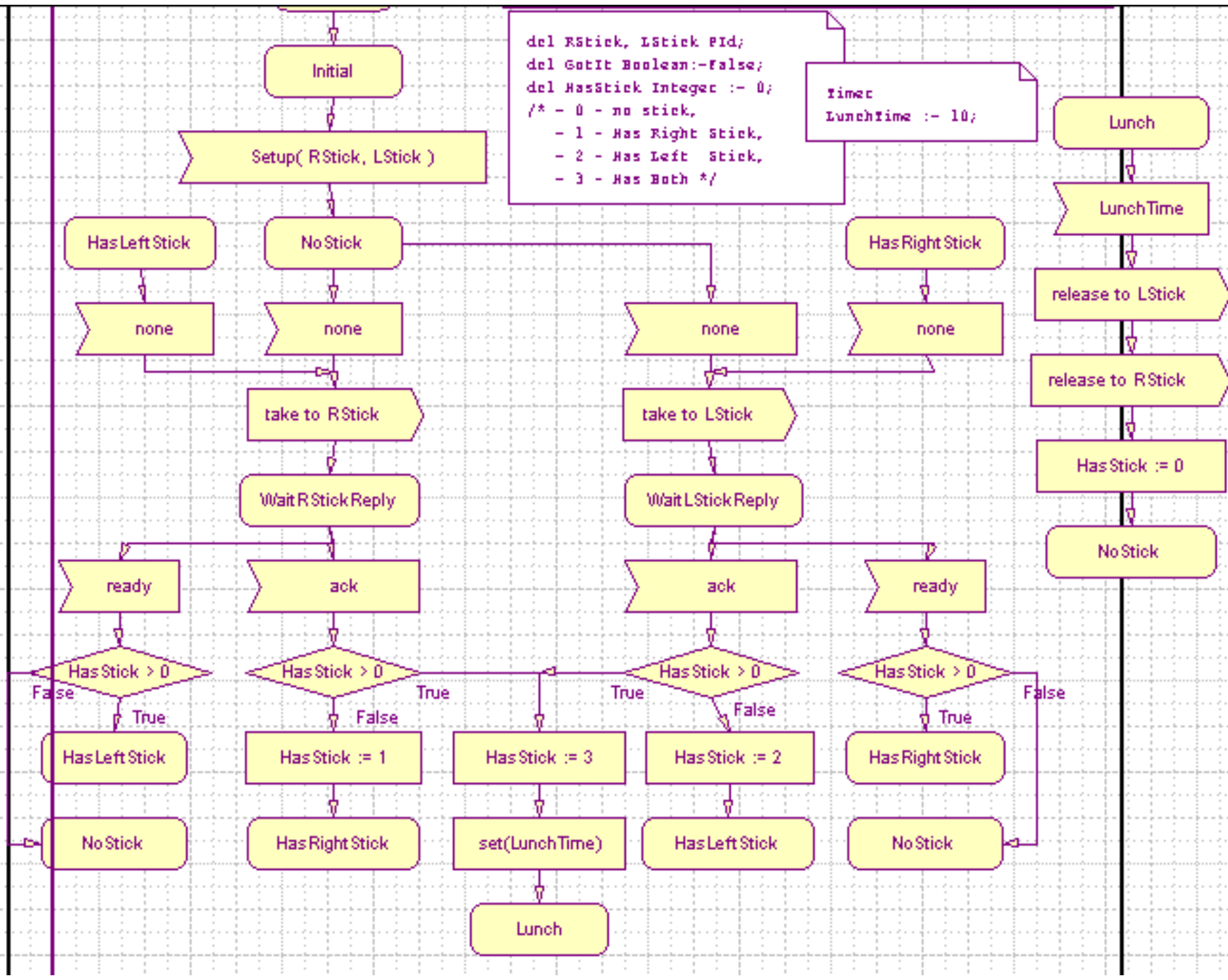
- Задача придумана Hoar'ом в 1972 году
- Философы (процессы) сидят за круглым столом
- Между ними вилки (ресурсы)
- Философ может взять вилку слева или справа, если она свободна
- Чтобы съесть спагетти нужны две вилки
- Придумать алгоритм
- Наивное решение (взять слева, взять справа и кушать) сразу приведет к тупику: у всех по вилке в левой руке и все ждут, когда освободится вилка справа
  - * Диаграммы сделаны в OSD Tool от LG Soft Lab

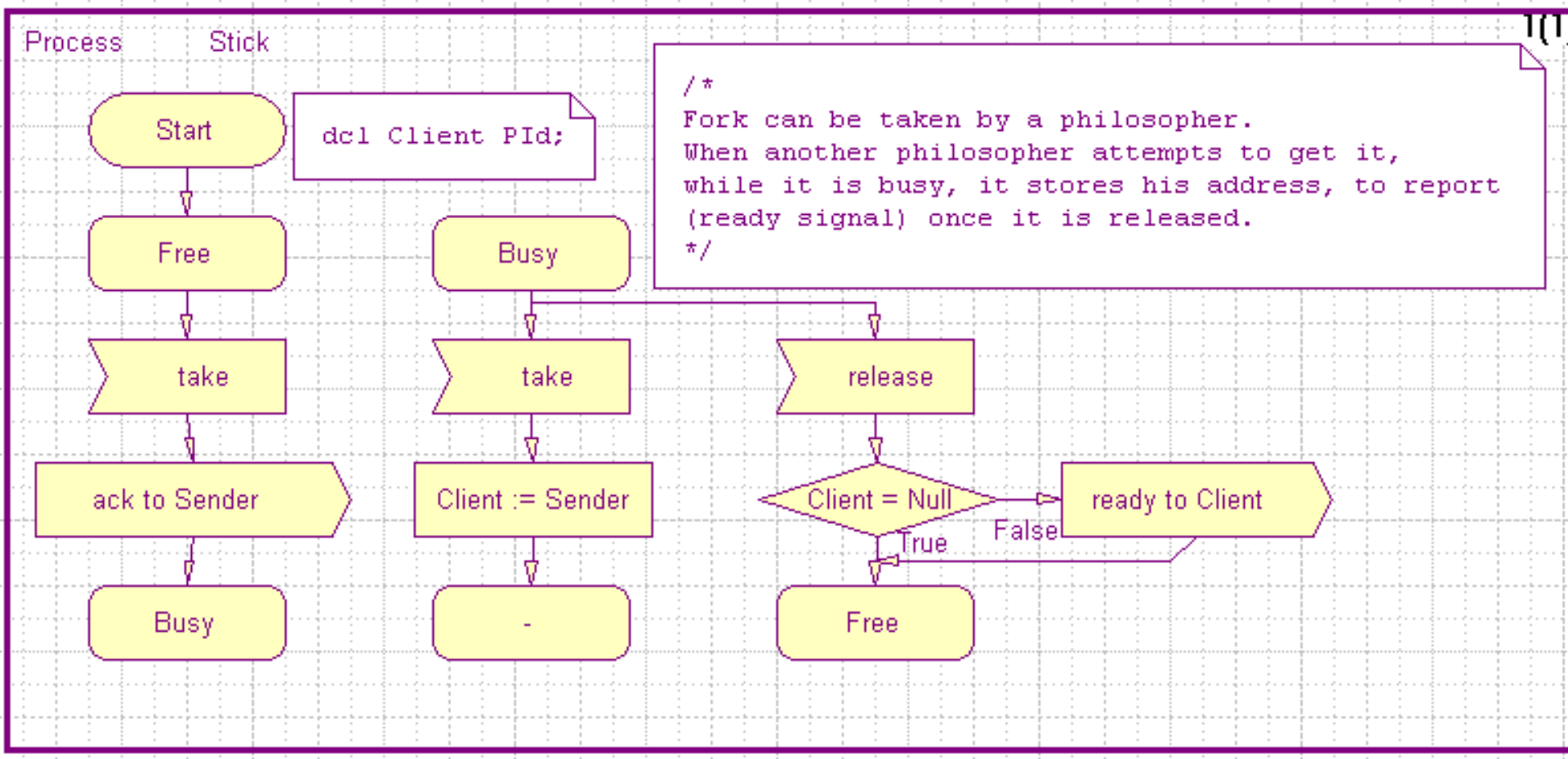

System
MySystem
MyBlock
/*
This sample represents a well-known dining philosphers problem.
See MyBlock for more information
*/

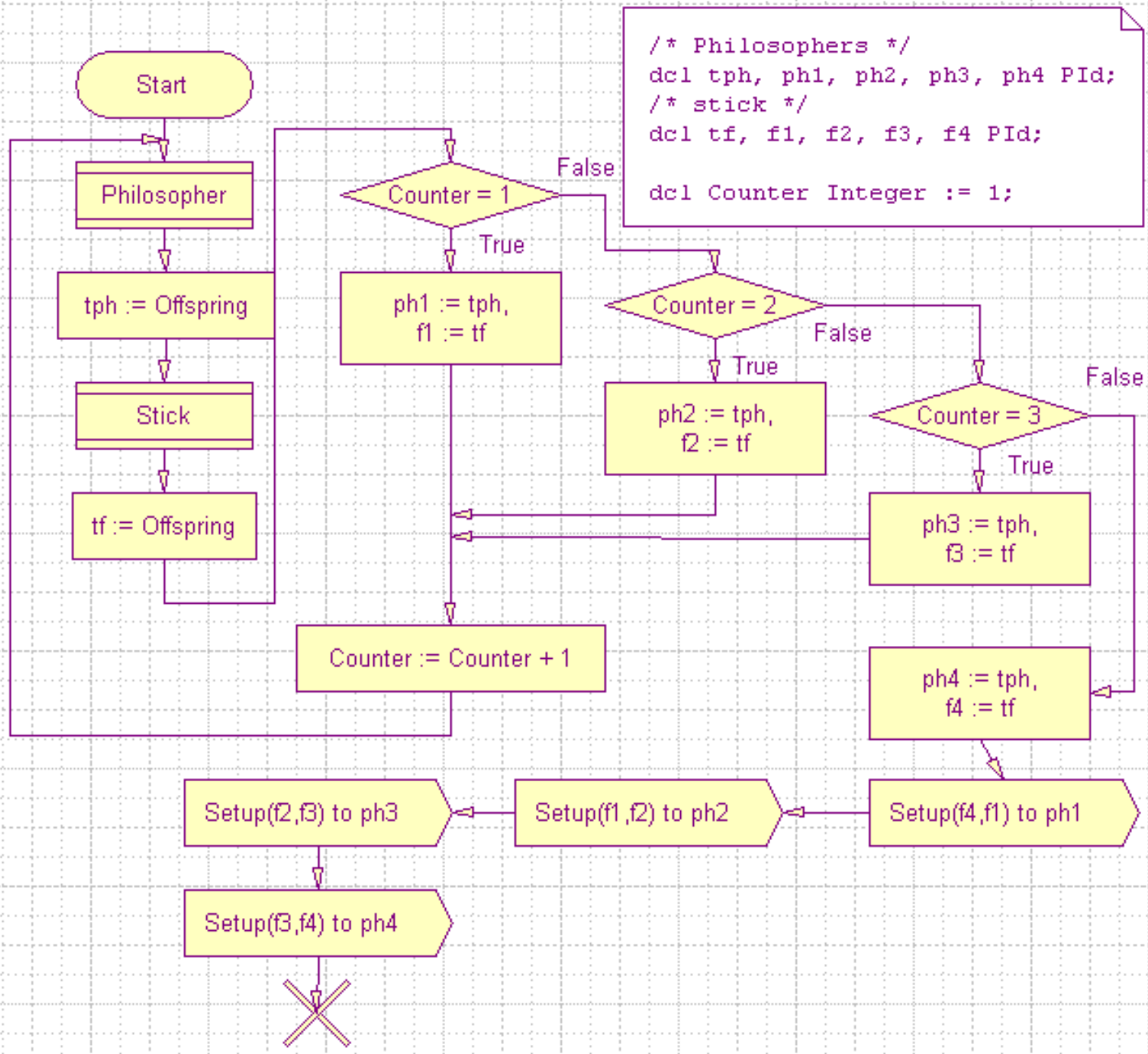
```
/*
This sample represents a well-known dining philosphers problem.
Three processes are present:
Creator - creates 4 philosofers and 4 sticks for them.
Then tells (Setup signal) each
philosopher his right and left stick location
Philosophers attempt to get two sticks each.
When a philosopher has 2 sticks he goes for lunch.
Stick can be taken by a philosopher.

The problem is - eventually each philosopher
will have exactly one stick, thus blocking others,
but not able to eat either.
*/
```


