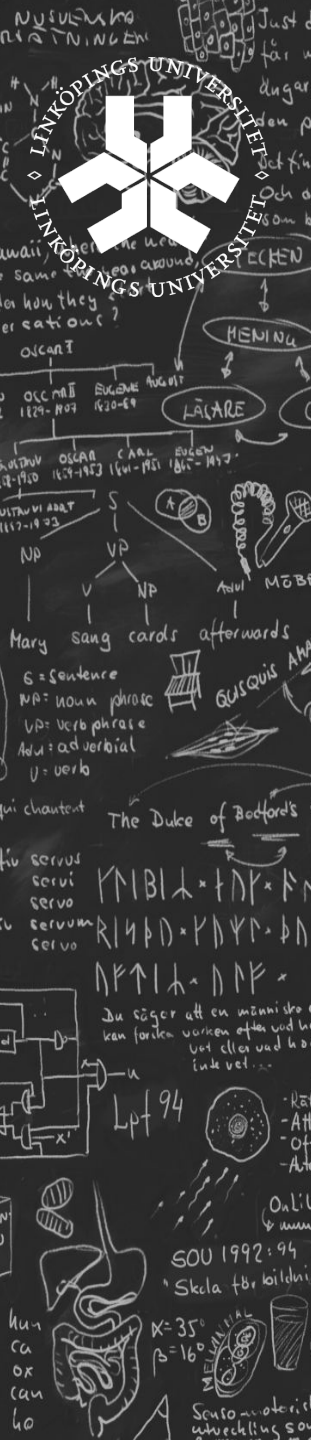


# Model-Based Requirements Engineering

Tutorial 2012-04-12

by

Kristian Sandahl

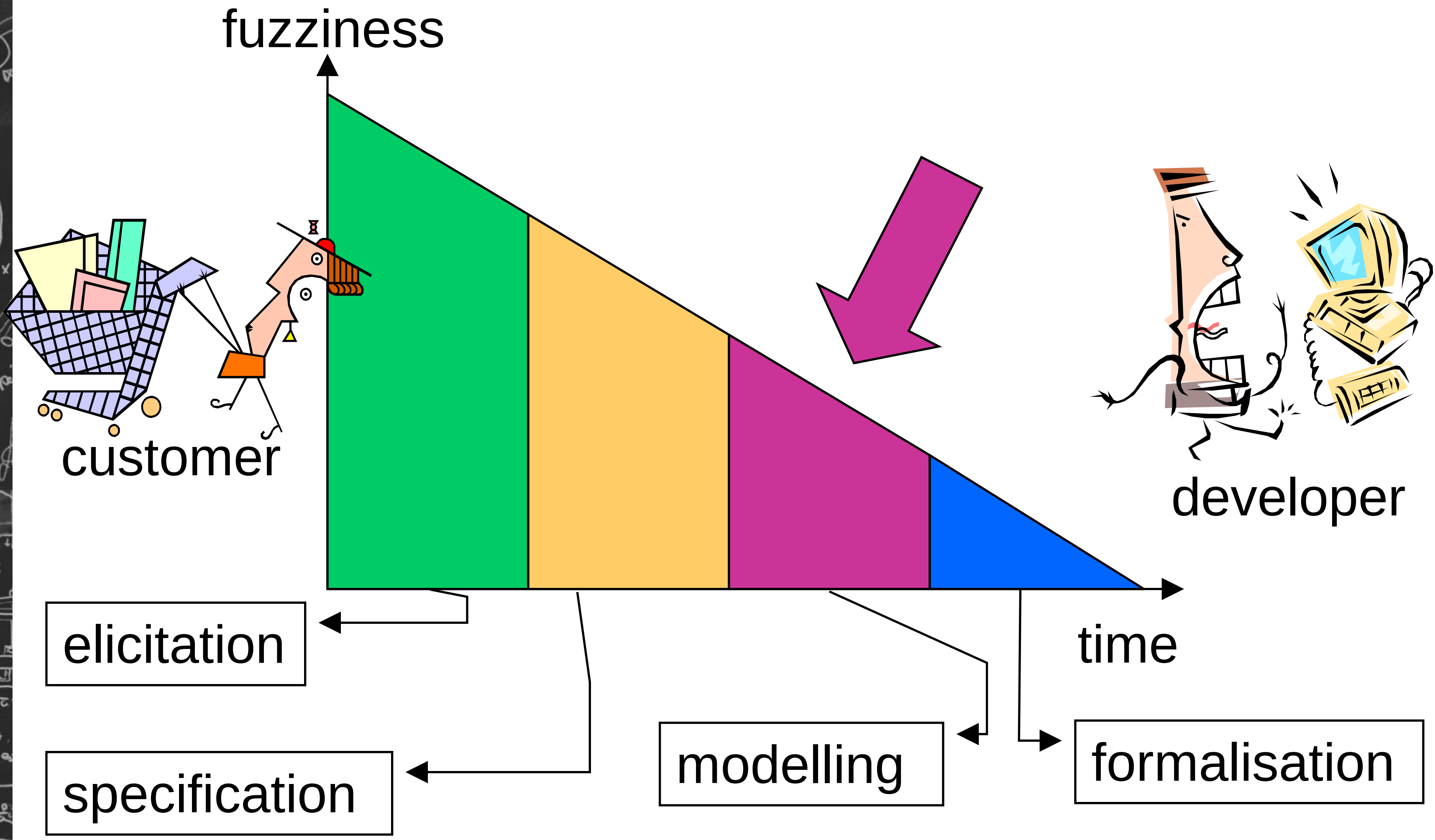
# Planned topics

- Modelling requirements in UML
- Requirement model traceability
- Non-functional software requirements
- Short introduction to requirements in SysML

# Analysis: Conceptual Modelling

- Representation in semi-formal notation
- Often diagrammatic representation
- Examples:
  - Object-orientation, use-cases, state-machines
  - Activity diagrams
  - Data flow diagrams
  - Entity-relationship models

# Requirement representation process



fuzziness

customer

developer

time

elicitation

specification

modelling

formalisation

# Introduction

- Models **supplement** natural language
- Models support both elicitation and design
- The boundaries between specification and design have to be decided
- There are high transition costs from functional to object-oriented models
- **UML** is becoming the standard notation

# Develop complementary system models

Benefits:

- Forces analysis from different views
- Different readers take different views

Implementation:

- The UML 4+1 model
- Combination of other diagrams

Drawbacks:

- Different readers make different interpretation
- Normally weak exception handling
- Hard to model non-functional requirements

# UML 4+1 Model

Views:

- Logical view: which parts belong together?
- Process view: what threads of control are there?
- Development view: what is developed by whom? reuse issues
- Physical view: which part will execute where?

+

- Use-case model: required system from the user’s point of view. static and dynamic

# Use-case modelling

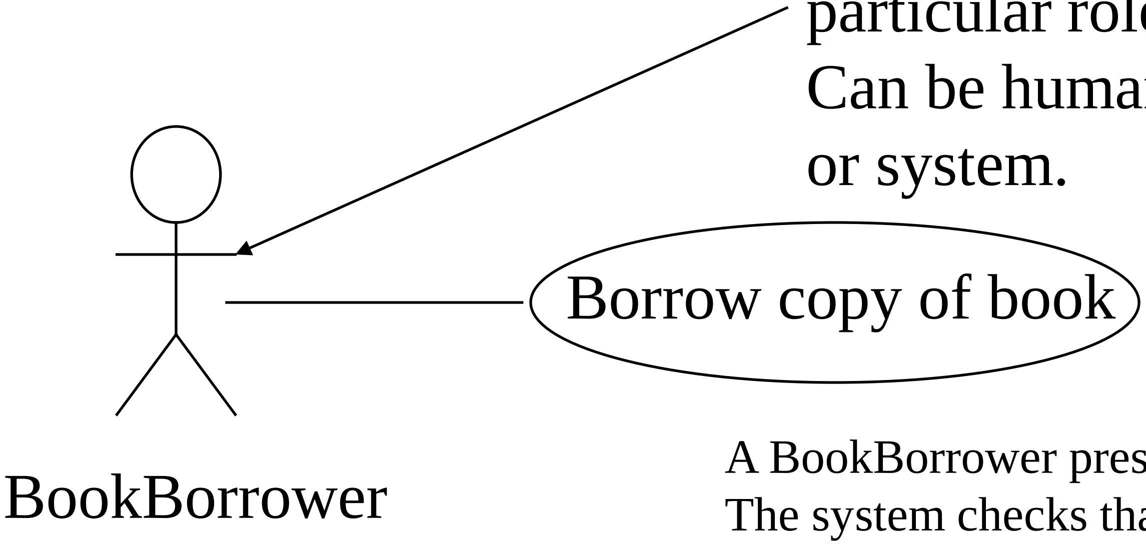
A use-case is:

"… a particular form or pattern or exemplar of usage, a scenario that begins with some user of the system initiating some transaction of sequence of interrelated events."

(*Jacobson, m fl 1992*)

