

UML

een overzicht

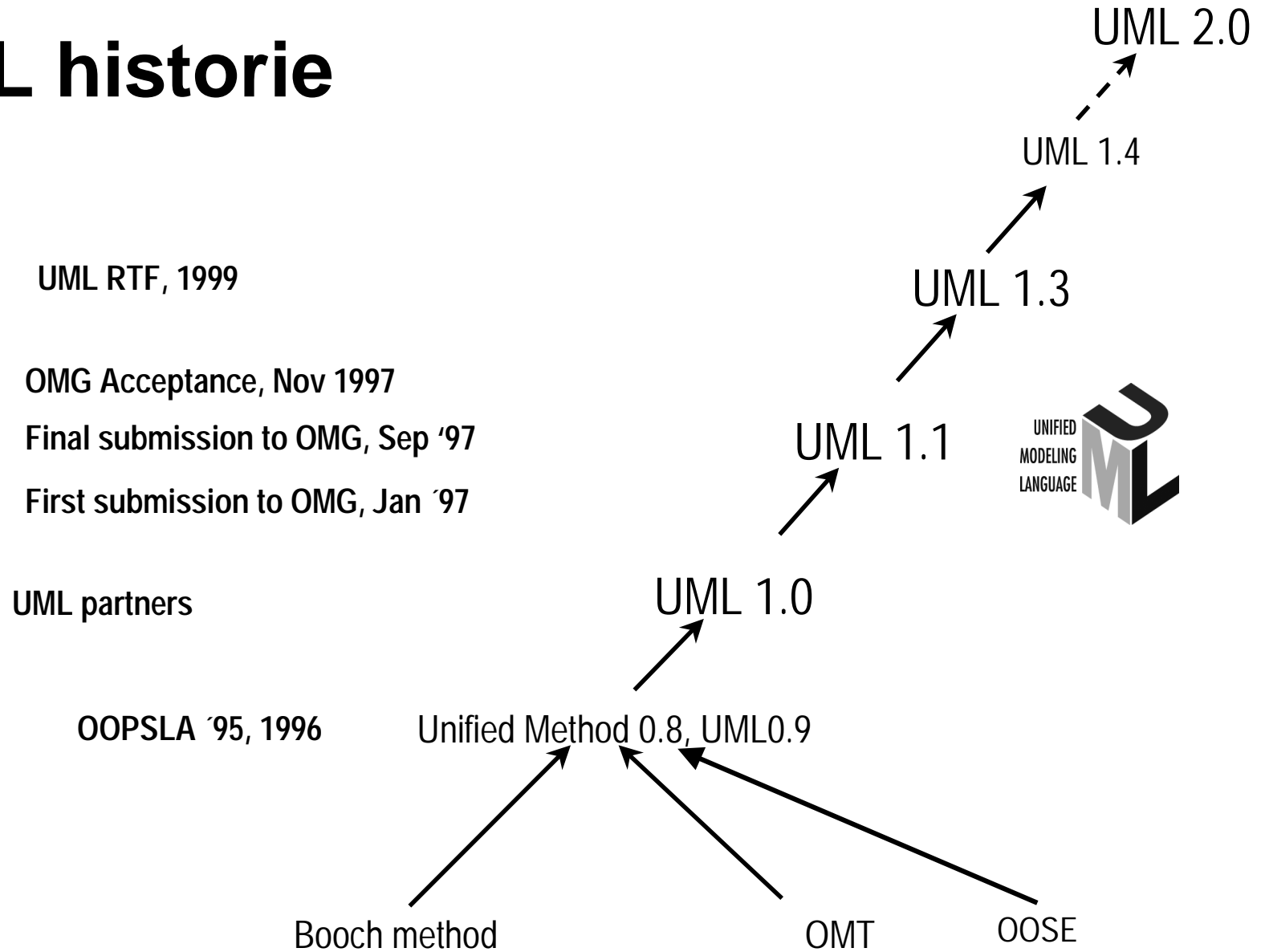
Gert Florijn
florijn@serc.nl

Wat is UML?