Block MyBlock
Creator
Signal Setup(PId, PId),
take, release, ack, ready;
CP
Setup
ack,
ready
PF
Philosopher(0,4)
Stick(0,4)
take,
release


dcl RStick, LStick PId;
dcl GotIt Boolean:=False;
dcl HasStick Integer := 0;
/* - 0 - no stick,
- 1 - Has Right Stick,
- 2 - Has Left Stick,
- 3 - Has Both */
Timer
LunchTime := 10;
Initial
Setup(RStick, LStick)
Has Left Stick
No Stick
Has Right Stick
none
none
none
none
take to RStick
take to LStick
Wait R Stick Reply
Wait L Stick Reply
ready
ack
ack
ready
Has Stick > 0
Has Stick > 0
Has Stick > 0
Has Stick > 0
False
True
True
False
True
False
True
False
Has Left Stick
Has Stick := 1
Has Stick := 3
Has Stick := 2
Has Right Stick
No Stick
Has Right Stick
set(LunchTime)
Has Left Stick
No Stick
Lunch
Lunch
LunchTime
release to LStick
release to RStick
Has Stick := 0
No Stick


Process
Stick
Start
dcl Client PId;
/*
Fork can be taken by a philosopher.
When another philosopher attempts to get it,
while it is busy, it stores his address, to report
(ready signal) once it is released.
*/
Free
Busy
take
take
release
ack to Sender
Client := Sender
Client = Null
ready to Client
True
False
Busy
-
Free


/* Philosophers */
dcl tph, ph1, ph2, ph3, ph4 PId;
/* stick */
dcl tf, f1, f2, f3, f4 PId;
dcl Counter Integer := 1;
Start
Philosopher
tph := Offspring
Stick
tf := Offspring
Counter = 1
False
True
ph1 := tph,
f1 := tf
Counter = 2
False
True
ph2 := tph,
f2 := tf
Counter = 3
False
True
ph3 := tph,
f3 := tf
ph4 := tph,
f4 := tf
Counter := Counter + 1
Setup(f4,f1) to ph1
Setup(f1,f2) to ph2
Setup(f2,f3) to ph3
Setup(f3,f4) to ph4

Абстрактные типы данных

- Множество носителей
- Множество операций
- Множество аксиом
- Инициальная алгебра – минимальная подалгебра свободной алгебры термов, удовлетворяющих аксиомам
- В OSD Tool никаких аксиом нет
- Зато операции можно задать внешним COM объектом

Тип Boolean (1)

- newtype **Boolean**
- literals **True, False;**
- operators
- "not" **: Boolean -> Boolean;**
- **/***
- **"=" : Boolean, Boolean -> Boolean; The "=" and "/=" operators**
- **"/=" : Boolean, Boolean -> Boolean; are implied. See § 5.3.1.4**
- */
- "and" **: Boolean, Boolean -> Boolean;**
- "or" **: Boolean, Boolean -> Boolean;**
- "xor" **: Boolean, Boolean -> Boolean;**
- **"=>" : Boolean, Boolean -> Boolean;**

Тип Boolean (2)

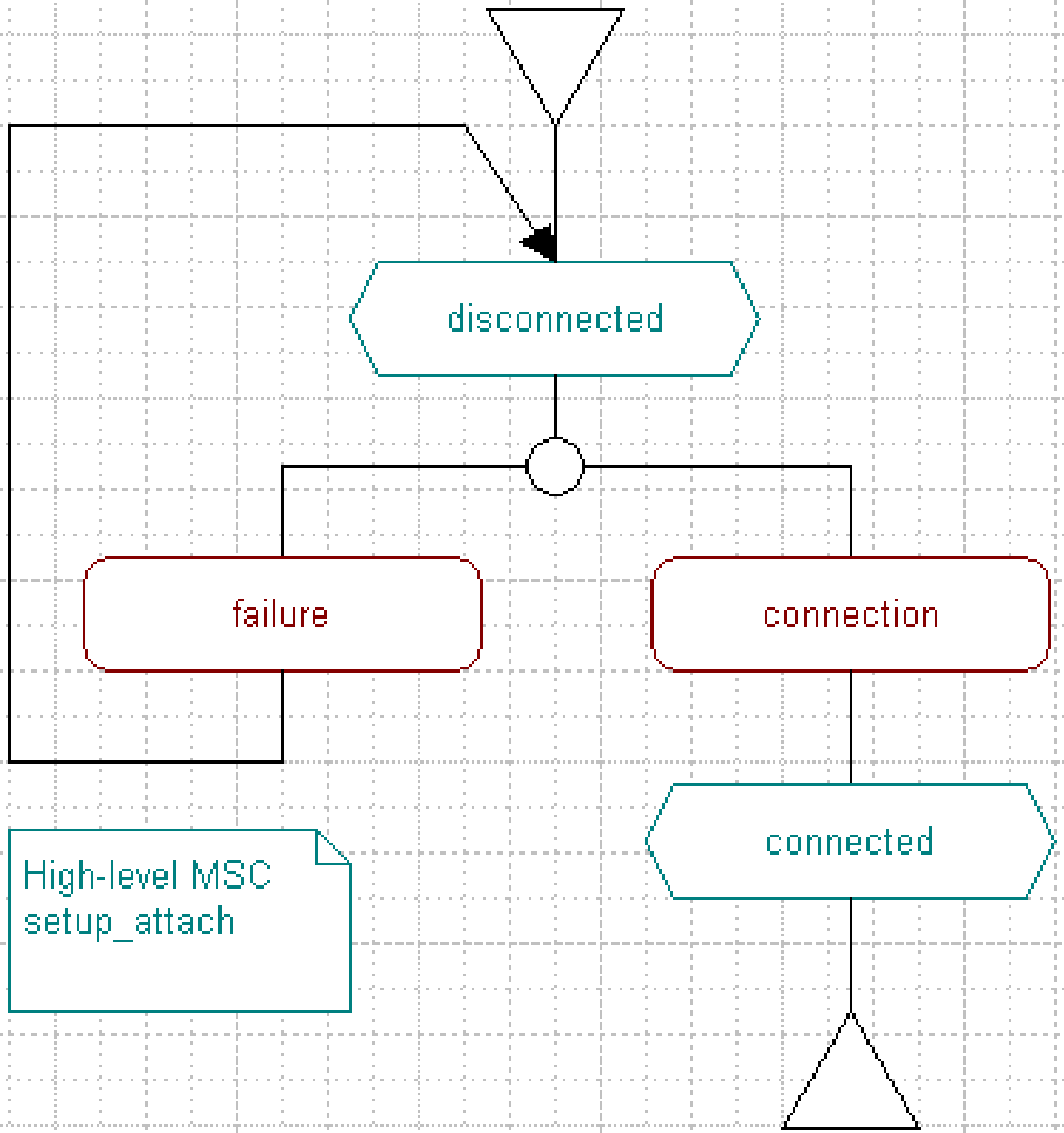
- axioms
- not **(True) = = False;**
- not **(False) = = True;**
- **True** and **True = = True;**
- **True** and **False = = False;**
- **False** and **True = = False;**
- **False** and **False = = False;**
- **True** or **True = = True;**
- **True** or **False = = True;**
- **False** or **True = = True;**
- **False** or **False = = False;**
- **True** xor **True = = False;**
- **True** xor **False = = True;**
- **False** xor **True = = True;**
- **False** xor **False = = False;**
- **True => True = = False;**
- **True => False = = False;**
- **False => True = = True;**
- **False => False = = True;**
- endnewtype **Boolean;**