# Use-case diagram

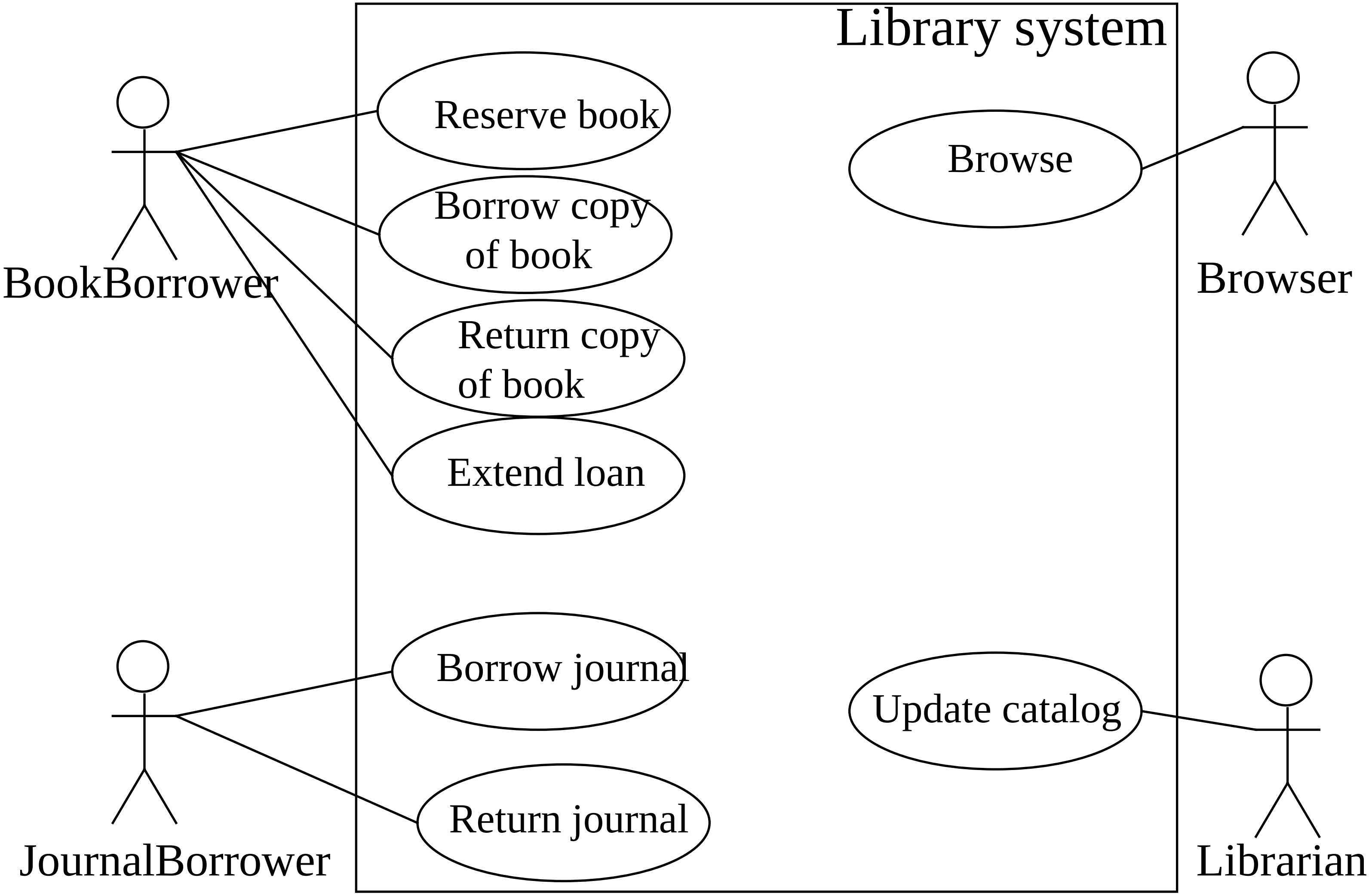
Actor: a user of the system in a particular role. Can be human or system.



Detail of use-case →

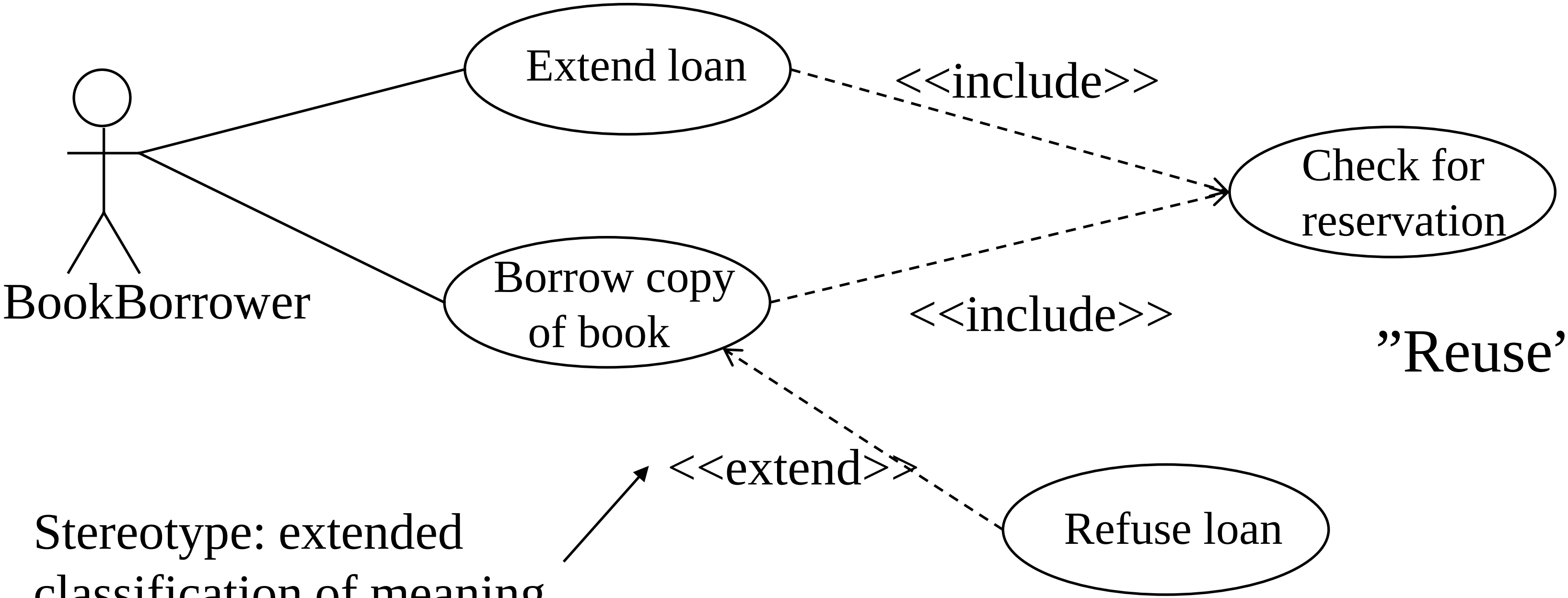
A BookBorrower presents a book. The system checks that the potential borrower is a member of the library, and that he/she doesn’t already have the maximum permitted book on loan. This maximum is 6 unless the member is a staff member, in which case it is 12. If both checks succeed, the system records that this library member has this copy of the book on loan. Otherwise it refuses the loan.

# Use-case diagram for the library



# Relations between use-cases

Please, keep as simple as possible.



BookBorrower

Extend loan

Borrow copy of book

<<include>>

Check for reservation

<<include>>

<<extend>>

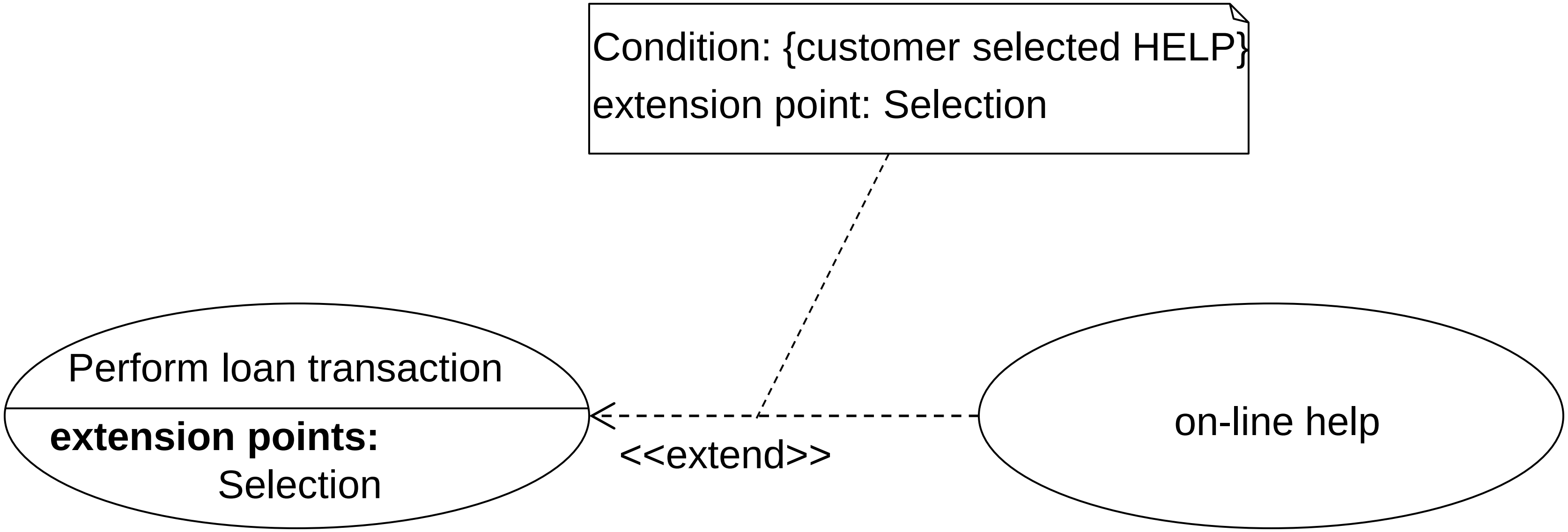
Refuse loan

”Reuse”

Stereotype: extended classification of meaning

”Separating scenarious”