Unified Modeling Language

- The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
- The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.
- The UML represents the culmination of best practices in practical object-oriented modeling. The UML is the product of several years of hard work, in which we focused on bringing about a unification of the methods most used around the world, the adoption of good ideas from many quarters of the industry, and, above all, a concentrated effort to make things simple.
 - Unified Modeling Language Specification 1.4, september 2001

UML historie



Wat is UML?

Unified Modelling Language

- Grafische notatietaal voor het specificeren, construeren, visualiseren en documenteren van (complexe) software systemen
- Niet (inherent) formeel
- Toepasbaar op meerdere niveaus (analyse vs. implementatie)
- Onafhankelijk van werkwijze of programmeertaal
- Gericht op OO systemen
- Integratie van bestaande notaties
- Gezamenlijk resultaat van vele partijen
- Object Management Group (OMG) Standaard
- Ondersteund door tools, integratie via XML (XMI)

UML Concepten

Een model is een representatie van een domein of systeem

- op een bepaald abstractieniveau, bijv. analyse, design, implementatie

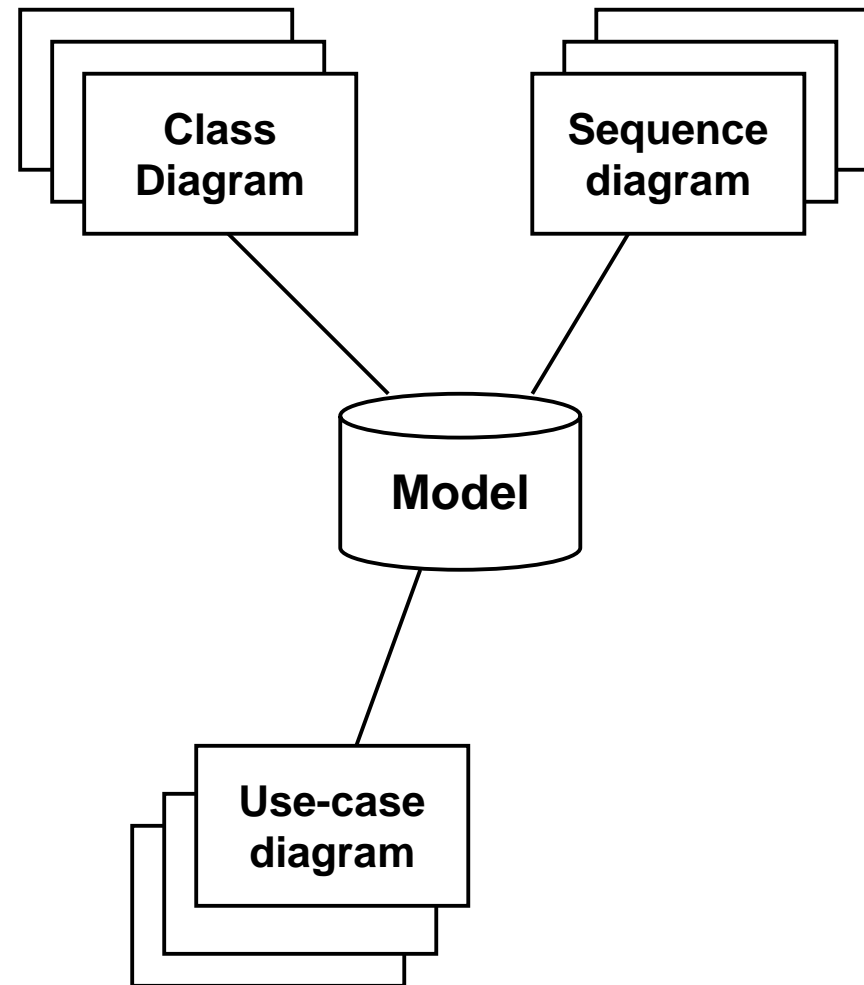
Modellen bestaan uit model elementen

- class, node, association, ...
- gedefinieerd in meta-model

Een diagram toont een bepaald perspectief van een model

- partieel, consistent

View elementen zijn grafische weergaven van model elementen in een diagram



UML Extensiemechanismen

Constraints

- Beperkingen die betrekking hebben op één type element
- Kunnen worden uitgedrukt in OCL (Object Constraint Language)
- voorbeeld: { ordered }