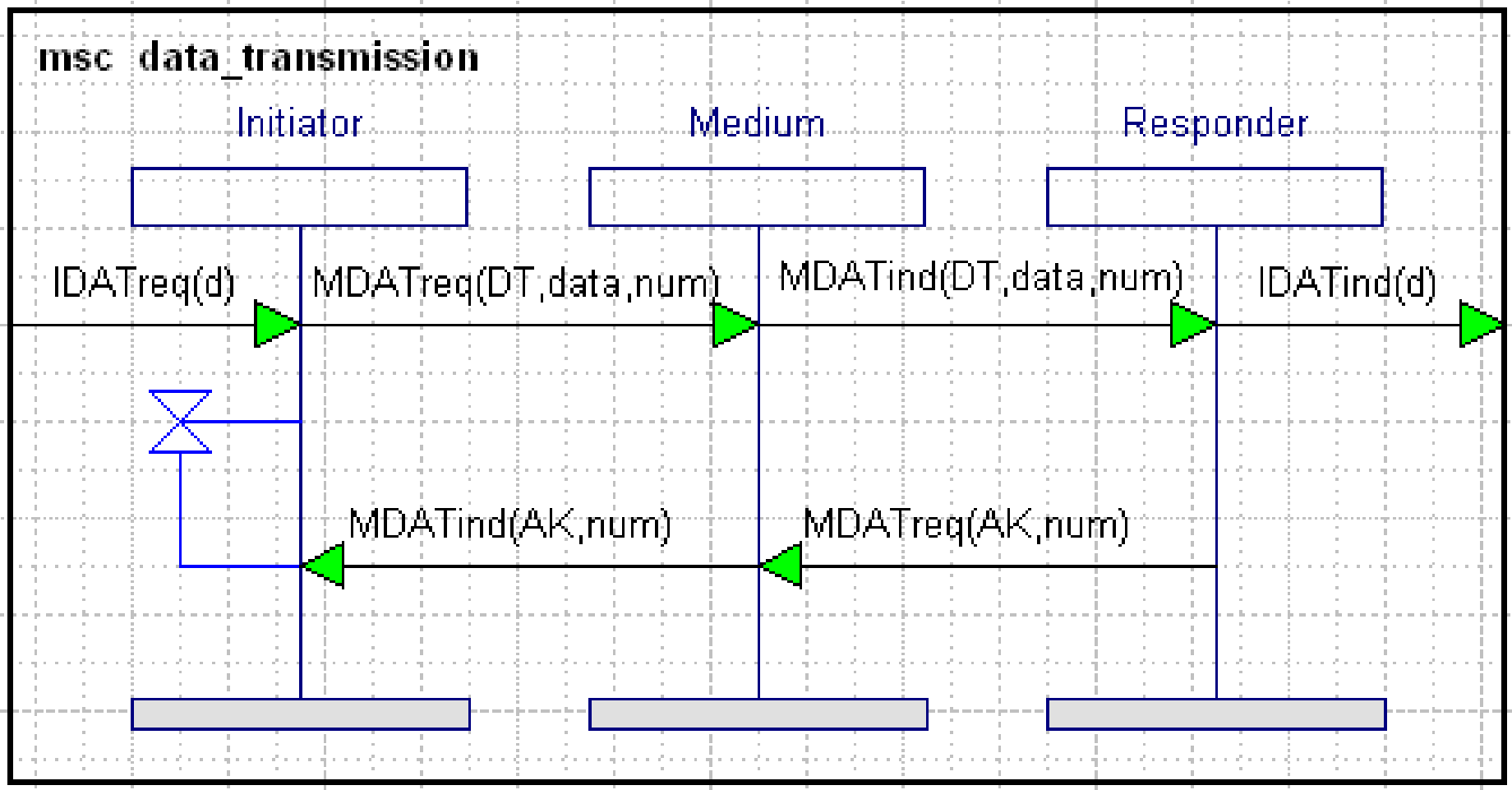
Message Sequence Chart

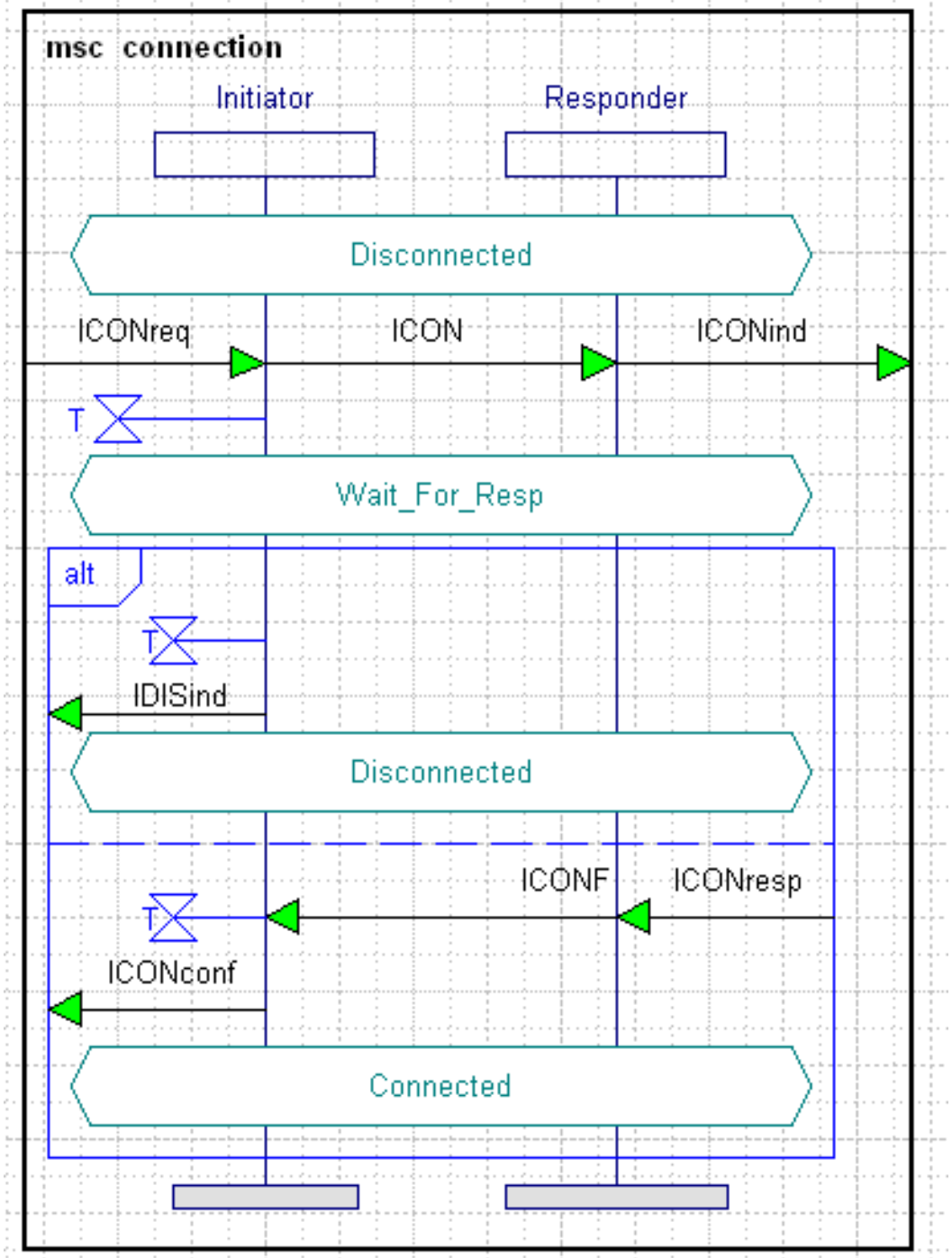
- **ITU-T Recommendation Z120, 1993-1996**
- SCOPE/OBJECTIVE
- **The purpose of recommending MSC (Message Sequence Chart) is to provide a trace language for the specification and description of the communication behavior of system components and their environment by means of message interchange. Since in MSCs the communication behavior is presented in a very intuitive and transparent manner, particularly in the graphical representation, the MSC-language is easy to learn, use and interpret. In connection with other languages, it can be used to support methodologies for system specification, design, simulation, testing, and documentation.**

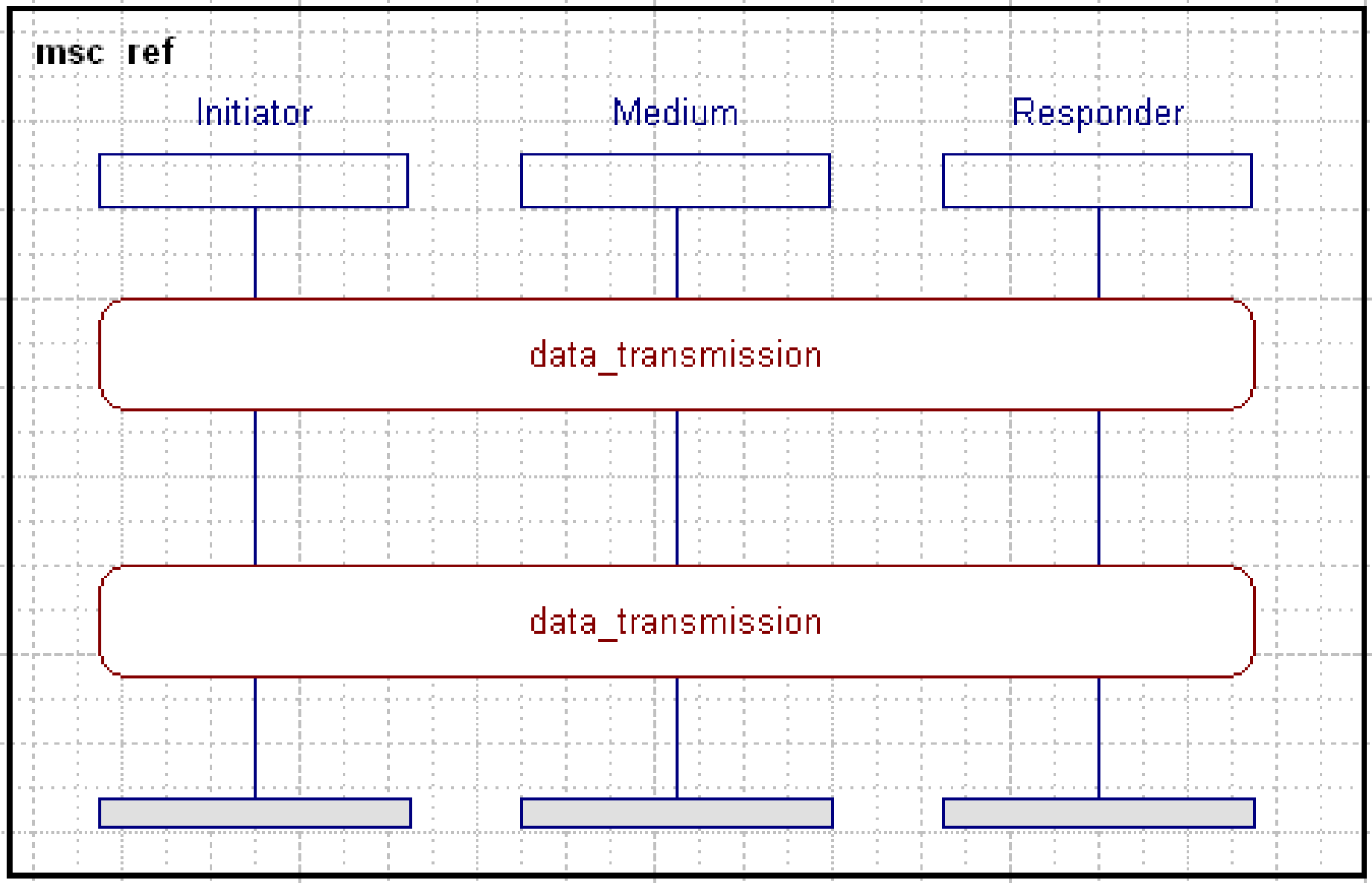
Plain MSC & High-level MSC

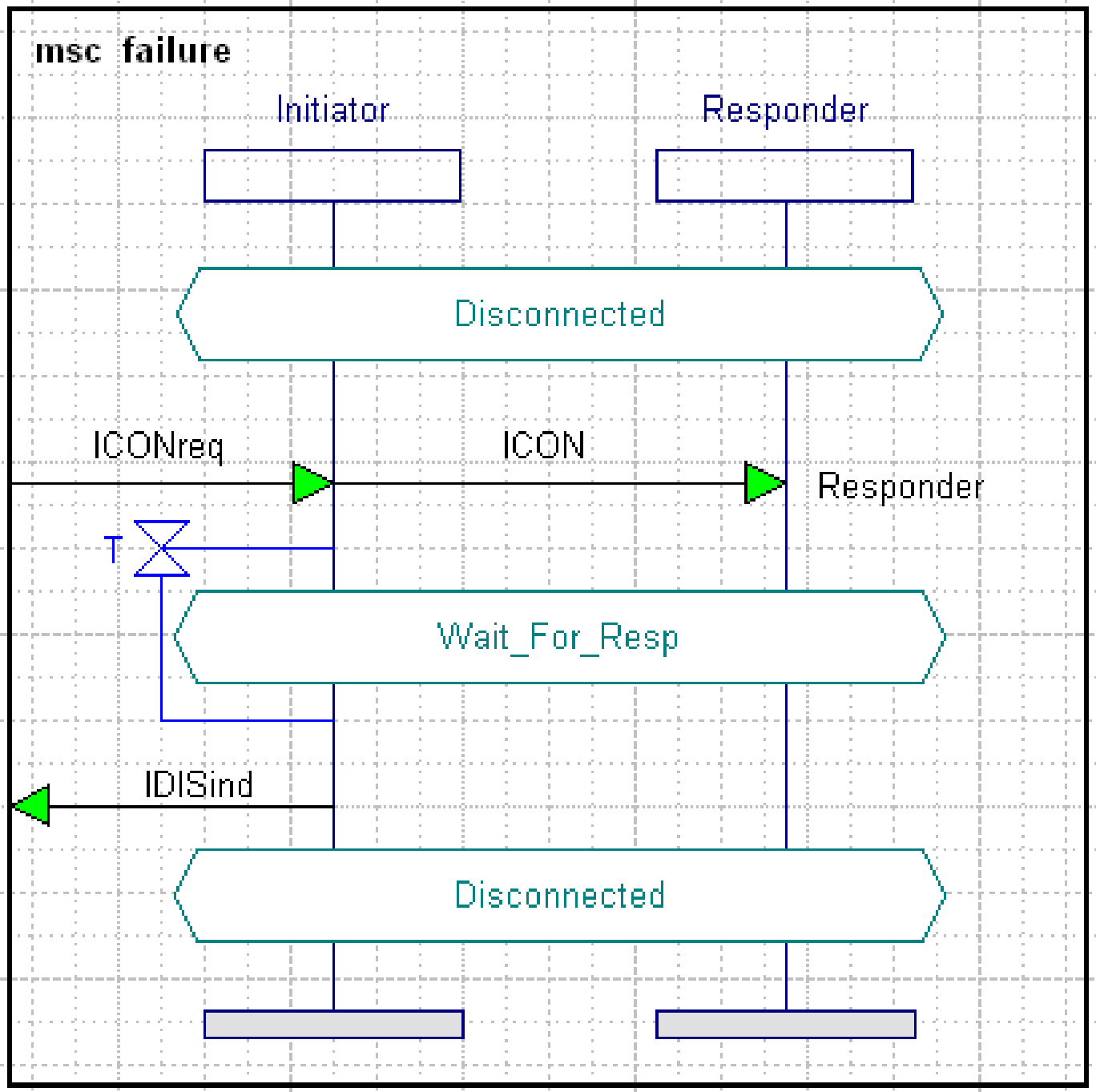
- Plain MSC = Sequence Diagram in UML
- Внутри диаграммы м.б. ссылка на другую диаграмму
- High-level MSC = простая блок схема, в которой каждый блок – это Plain MSC
  - Введены в UML 2.0 под именем Interaction Overview Diagram = обзорные диаграммы взаимодействия
- Явные манипуляции с таймерами
  - Таймер можно установить переустановить
  - От таймера можно получить сигнал (сообщение)
  - * Диаграммы сделаны в OSD Tool от LG Soft Lab


disconnected
failure
connection
connected
High-level MSC
setup_attach


msc data_transmission
Initiator
Medium
Responder
IDATreq(d)
MDATreq(DT,data,num)
MDATind(DT,data,num)
IDATind(d)
MDATind(AK,num)
MDATreq(AK,num)


msc connection
Initiator
Responder
Disconnected
ICONreq
ICON
ICONind
T
Wait_For_Resp
alt
T
IDISind
Disconnected
ICONF
ICONresp
T
ICONconf
Connected


msc ref
Initiator
Medium
Responder
data_transmission
data_transmission


msc failure
Initiator
Responder
Disconnected
ICONreq
ICON
Responder
T
Wait_For_Resp
IDISind
Disconnected

Выводы

- Все это не очень сложно – проще чем UML
- Графика проста и наглядна
- Эффективное практическое применение
  - Преобразование протокола выполнения SDL системы в Plain MSC
- В UML 2.0 сделан сильный поворот в эту сторону – практически всё заимствовано с точностью до обозначений
- ITU рассматривает вопрос о замене своих обозначений на обозначения UML!