# Extension points



Condition: {customer selected HELP}
extension point: Selection

Perform loan transaction

**extension points:**
Selection

<<extend>>

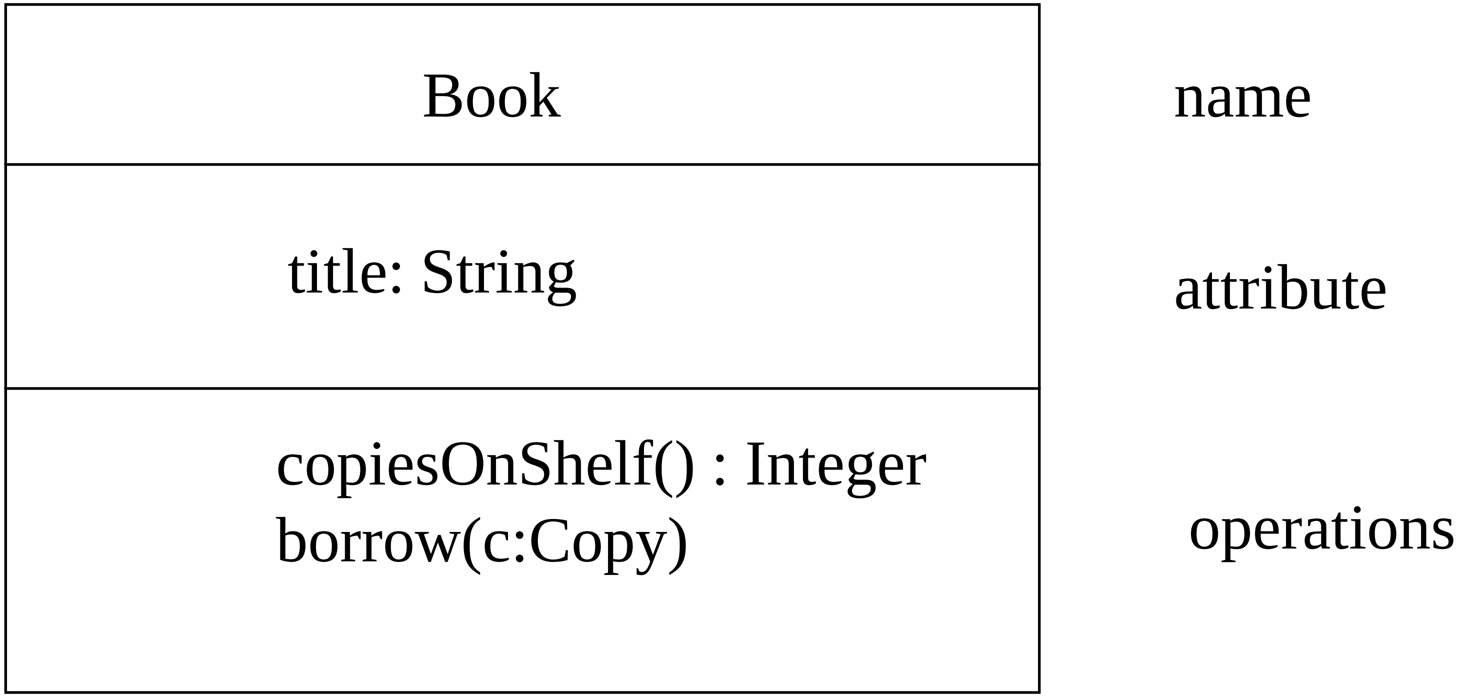
on-line help

# Identifying classes: noun analysis

A BookBorrower presents a book. The system checks that the potential borrower is a member of the library, and that he/she doesn’t already have the maximum permitted book on loan. This maximum is six unless the member is a staff member, in which case it is 12. If both checks succeed, the system records that this library member has this copy of the book on loan. Otherwise it refuses the loan.

- **book – real noun handled by the system**
- system – meta-language
- borrower – already actor
- **library member – handled by the system**
- **staff member – handled by the system**
- checks – event
- **copy of book – handled by the system**

# The single class model



Book

title: String

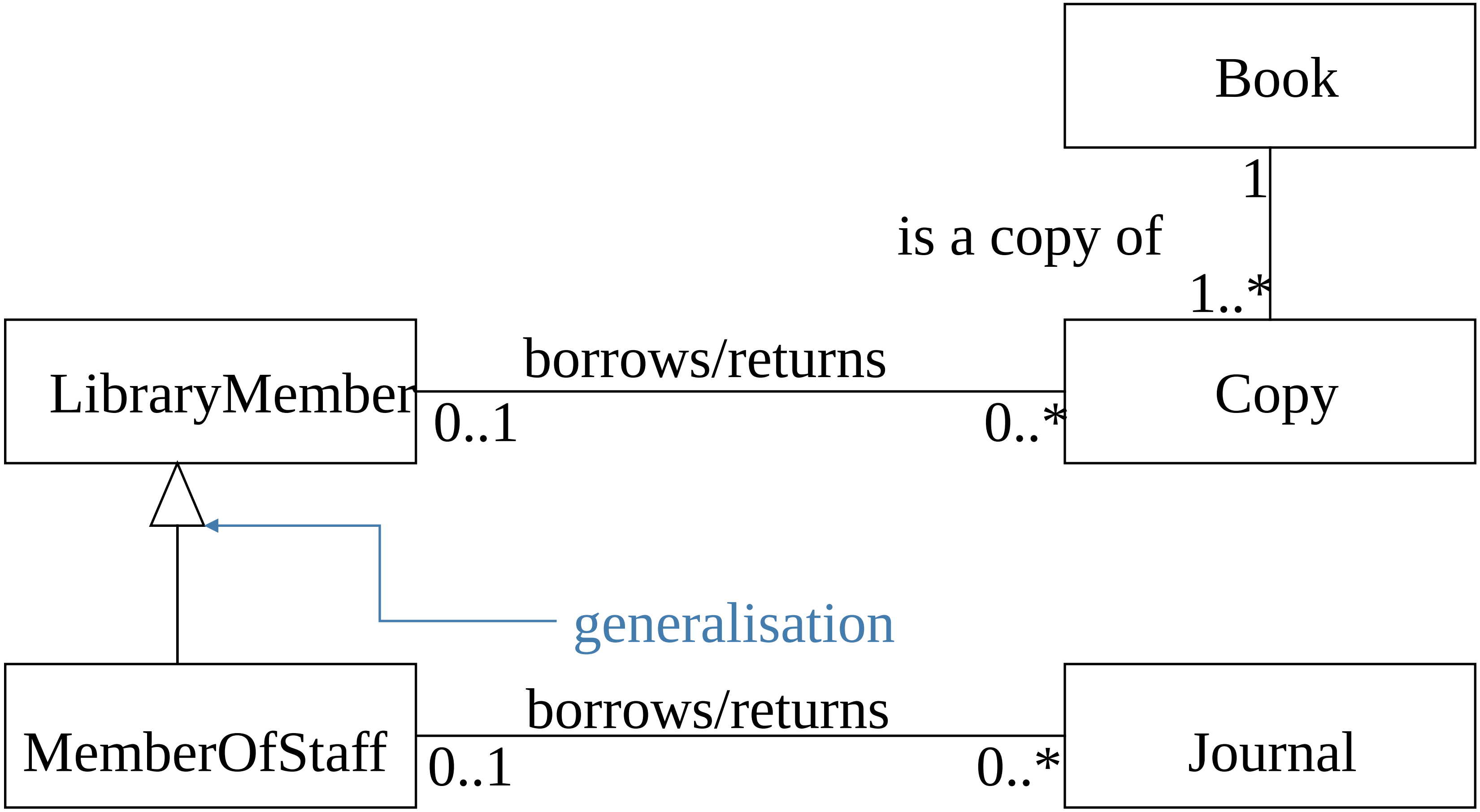
copiesOnShelf() : Integer
borrow(c:Copy)

name

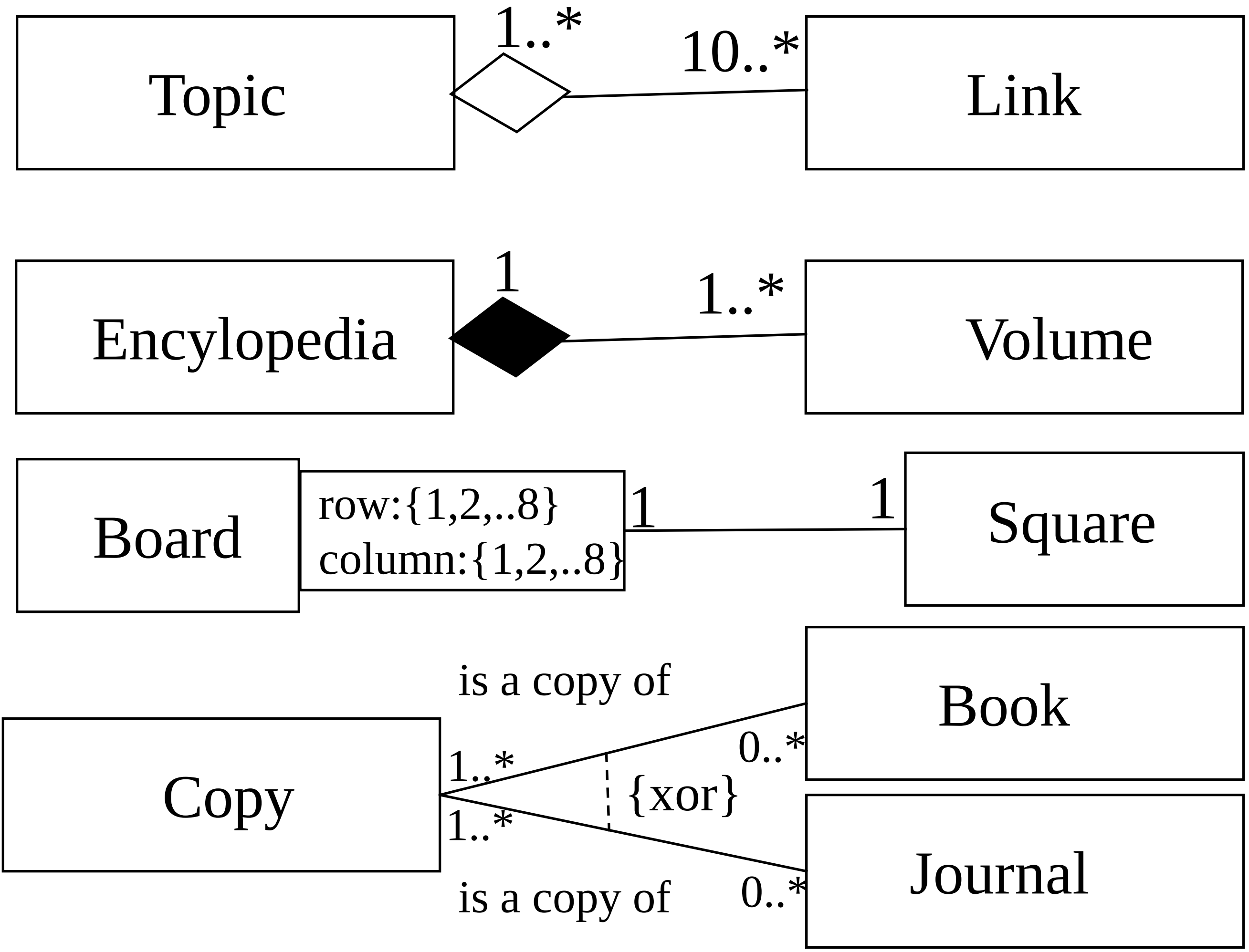
attribute

operations

# The library class model



# More relations between classes



aggregation

composition

qualified association

constraint

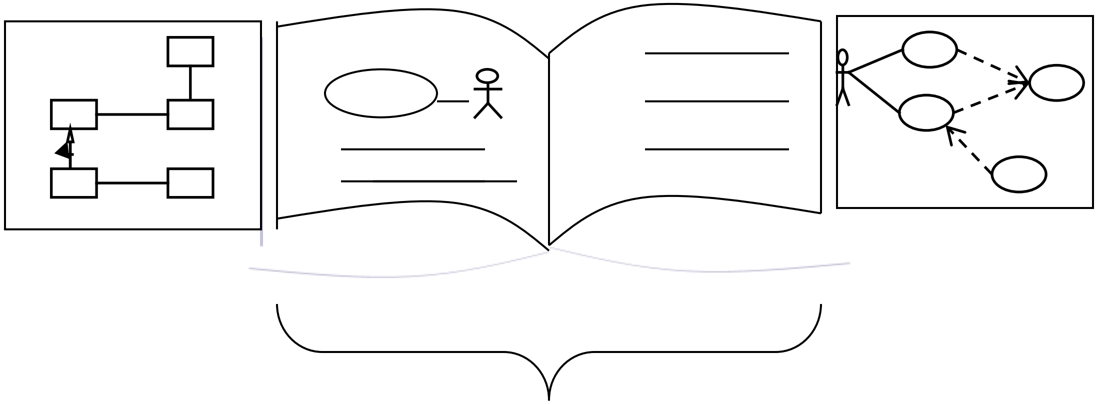
# Where to go now?