7.2. Изменение контекста применения

- Расширение области применения языка
- Массовое языкотворчество
- Диалекты и говоры

Расширение области применения

- UML для моделирования бизнеса и «реинжиниринга»
- UML для моделирования приложений реального времени
- UML для моделирования приложений баз данных
- UML для моделирования мобильных устройств
- …
- Принимаются основные идеи, выкидываются все непонятные, добавляются свои значки и накладывается привычная семантика

Массовое языкотворчество

- Курс по UML во всех университетах на программистских специальностях
- Инициативные и отрытые проекты инструментов (UniMod, Argo UML, …)
- Десятки оригинальных книг по UML
- Tiny UML, Executable UML, …
- «Особое мнение» Microsoft:
  - UML является стратегическим направлением развития инструментов Microsoft
  - Если международные стандарты не соответствуют потребностям Microsoft, то тем хуже для стандартов

Диалекты и говоры

- Намеренные (профили) и случайные (ошибки) диалекты инструментов
  - Интероперабельность пока не полная
- Использование нотации UML для реализации ДРУГОЙ (хотя и похожей) семантики
  - UniMod Гурова (2005): Схема связей – нотация диаграмм классов, графы переходов – нотация диаграмм состояний
  - Предопределенная архитектура ≈ универсальный образец проектирования
  - Источник<u>и</u> событий → Управляющие автомат<u>ы</u> → Объект<u>ы</u> управления

7.3. Модельно центрированная разработка

- 2002 инициатива OMG – MDA (Model Driven Architecture)
  - MDD – Model Driven Development
- Модель (а не код программы) является главным и центральным артефактом процесса разработки
  - Визуальное конструирование программ
- Интеграция различных идей и стандартов
- Ясная идея («чертежи»), но очень много нерешенных и трудных вопросов

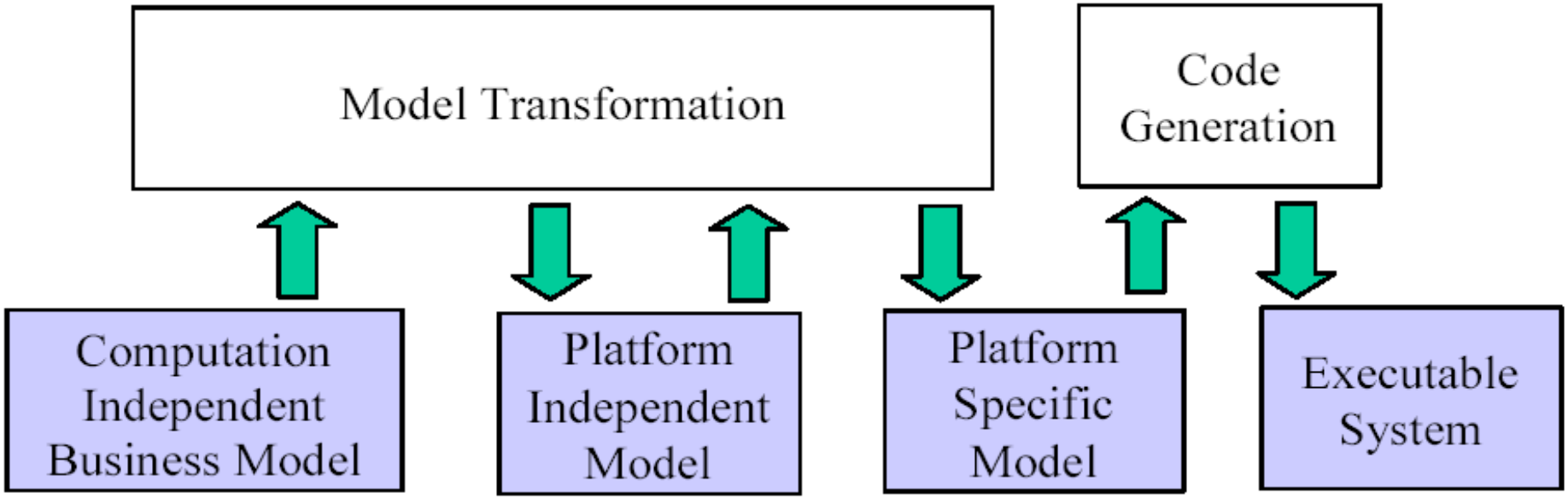
История и текущее состояние

- **Моделирование + эмуляция = программирование во встроенных системах ВСЕГДА, но без названия**
- **2002 год – инициатива OMG: MDA! MDA! MDA! MDA! MDA! …**
- **Независимые разработки: IBM, Oracle, Unisys, IONA, TeleLogic, …**
- **В 2004 году рынок инструментов MDA оценивался в $500 млн., пока ежегодное удвоение**

Основные идеи

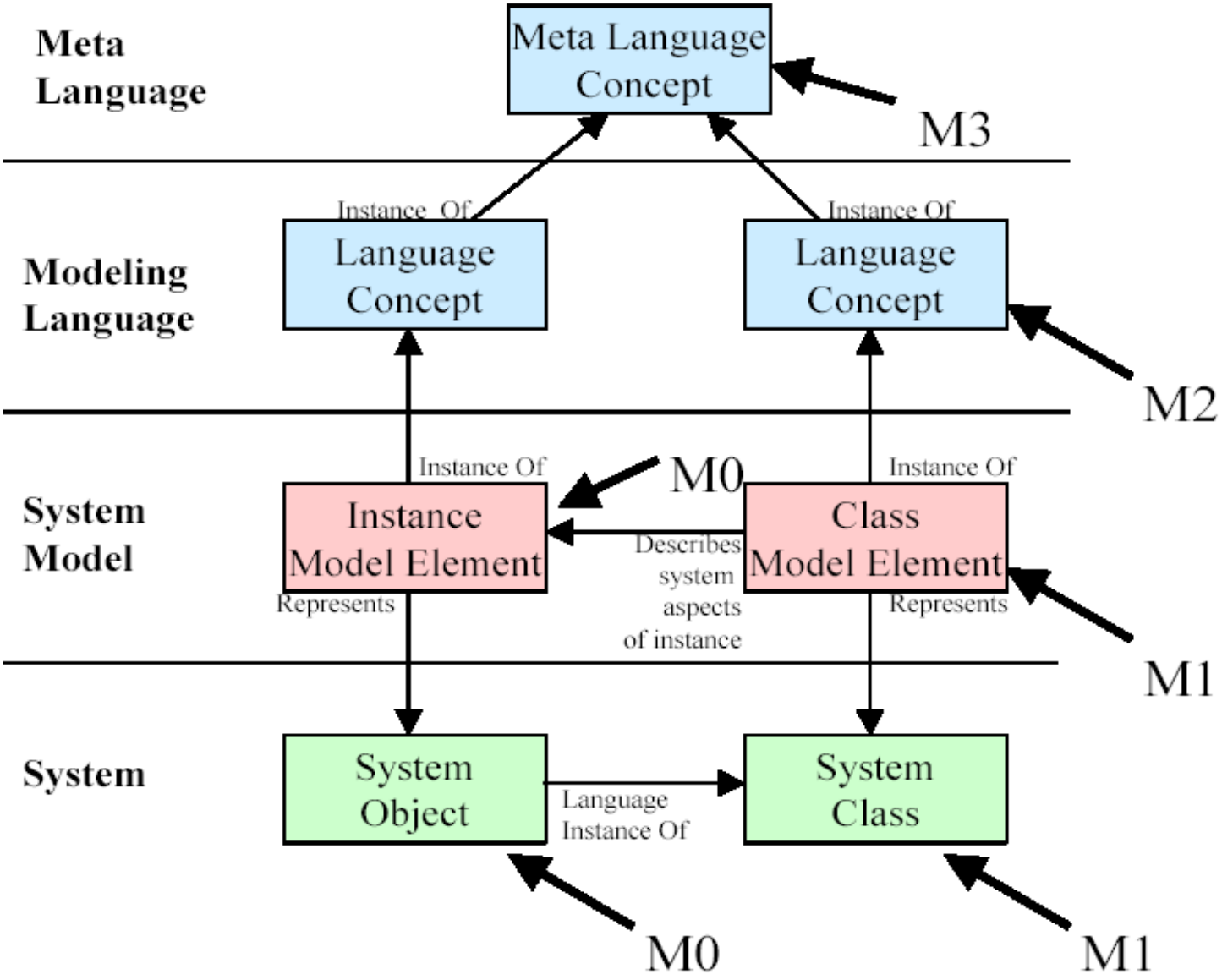
- Поэтапные трансформации моделей, ведущие к исполнимому приложению
- Согласованное определение всех моделей на единой основе – MOF
- Использование фундаментальных концепций теоретического программирования: повышение уровня абстракции, услуги по контракту, …
- Интеграция различных стандартов

Трансформации



- **Computation Independent Model (CIM);**
- **Platform-independent model (PIM);**
- **Platform-specific model (PSM).**

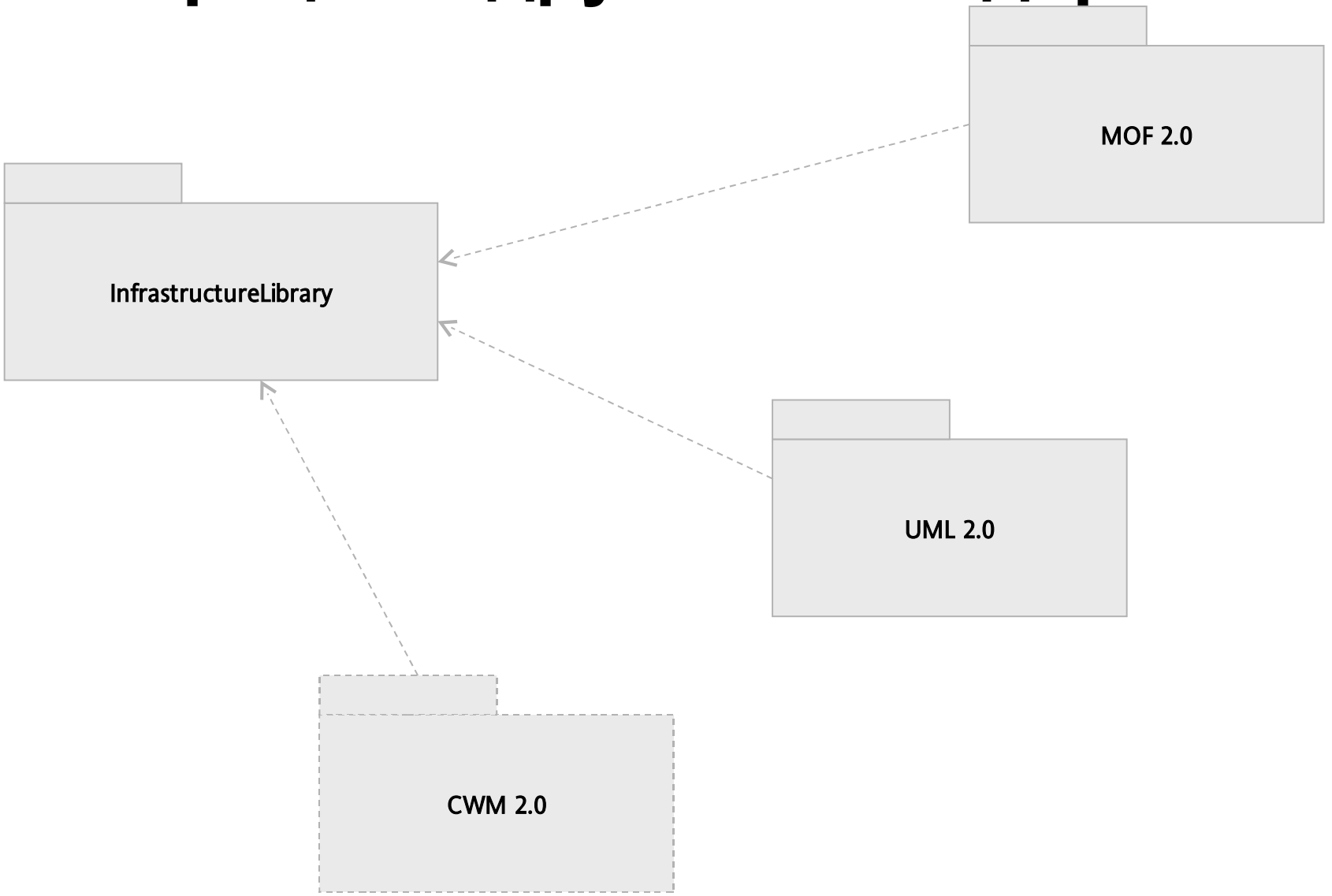
Уровни моделирования



Meta Object Facility – MOF

- UML стандартизирует представление результатов анализа и представления в форме моделей
- Meta-Object Facility (MOF) стандартизует представление моделей в форме объектов с унифицированными интерфейсами
- XML Metadata Interchange (XMI) стандартизируют обмен моделями между различными средствами их построения, а также средствами разработки приложений
- MOF = Classes UML + OCL operations – (qualifiers + aggregation + etc.)