1. Continue with a traditional specification
2. Writing a detailed use-case specification
3. Continue modelling

# Writing a detailed use-case specification

- Name
- Brief Description
- Flow of Events: Write the description so that the customer can understand it. The flows can include a basic flow, alternative flows, and sub flows.
- (Key scenarios)
- Special Requirements
- Preconditions
- Post-conditions
- Extension points

# “Classical” use-case specification



max 40 pages

# Use-cases need System-wide requirements

1. Introduction
2. System-Wide Functional Requirements
3. System Qualities
   - 3.1 Usability
   - 3.2 Reliability
   - 3.3 Performance
   - 3.4 Supportability
4. System Interfaces

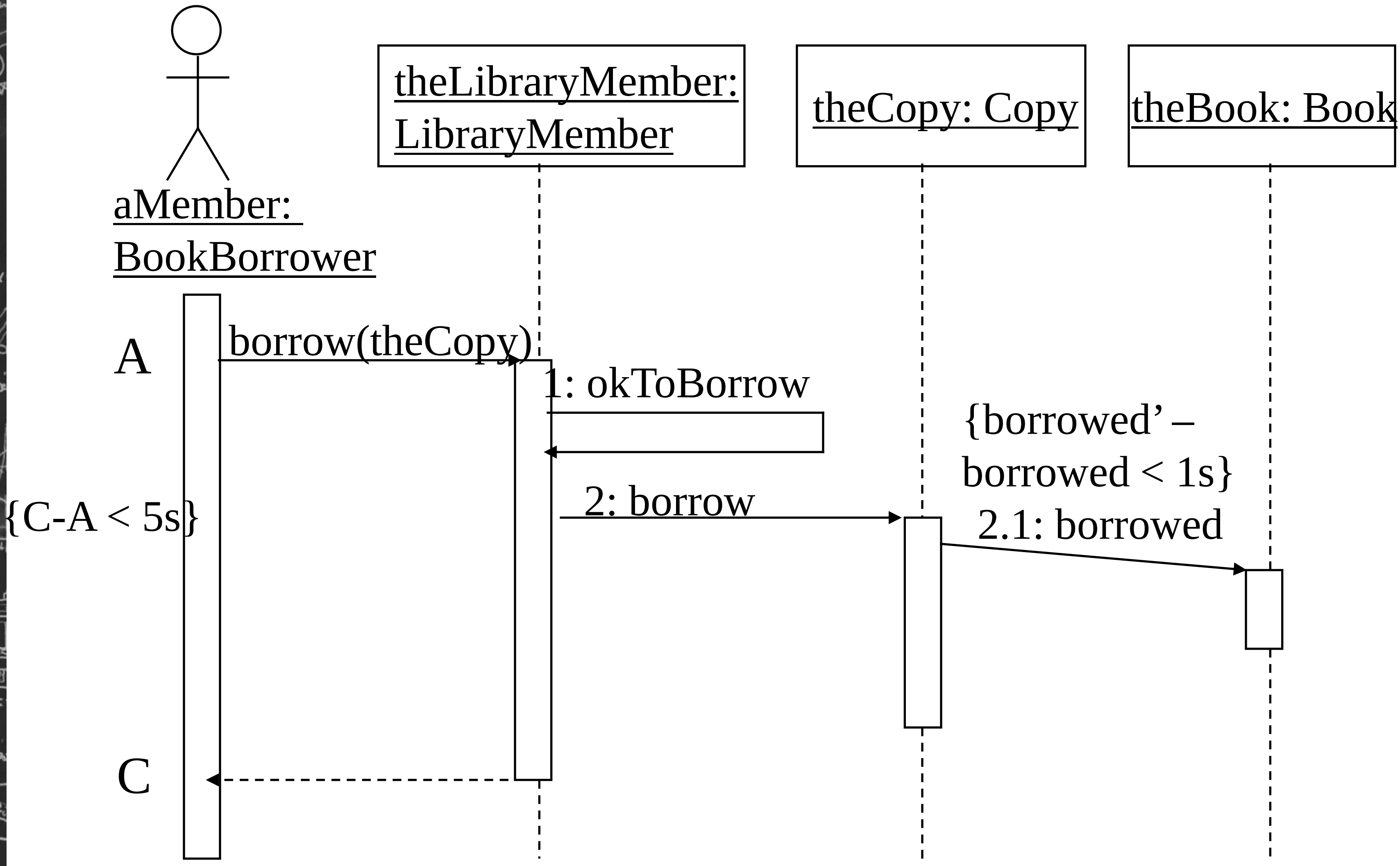
4.1 User Interfaces
   - 4.1.1 Look & Feel
   - 4.1.2 Layout and Navigation Requirements
   - 4.1.3 Consistency
   - 4.1.4 User Personalization & Customization Requirements

4.2 Interfaces to External Systems or Devices

- 4.2.1 Software Interfaces
- 4.2.2 Hardware Interfaces
- 4.2.3 Communications Interfaces

5. Business Rules
6. System Constraints
7. System Compliance
   - 7.1 Licensing Requirements
   - 7.2 Legal, Copyright, and Other Notices
   - 7.3 Applicable Standards
8. System Documentation

# Continue modelling :Sequence diagram



aMember: BookBorrower

theLibraryMember: LibraryMember

theCopy: Copy

theBook: Book

A

borrow(theCopy)