Интеграция с другими стандартами



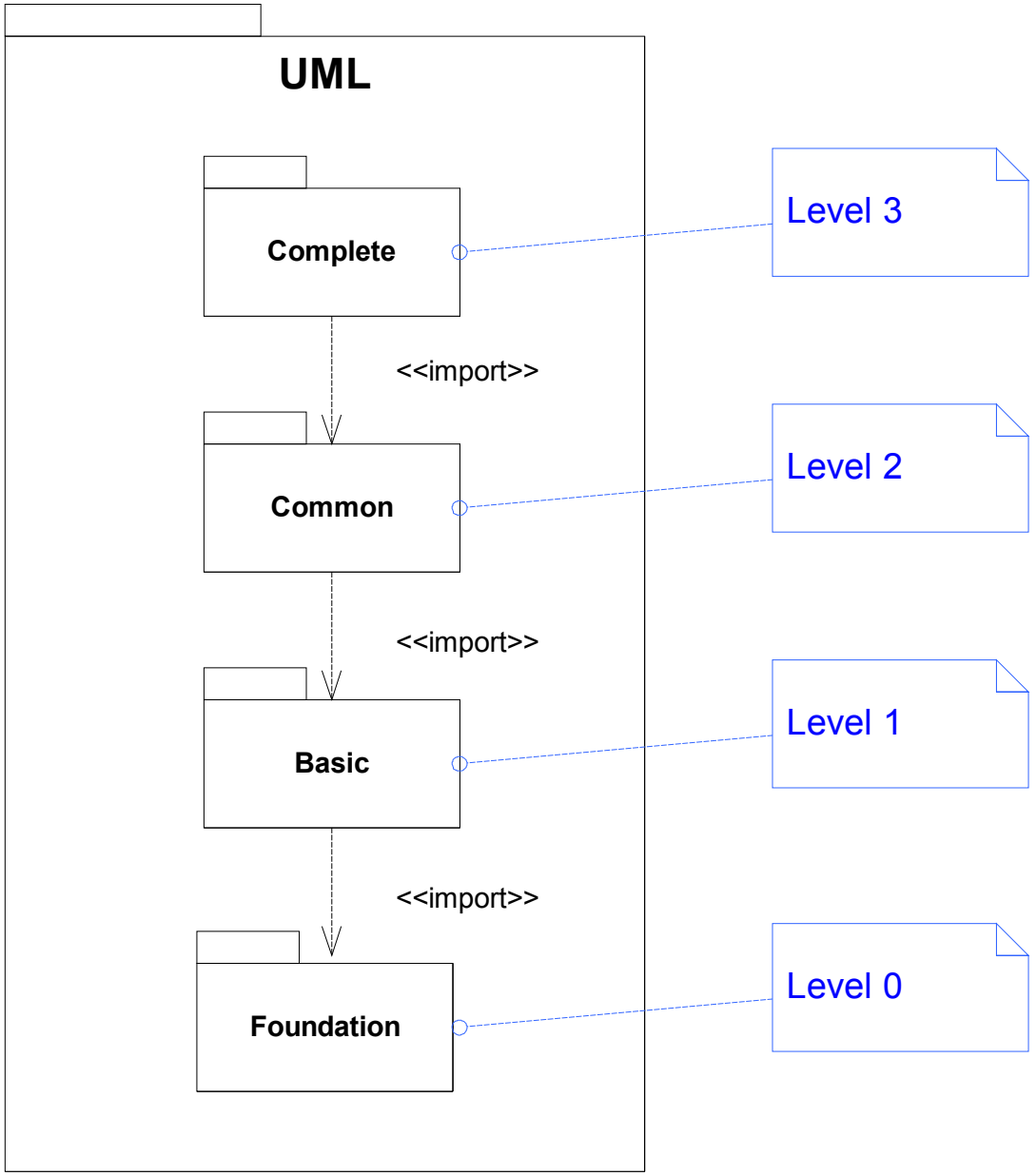
7.4. Новые средства в UML 2.0

- **Инициатива OMG**
  - **Требования к предложениям**
  - **Конкурсная основа**
- **7 альтернативных предложений**
  - **Некоторые – очень оригинальные**
  - **Победил консорциум U2P – наиболее мощная группа компаний**
  - **Alcatel, CA, ENEA Business Software, Ericsson, Fujitsu, HP, IBM, I-Logix, IONA, Kabira, Motorola, Oracle, Rational, SOFTEAM, Telelogic, Unisys, WebGain**
  - **Консервативное расширение 1.x с адаптацией идей конкурентов**

Основные цели

- Реструктурировать и уточнить язык, чтобы его было легче реализовывать, настраивать, применять (infrastructure)
- Улучшить поддержку компонентного программирования (EJB, COM+, …)
- Улучшить масштабируемость и структурированность (SDL blocks, etc., …)
- Уточнить и усовершенствовать описание поведения (MSC, сети Петри, …)

Уровни совместимости



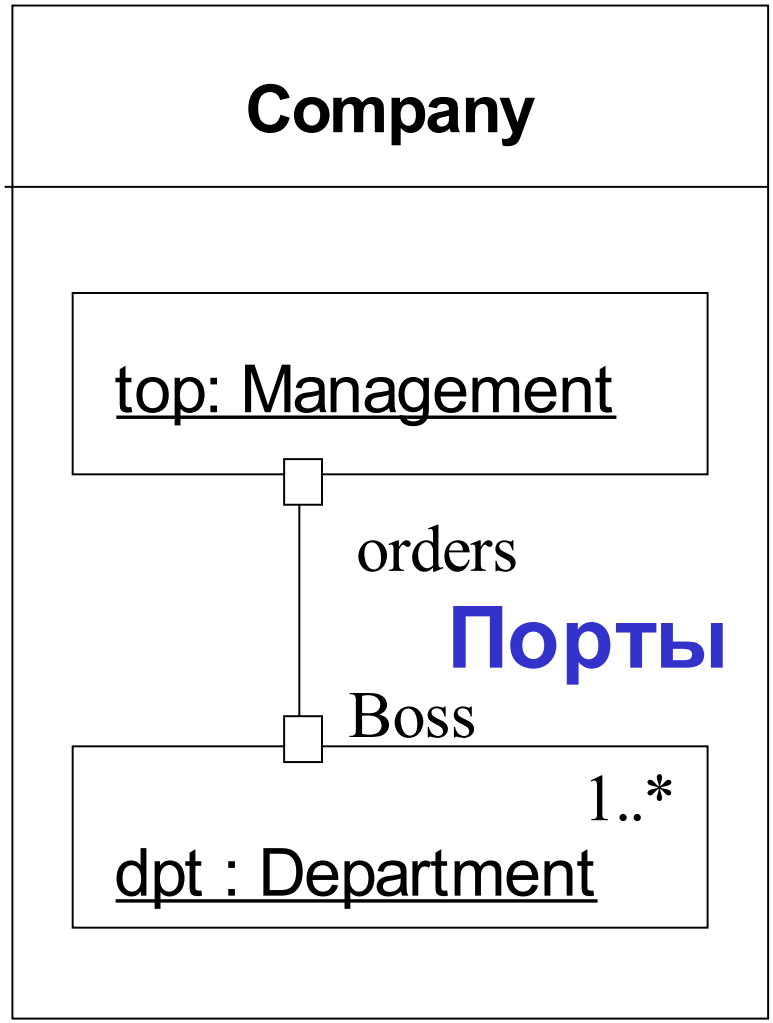
Почва для совместимых диалектов и специальных расширений

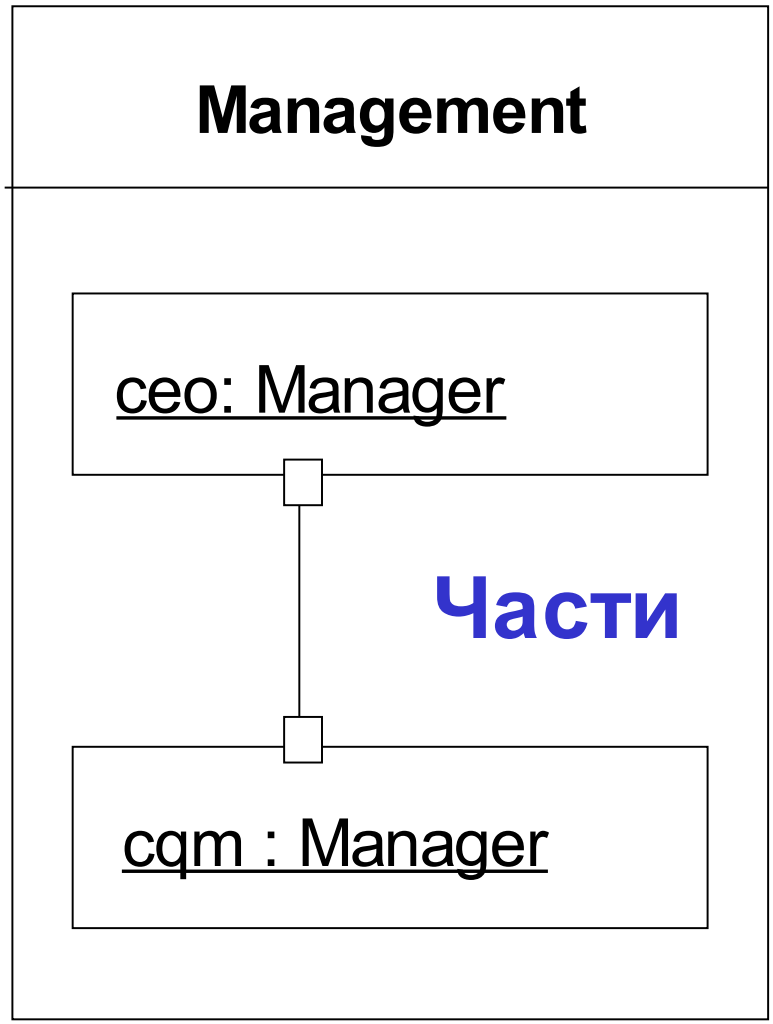
Основные нововведения

- Структурированные классификаторы (Structured Classifiers)
- Компоненты (Components)
- Взаимодействия (Interactions)
- Диаграммы деятельности (Activities)
- Машины состояний (State Machines)

Структурированные классификаторы

- Объект (экземпляр классификатора) может состоять из частей (parts)
  - на любую глубину вложенности
- Части имеют тип (класс) и могут взаимодействовать
- Взаимодействие осуществляется через соединители (connectors)
- Соединители присоединяются к объектам и частям через порты (ports)





Вложенные классы

Компоненты

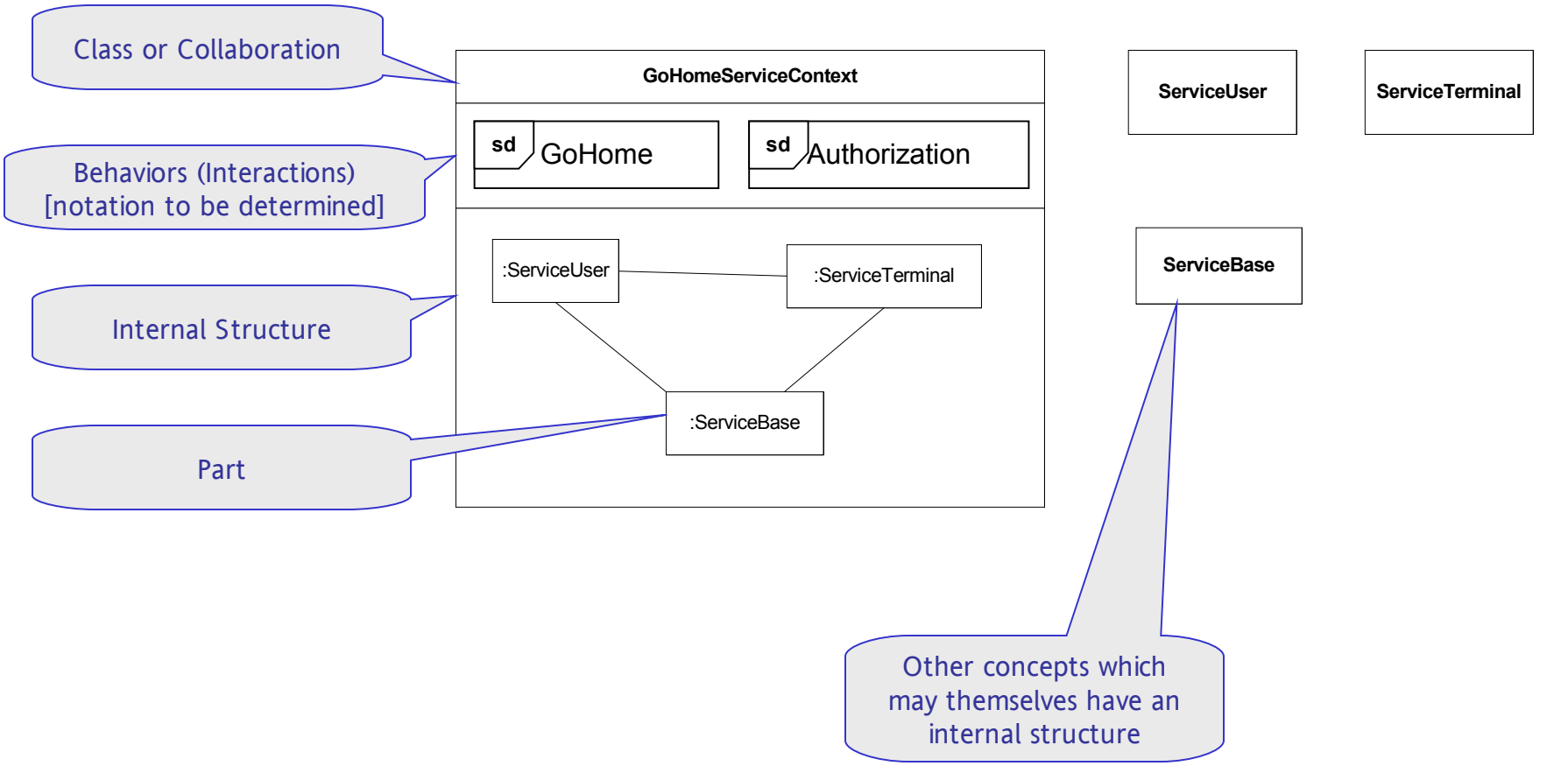
- Компонент имеет предоставляемые (provided) и запрашиваемые (required) интерфейсы
- Взаимодействие осуществляется через порты
- Протокол взаимодействия (множество допустимых последовательностей вызовов) описывается специальной машиной состояний