1: okToBorrow

{borrowed’ – borrowed < 1s}

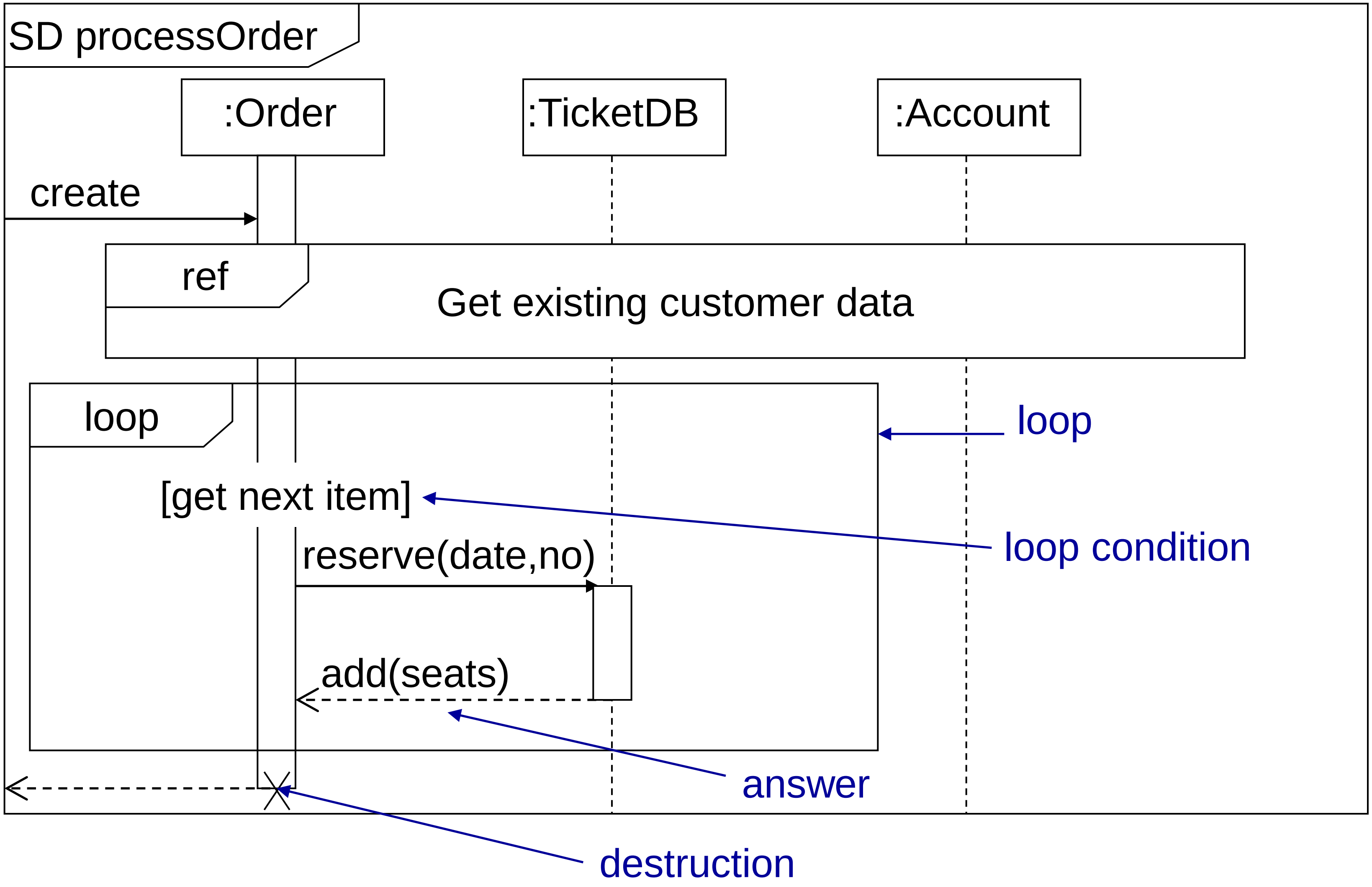
2: borrow

2.1: borrowed

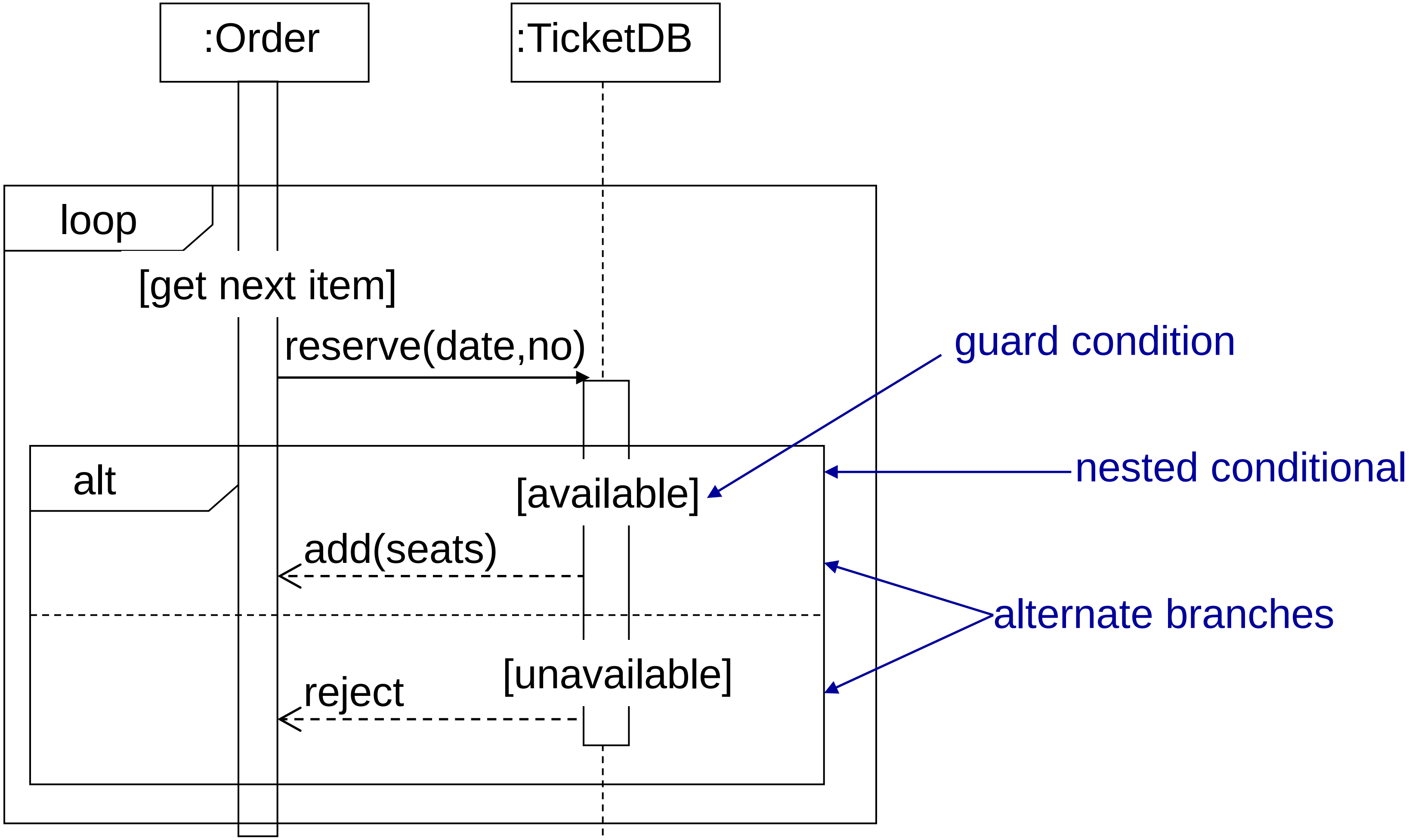
{C-A < 5s}

C

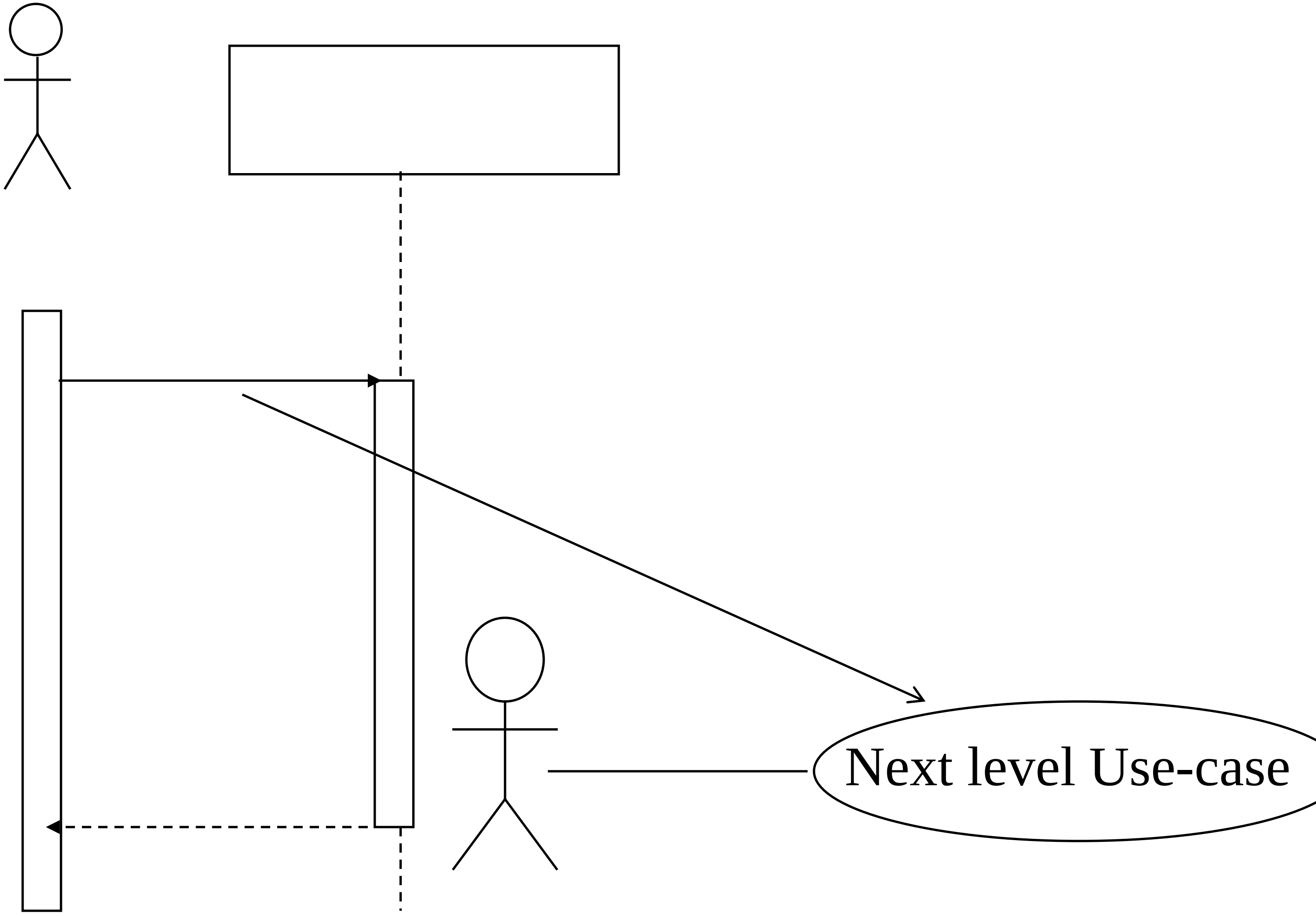
# Combining fragments of sequence diagrams



# More fragments of sequence diagrams

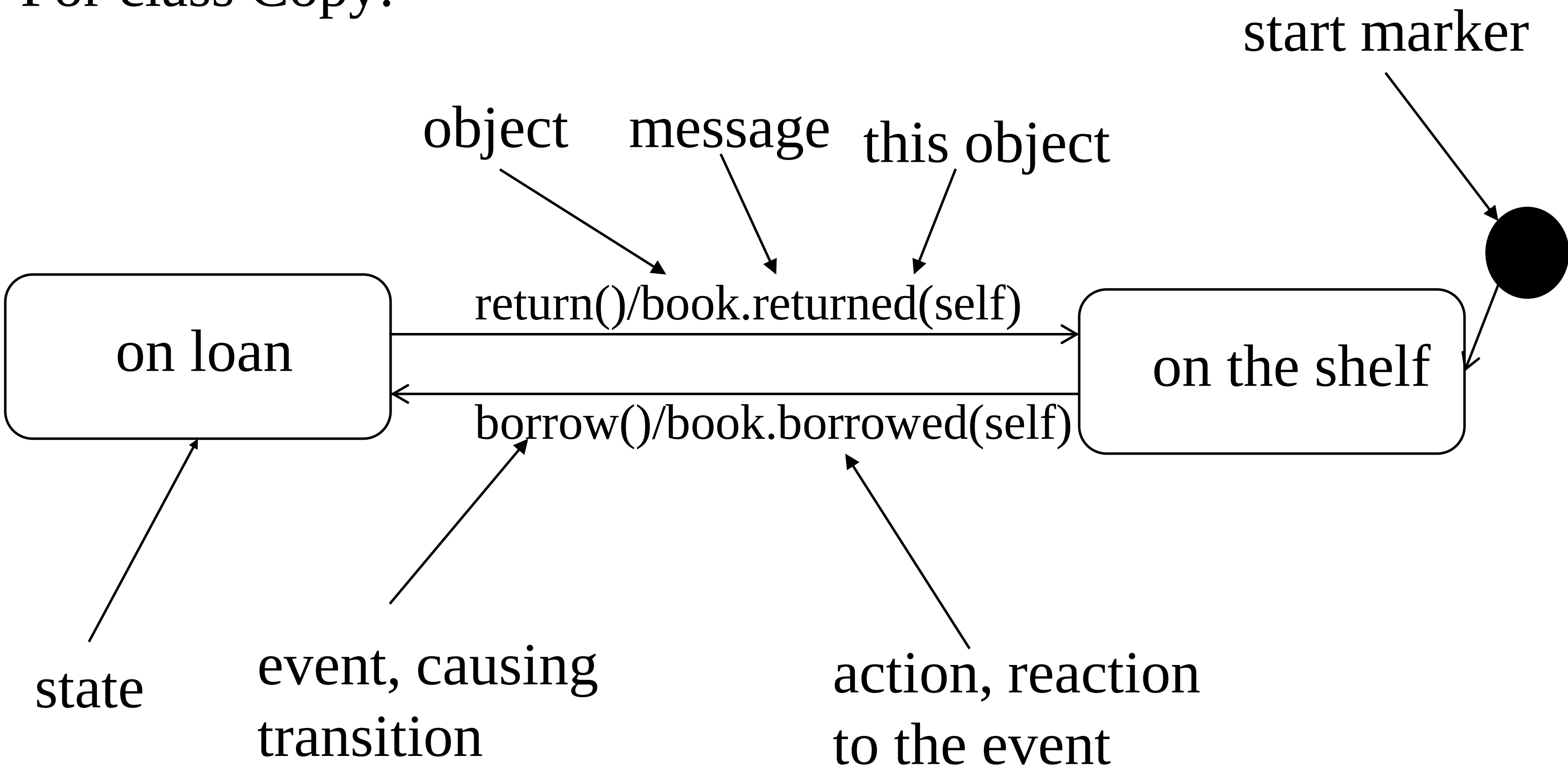


# Continue modelling: next level



# State diagram

For class Copy:



start marker

object

message

this object

return()/book.returned(self)

on loan

on the shelf

borrow()/book.borrowed(self)

state

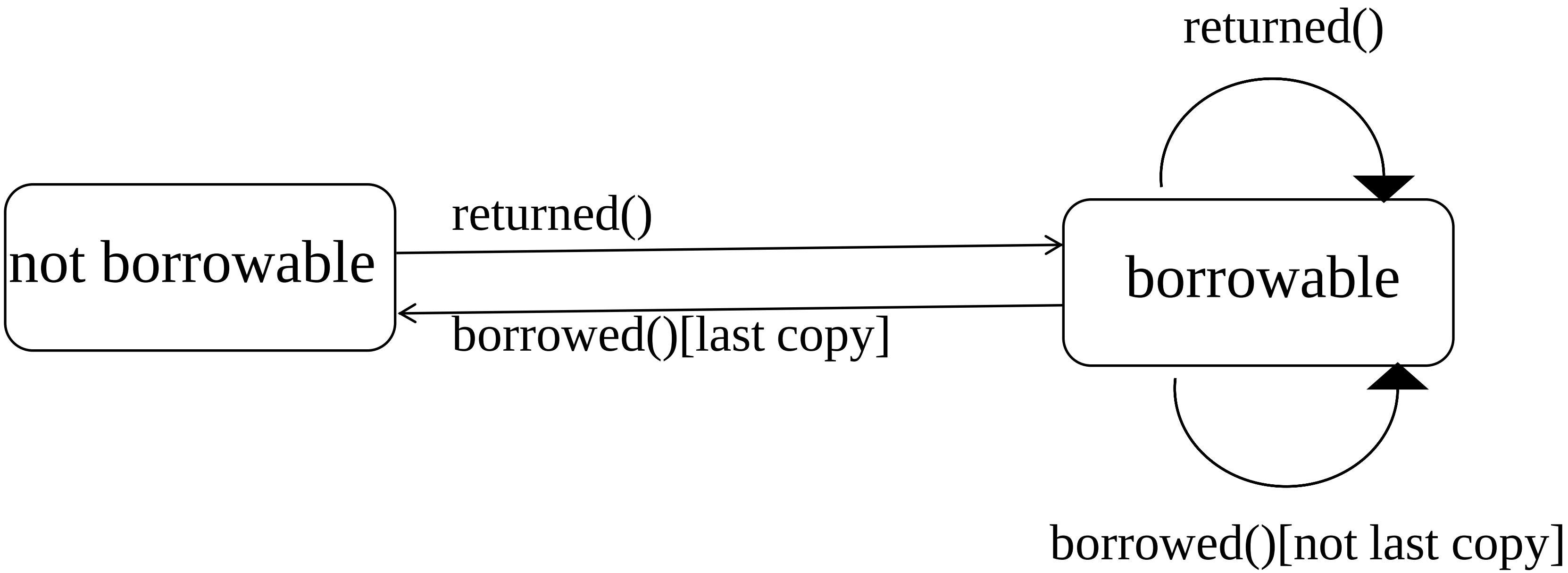
event, causing transition

action, reaction to the event

# State diagram with guards

**Wiht OCL, Object Constraint Language, this becomes very powerful**

For class Book:



not borrowable

returned()

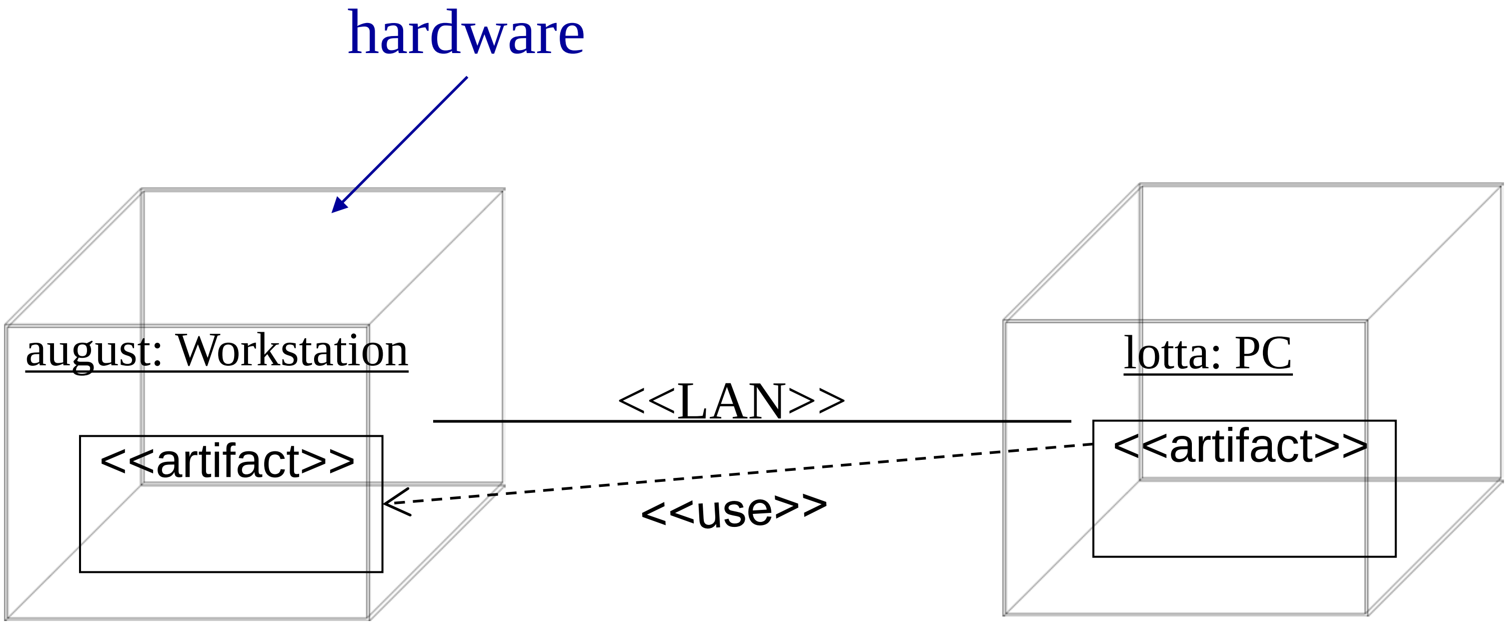
borrowed()[last copy]

borrowable

returned()

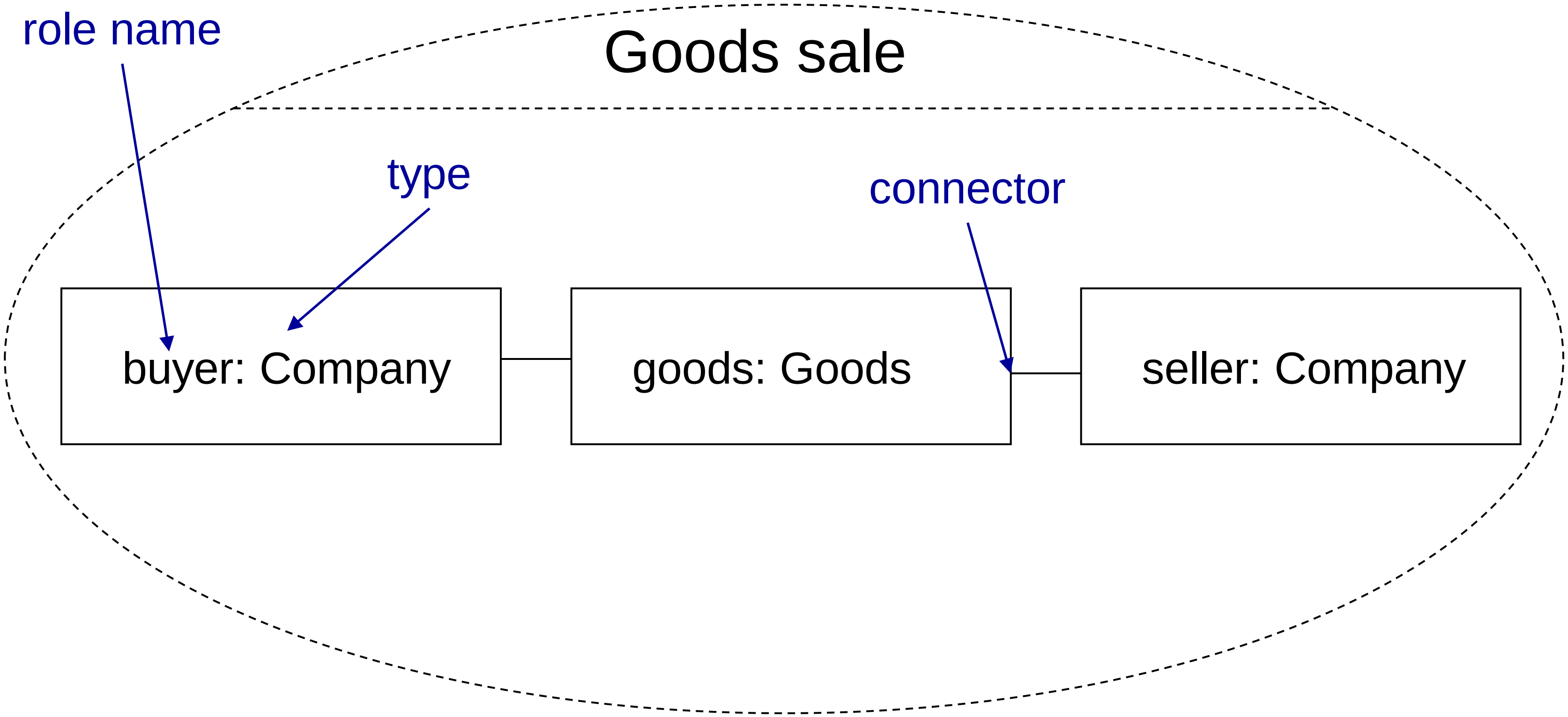
borrowed()[not last copy]