Staff Management
Staff Forms
Компонент (значок)
Предоставляемый интерфейс
Positions
Запрашиваемый интерфейс
Personnel Forms
Personnel Management

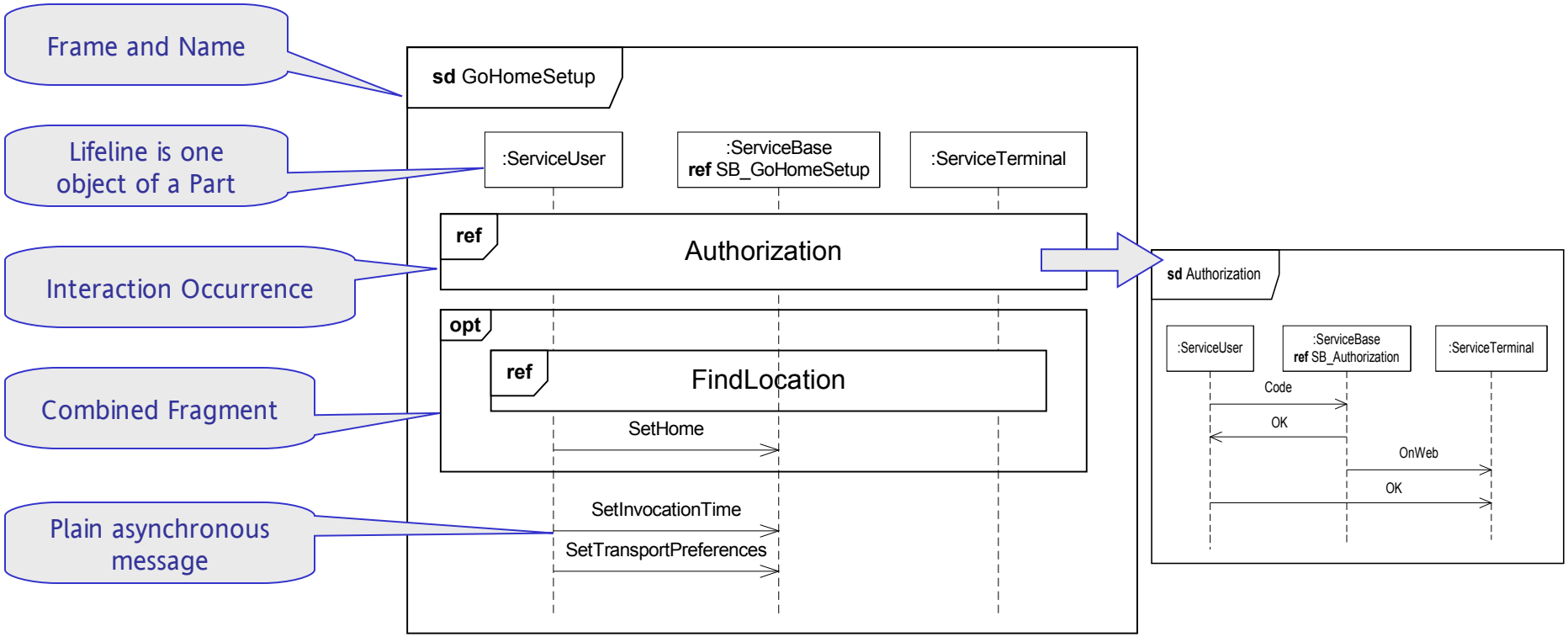
Описание взаимодействия

- Линия жизни = экземпляр объекта во взаимодействии
- Структурная декомпозиция (ref)
- Составные шаги (combined fragments) = структуры управления
- Обзорные диаграммы (interaction overview) = high-level in MSC
- Диаграммы синхронизации (Timing diagrams)

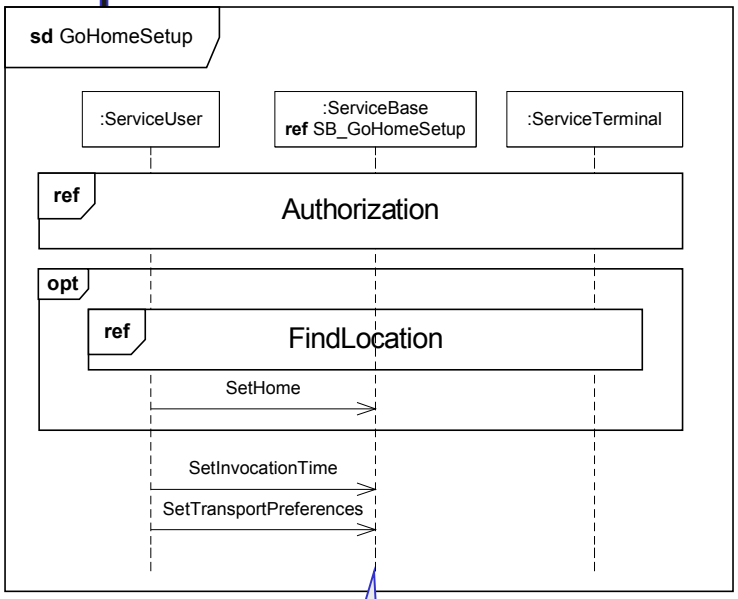
Interaction Context



Interaction Occurrences

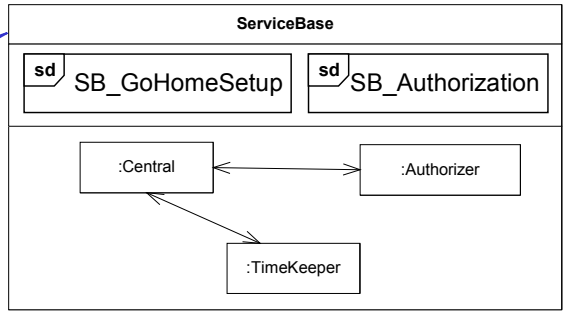


Decomposition

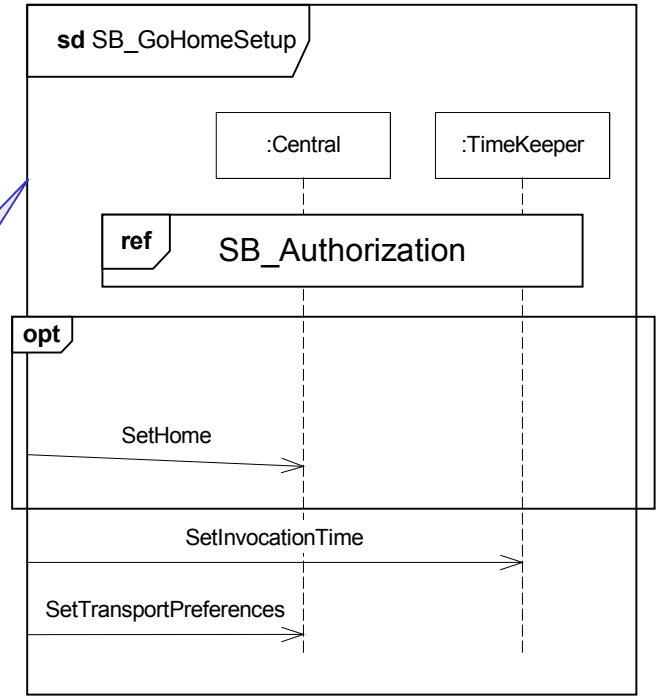


Decomposed lifeline

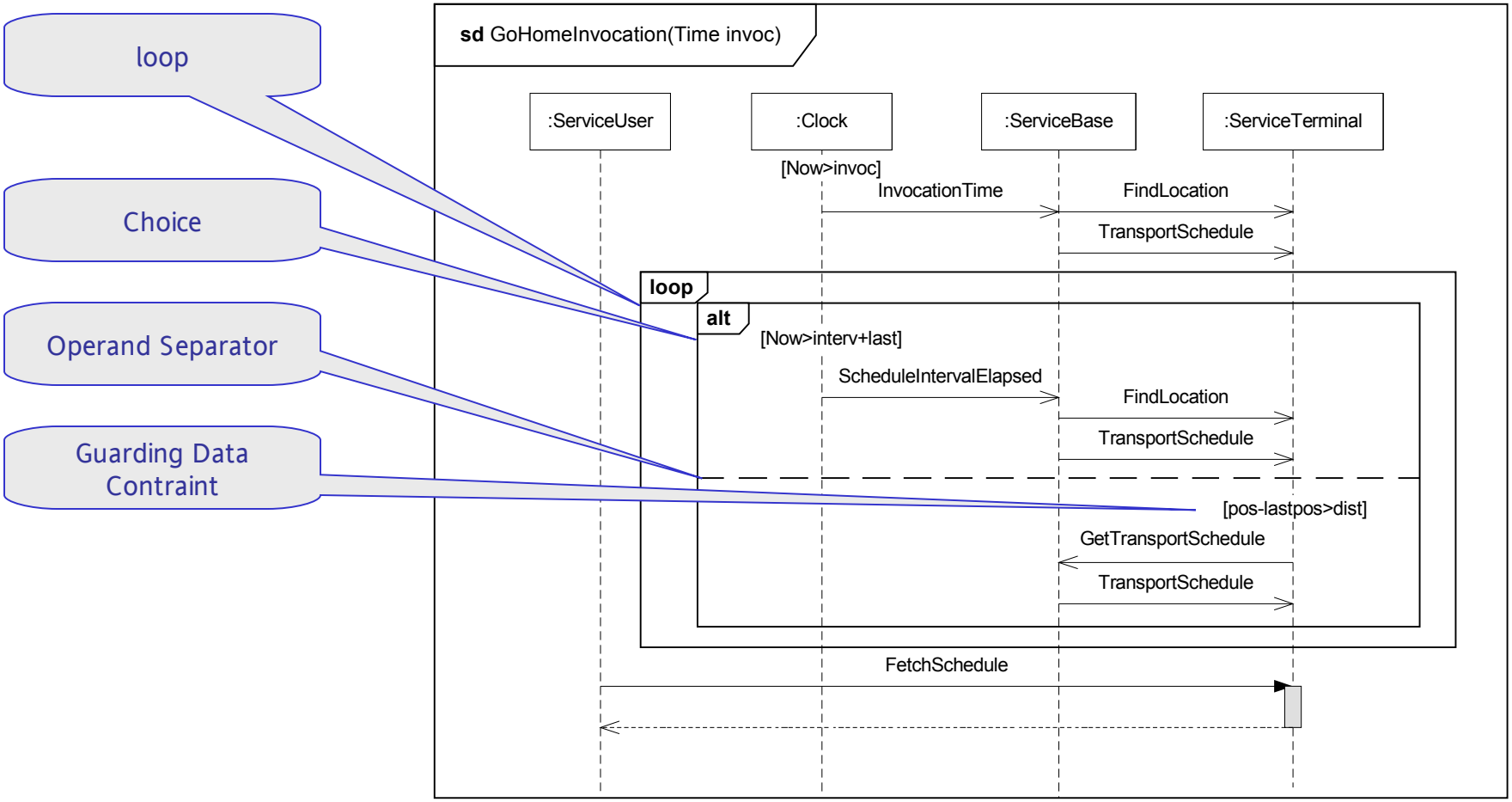
Detailed context



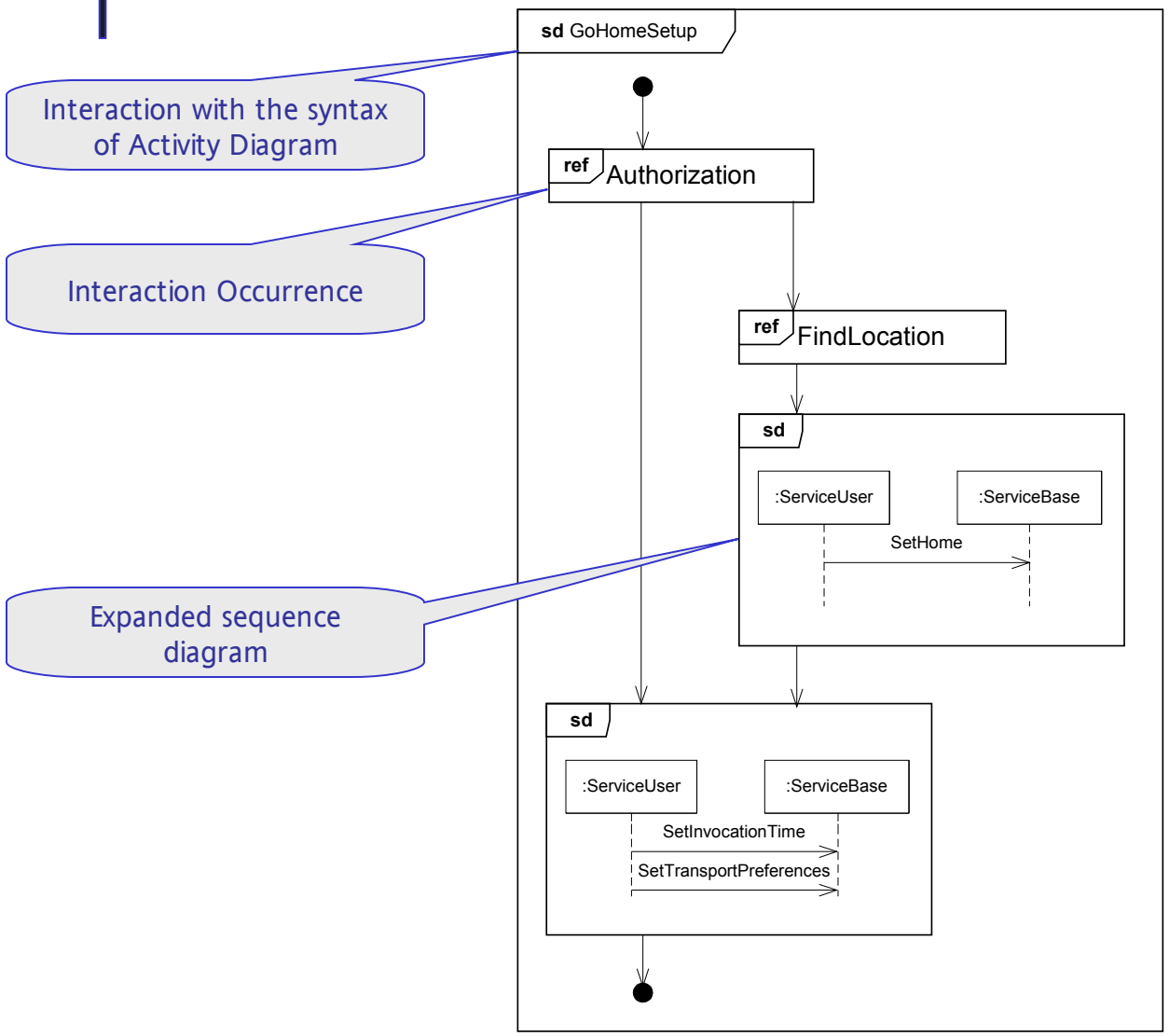
Decomposition with global constructs corresponding to those on decomposed lifeline



Combined Fragments and Data



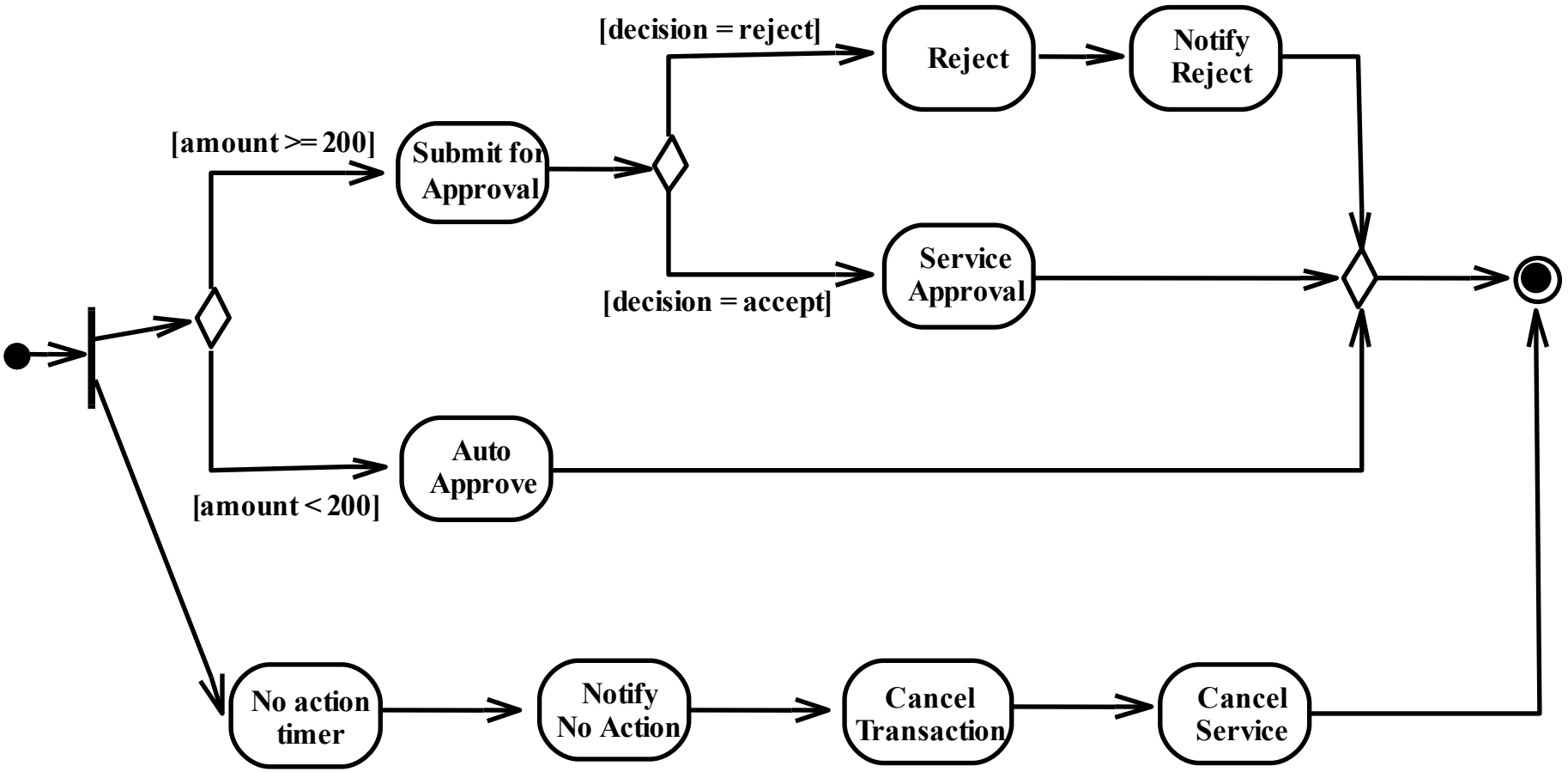
Interaction Overview Diagram



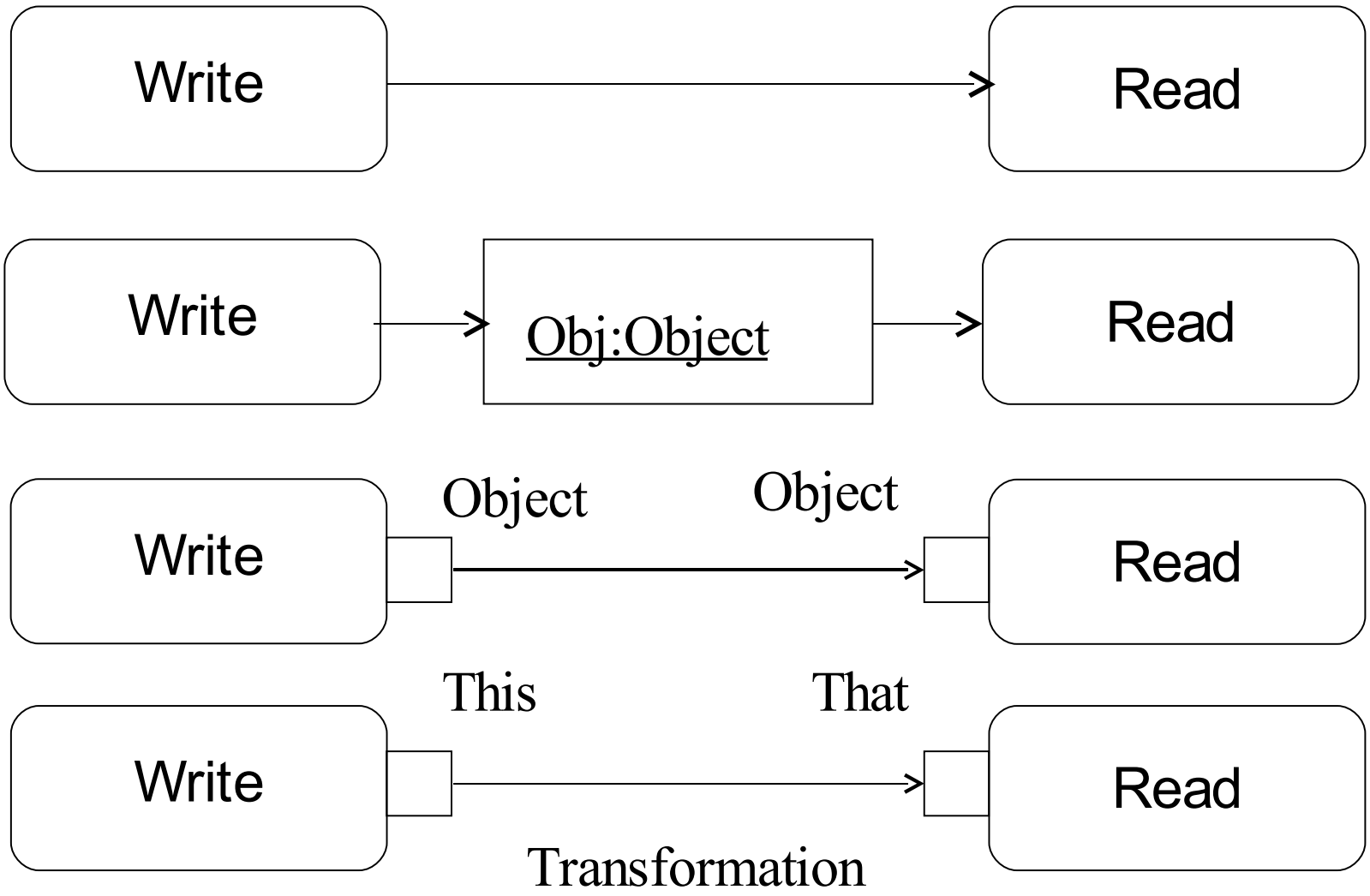
Диаграммы деятельности

- **Новая семантика (сети Петри):**
  - **Диаграмма состоит из узлов и дуг. По дугам движутся маркеры (tokens), в т.ч. управление**
  - **Узлы действий имеют входные и выходные контакты (pins)**
  - **Действие выполняется, если ВСЕ входные контакты могут получить маркеры по дугам**
  - **Когда действие выполняется, его выходные контакты получают маркеры, и эти маркеры могут быть переданы по дугам**
  - **Узлы управления могут направлять, удалять и копировать маркеры**

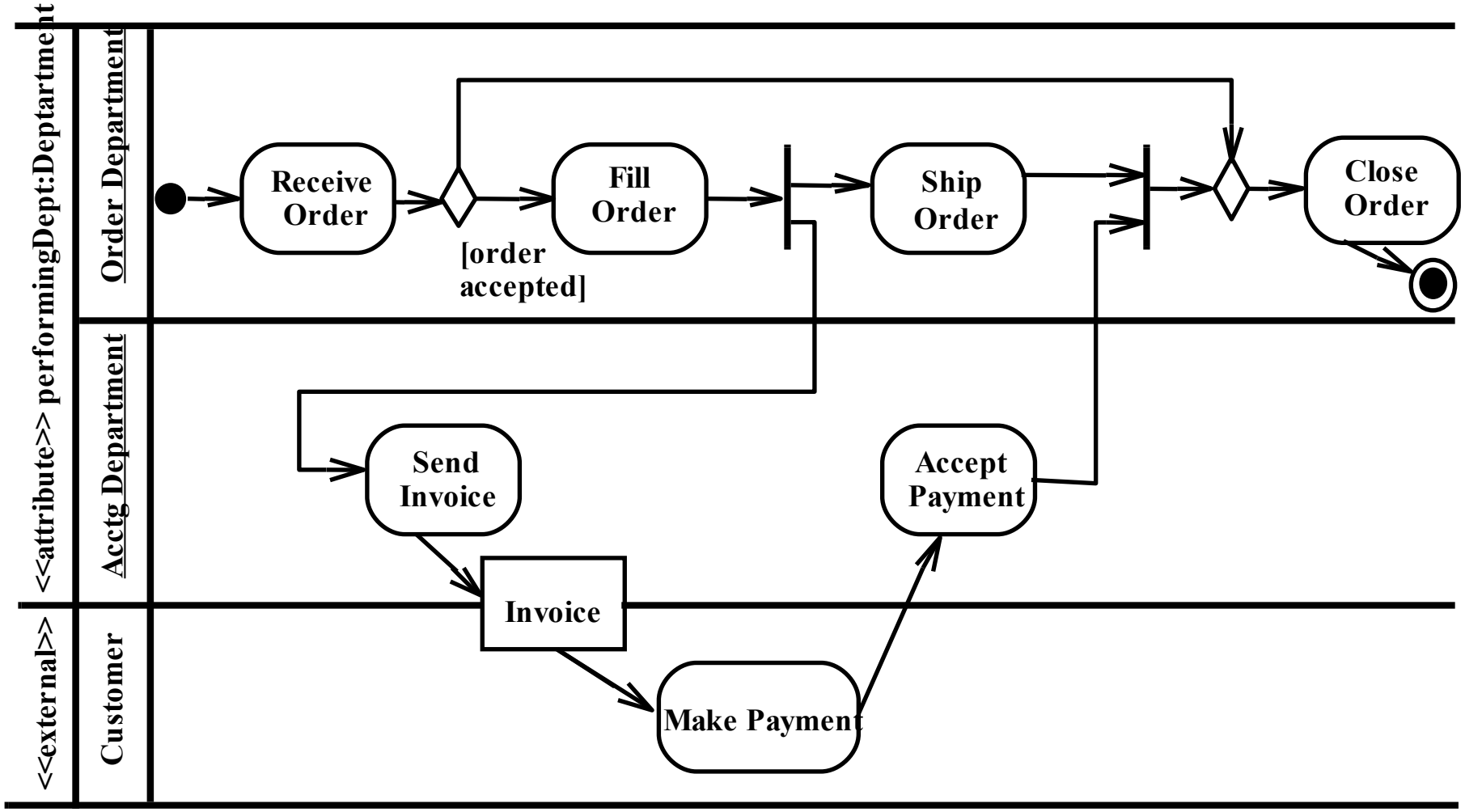
Пример (control token)