# Deployment diagram

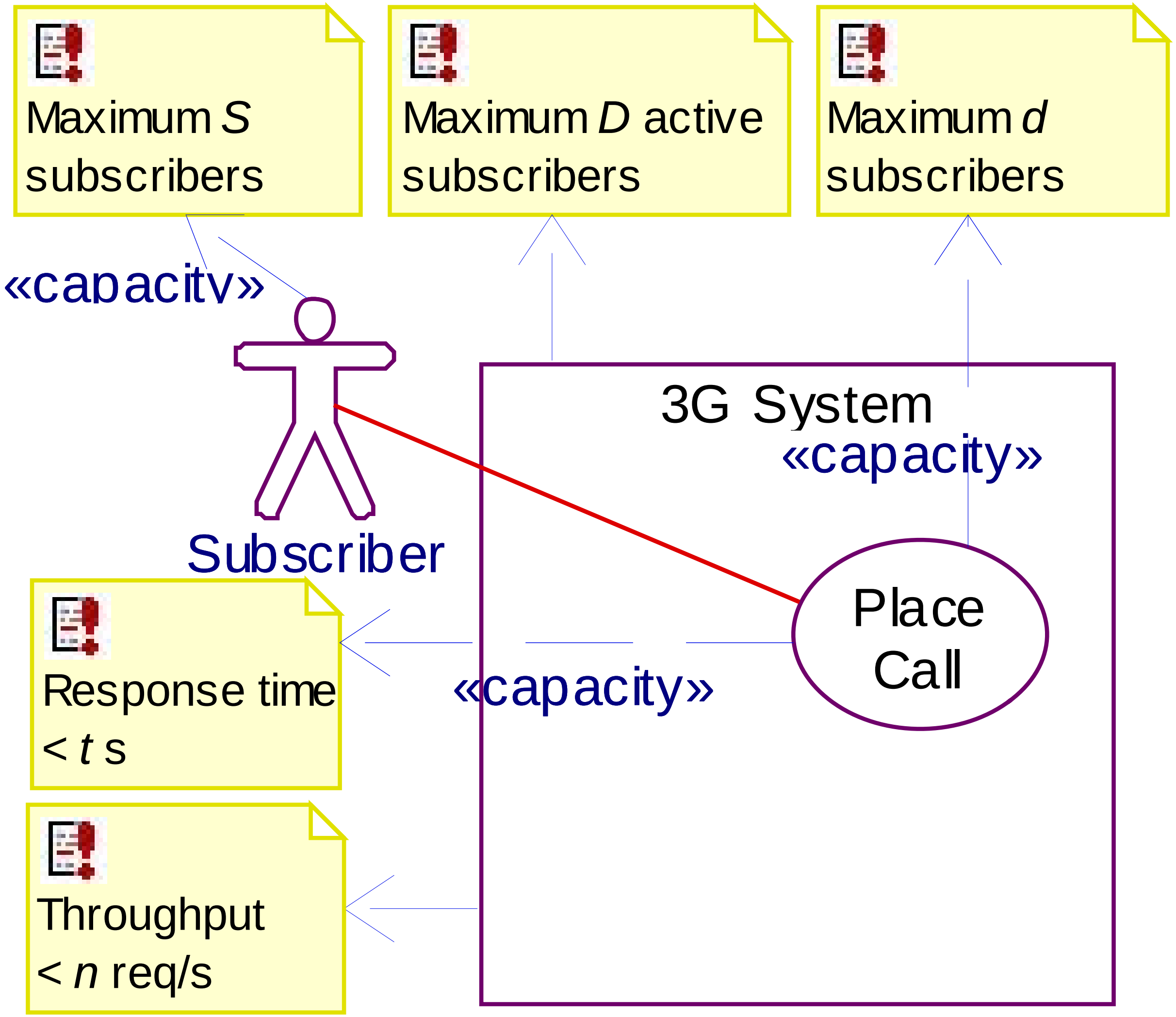


# Collaboration

- Provides a focused view of how instances of classes may collaborate to achieve something, for example, a use-case



# Annotating UML models



Maximum *S* subscribers

Maximum *D* active subscribers

Maximum *d* subscribers

«capacity»

Subscriber

3G System

«capacity»

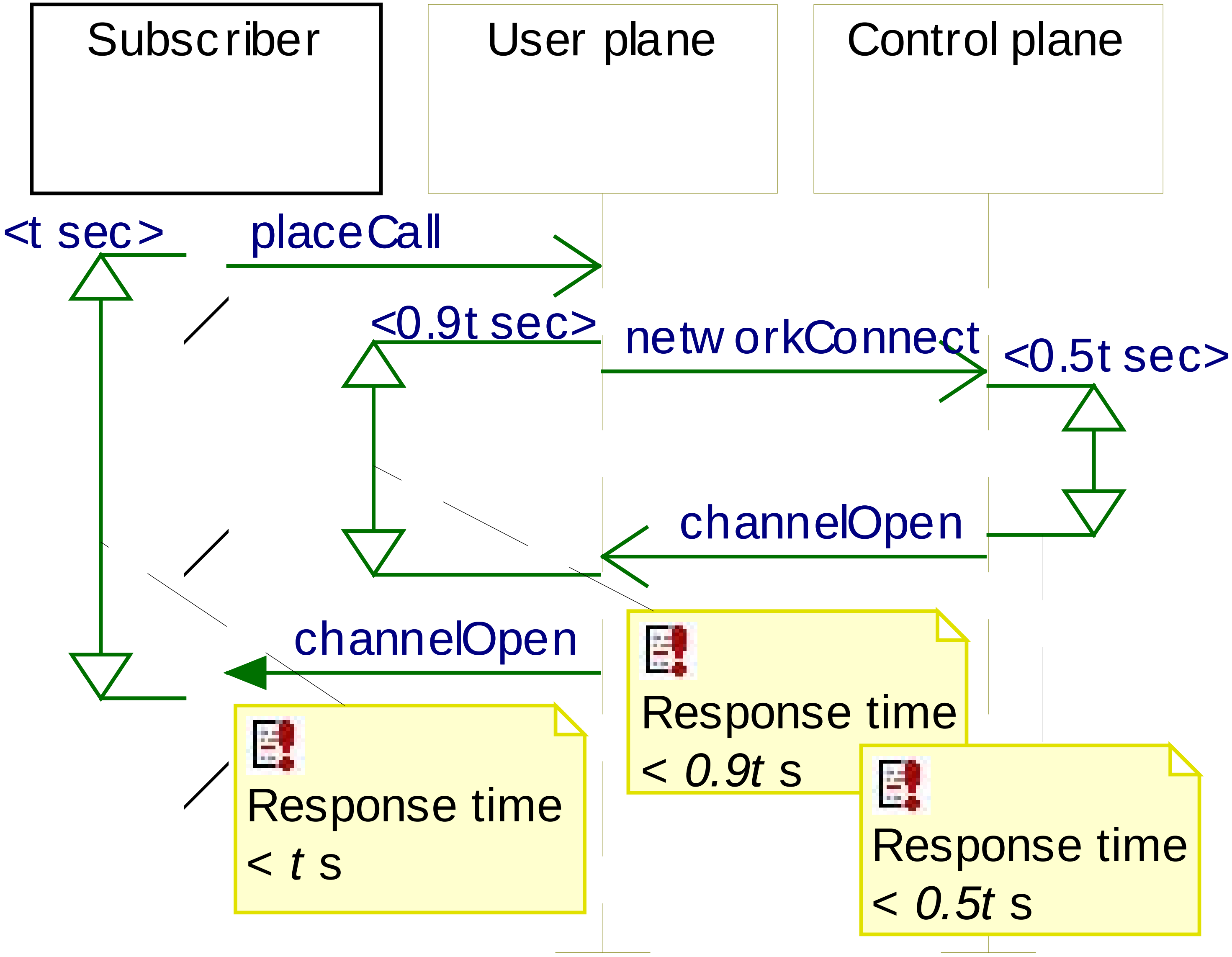
Place Call

«capacity»

Response time < *t* s

Throughput < *n* req/s

# Time constraints in a sequence diagram



Subscriber

User plane

Control plane

<t sec>

placeCall

<0.9t sec>

networkConnect

<0.5t sec>

channelOpen

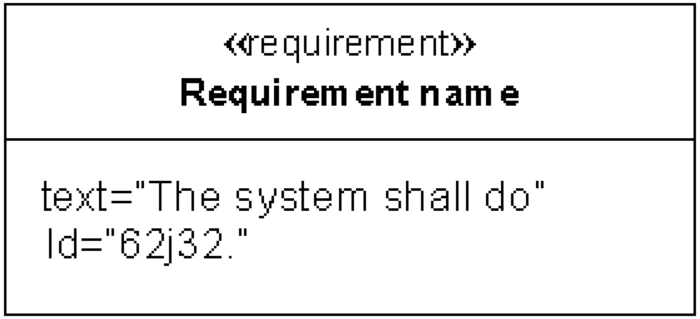
channelOpen

Response time < *0.9t* s

Response time < *t* s

Response time < *0.5t* s

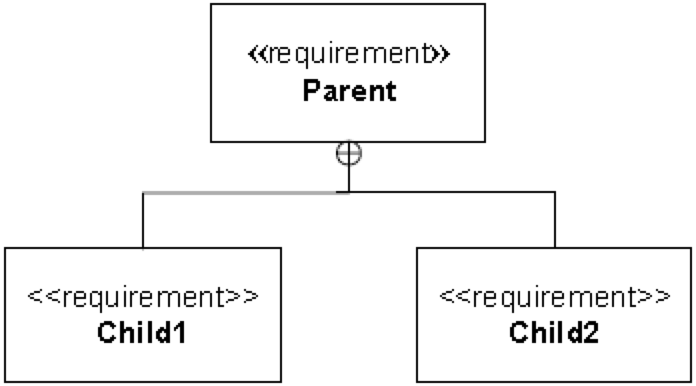
# Requirements in SysML



«requirement»

**Requirement name**

text="The system shall do"
Id="62j32."