Пример (data token)



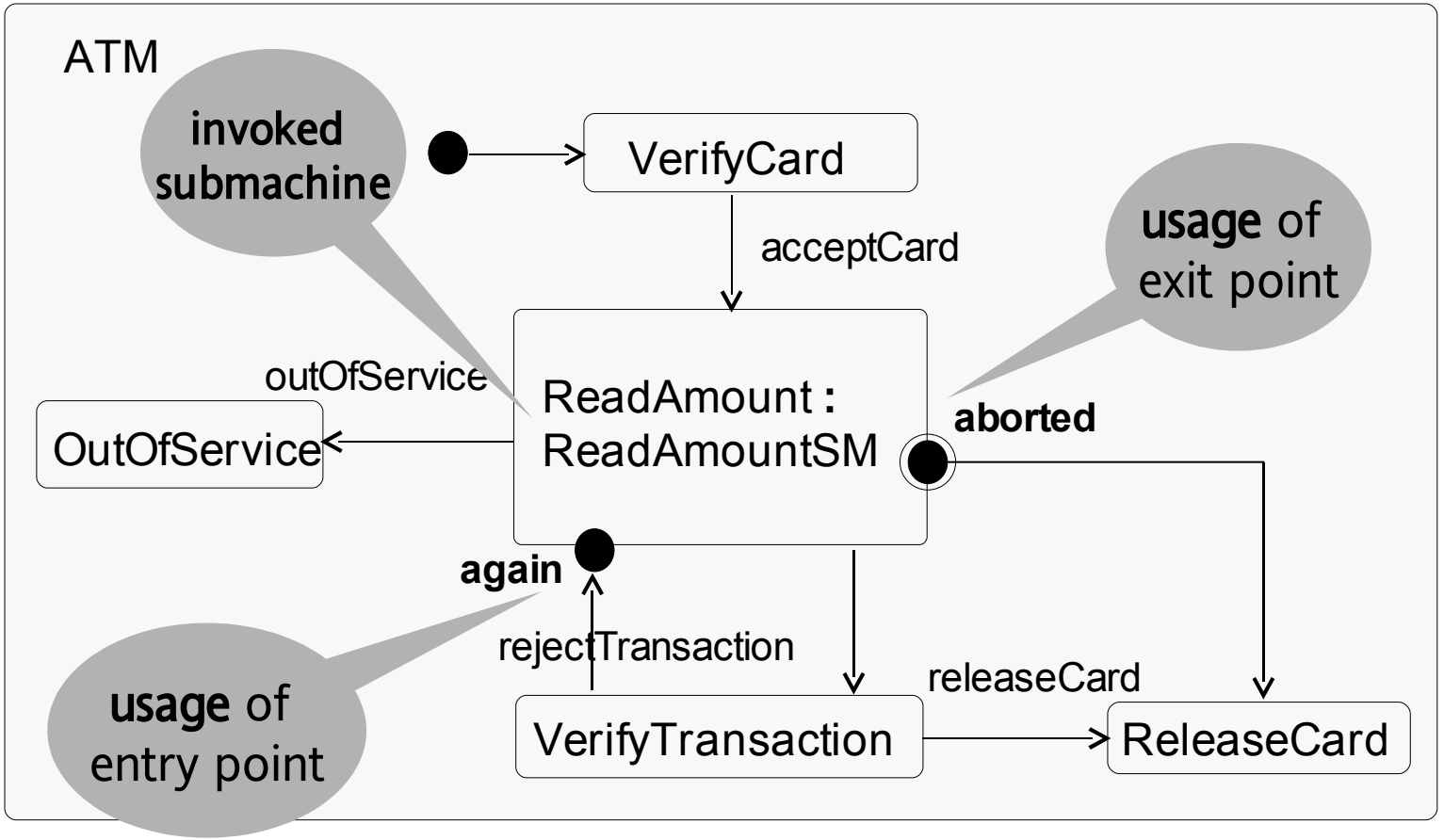
Подразделения (дорожки)



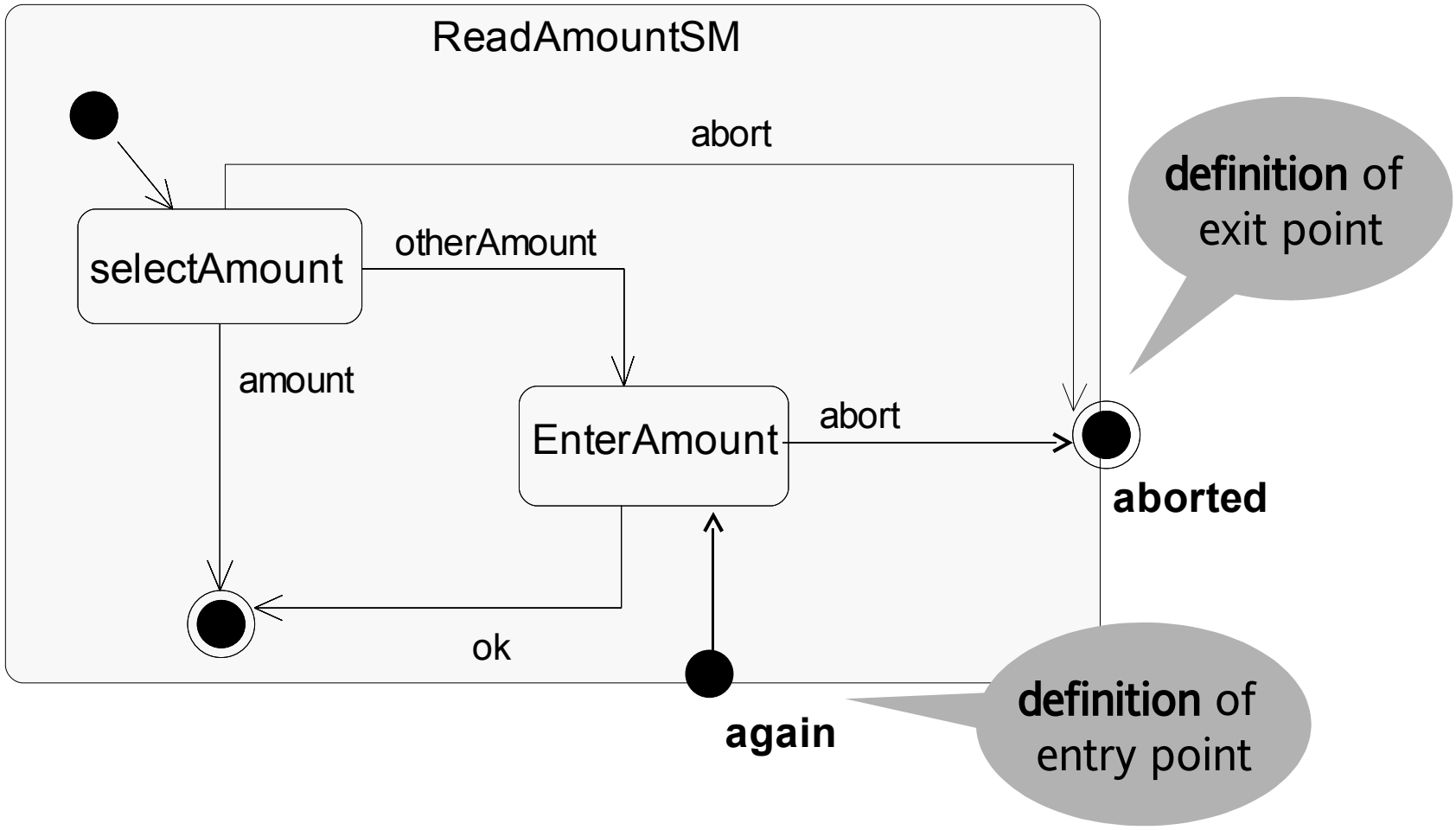
Машины состояний

- Два типа автоматов для описания:
  - Поведения объекта класса (behavior state machine)
  - Допустимой последовательности событий (операций) порта или интерфейса (protocol state machine)
- Точки входа выхода вложенной машины (составного состояния)

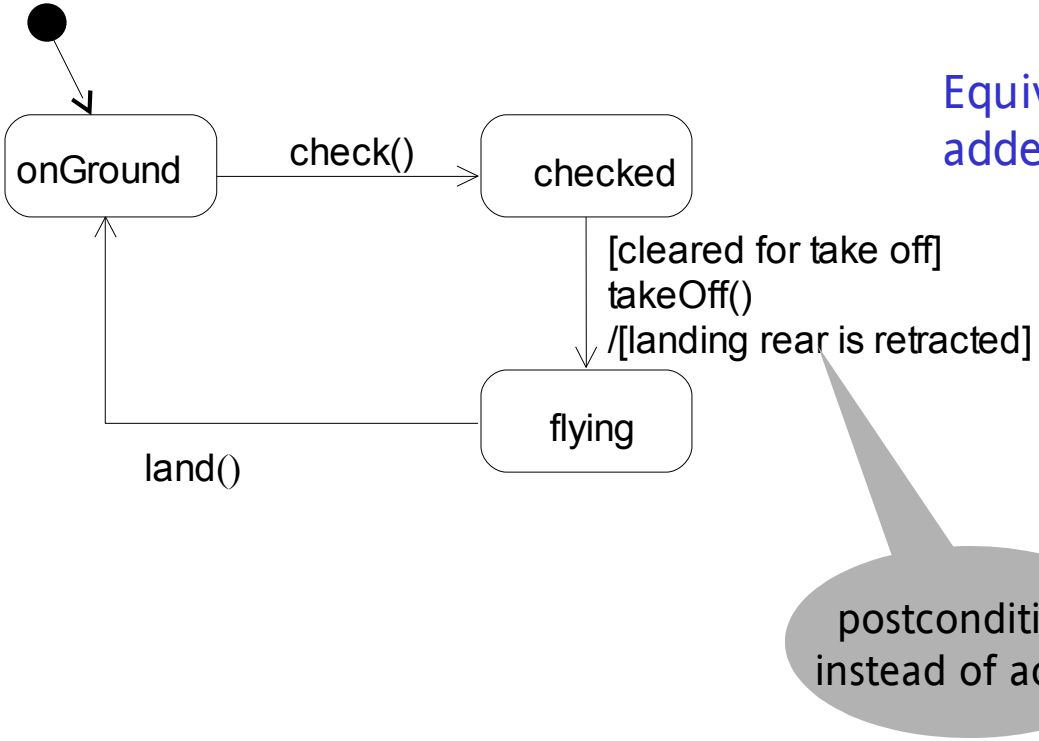
Entry/Exit Points: Usage



Entry/Exit Points: Definition



Protocol State Machines



Equivalent to pre and post conditions added to the related operations:

takeOff()

Pre

-in state "checked"
-cleared for take off

Post

-landing rear is retracted
-in state "flying"

Статус UML 2.0

- Принятый стандарт
  - Final adopted в 2005
- Отзывы общественности
  - UML = Unwanted Modeling Language
  - Already done (к сожалению, неправда)
- «Большие» поставщики готовят продукты
- Небольшое увеличение количества
- Значительное улучшение качества

Выводы

- UML является самой передовой технологией разработки
- Область применения UML быстро расширяется
- Стандарты UML отстают от практики
- Инструменты отстают от стандартов
- Детские болезни роста быстро проходят