«requirement»
**Parent**

<<requirement>>
**Child1**

<<requirement>>
**Child2**

# Table representation

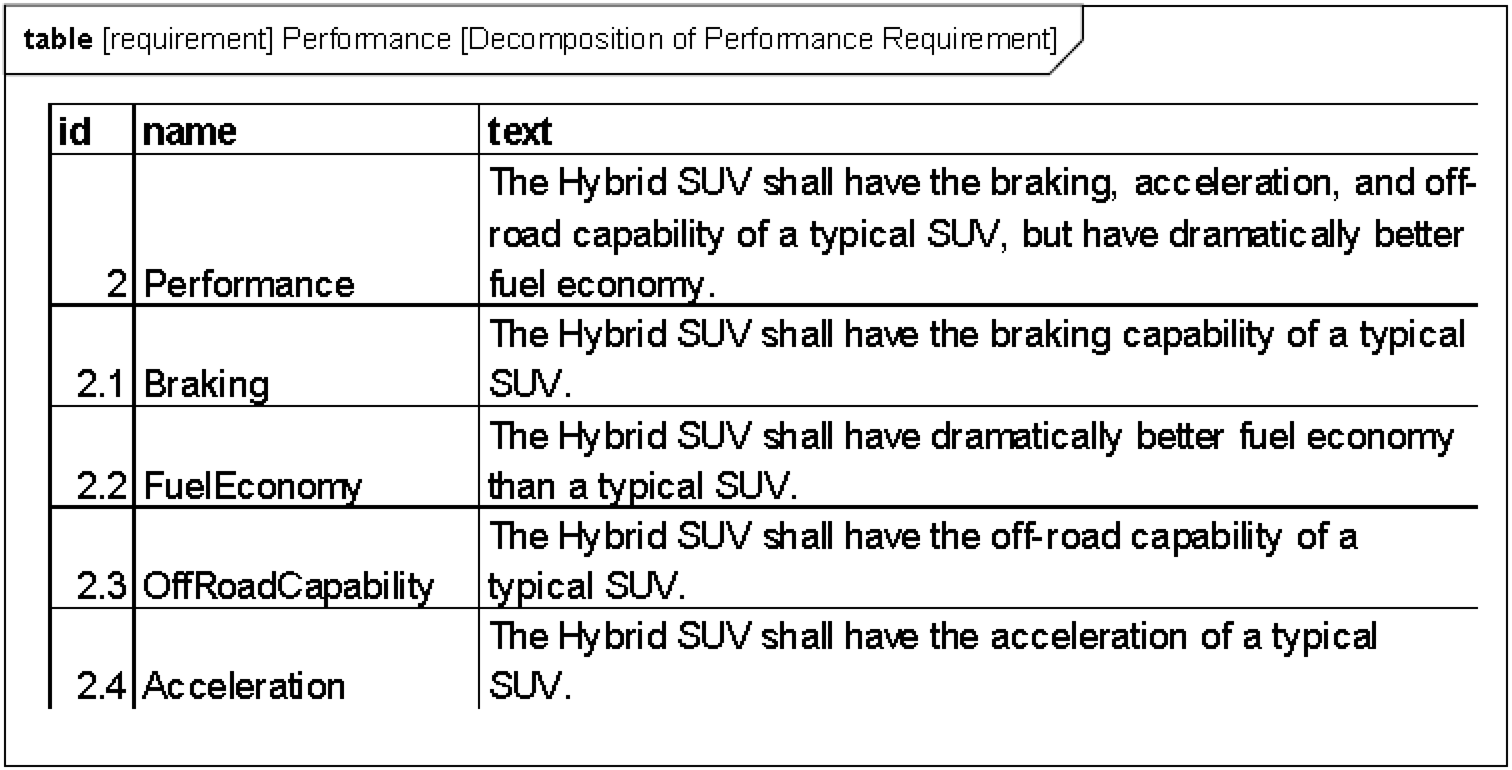
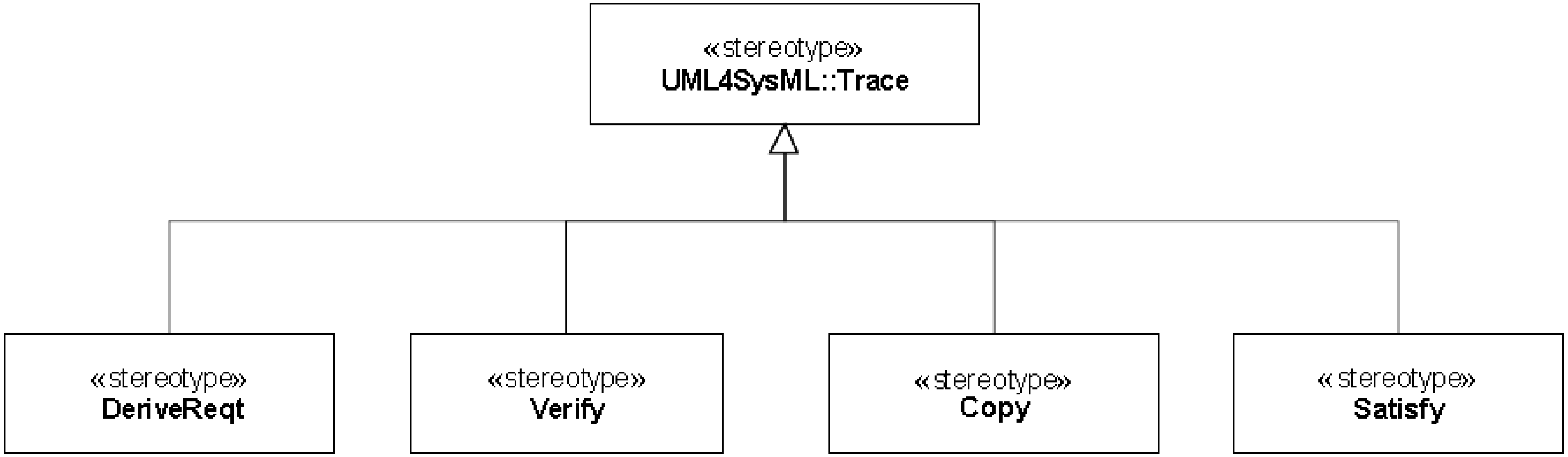
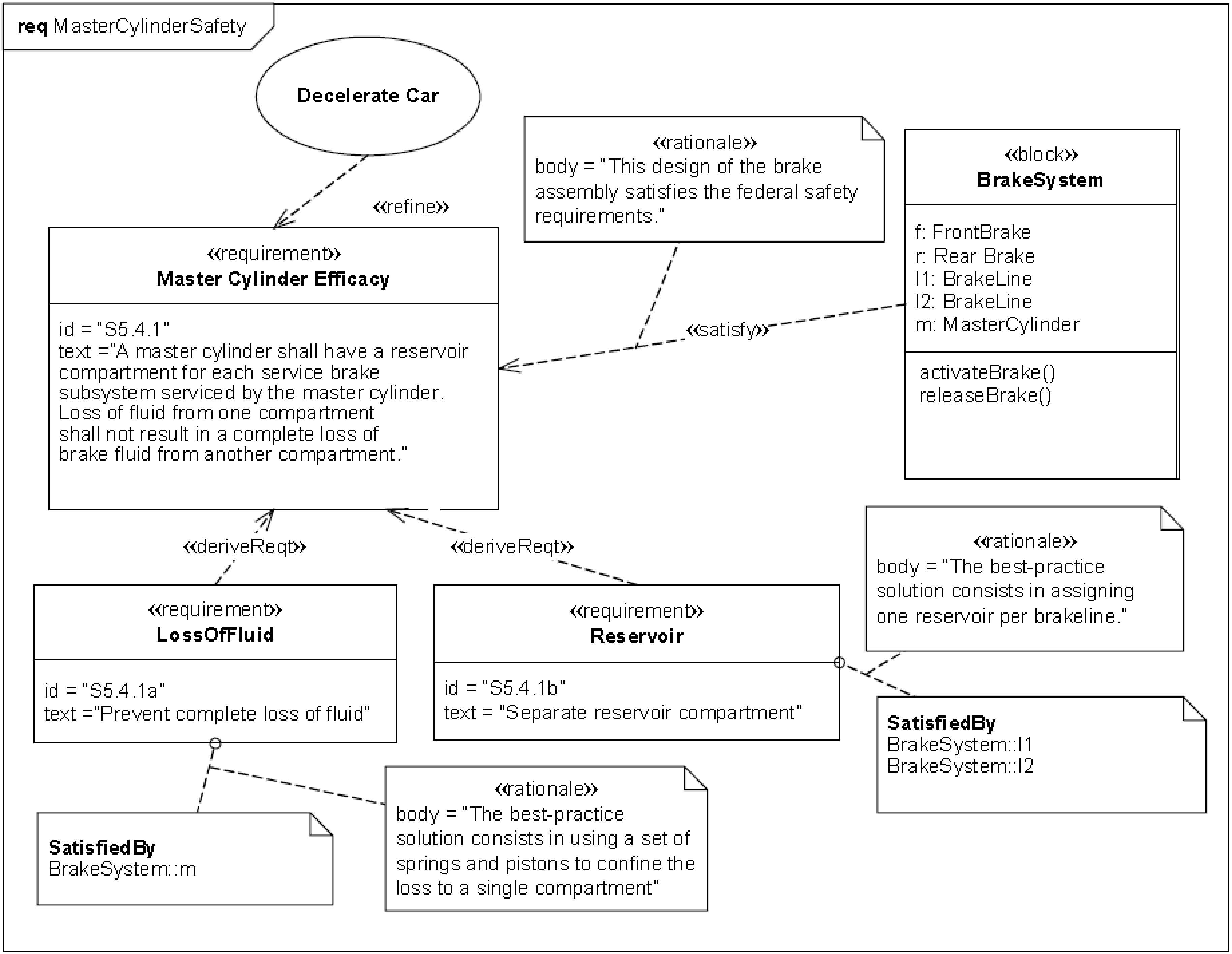


**table** [requirement] Performance [Decomposition of Performance Requirement]

| id | name | text |
|---|---|---|
| 2 | Performance | The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV, but have dramatically better fuel economy. |
| 2.1 | Braking | The Hybrid SUV shall have the braking capability of a typical SUV. |
| 2.2 | FuelEconomy | The Hybrid SUV shall have dramatically better fuel economy than a typical SUV. |
| 2.3 | OffRoadCapability | The Hybrid SUV shall have the off-road capability of a typical SUV. |
| 2.4 | Acceleration | The Hybrid SUV shall have the acceleration of a typical SUV. |

# Relations


«stereotype»
UML4SysML::Trace
«stereotype»
DeriveReqt
«stereotype»
Verify
«stereotype»
Copy
«stereotype»
Satisfy



**req** MasterCylinderSafety

**Decelerate Car**

«refine»

«requirement»
**Master Cylinder Efficacy**
id = "S5.4.1"
text ="A master cylinder shall have a reservoir compartment for each service brake subsystem serviced by the master cylinder. Loss of fluid from one compartment shall not result in a complete loss of brake fluid from another compartment."

«rationale»
body = "This design of the brake assembly satisfies the federal safety requirements."

«satisfy»

«block»
**BrakeSystem**
f: FrontBrake
r: Rear Brake
l1: BrakeLine
l2: BrakeLine
m: MasterCylinder
activateBrake()
releaseBrake()

«deriveReqt»

«deriveReqt»

«requirement»
**LossOfFluid**
id = "S5.4.1a"
text ="Prevent complete loss of fluid"

«requirement»
**Reservoir**
id = "S5.4.1b"
text = "Separate reservoir compartment"

«rationale»
body = "The best-practice solution consists in assigning one reservoir per brakeline."

**SatisfiedBy**
BrakeSystem::l1
BrakeSystem::l2

**SatisfiedBy**
BrakeSystem::m

«rationale»
body = "The best-practice solution consists in using a set of springs and pistons to confine the loss to a single compartment"