Properties (tagged values)

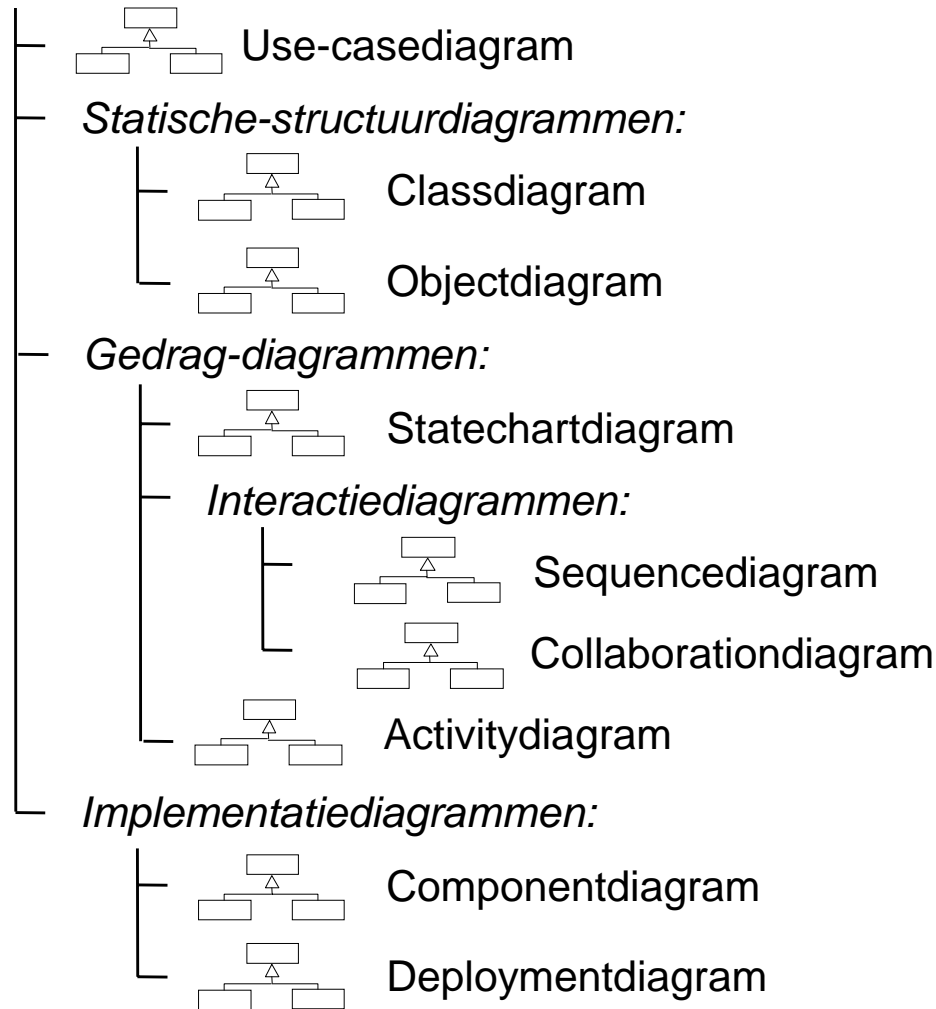
- Eigenschappen van een element
- voorbeeld: { Author = "Danny Greefhorst" }

Stereotypes

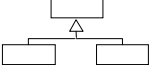
- Semantisch specialiseren van bestaande model elementen
- voorbeeld: <<interface>>, <<source>>

UML Diagrammen

UML-diagrammen:



cursief : Diagramsoort

 : Concreet UML-diagram

Use-case diagram

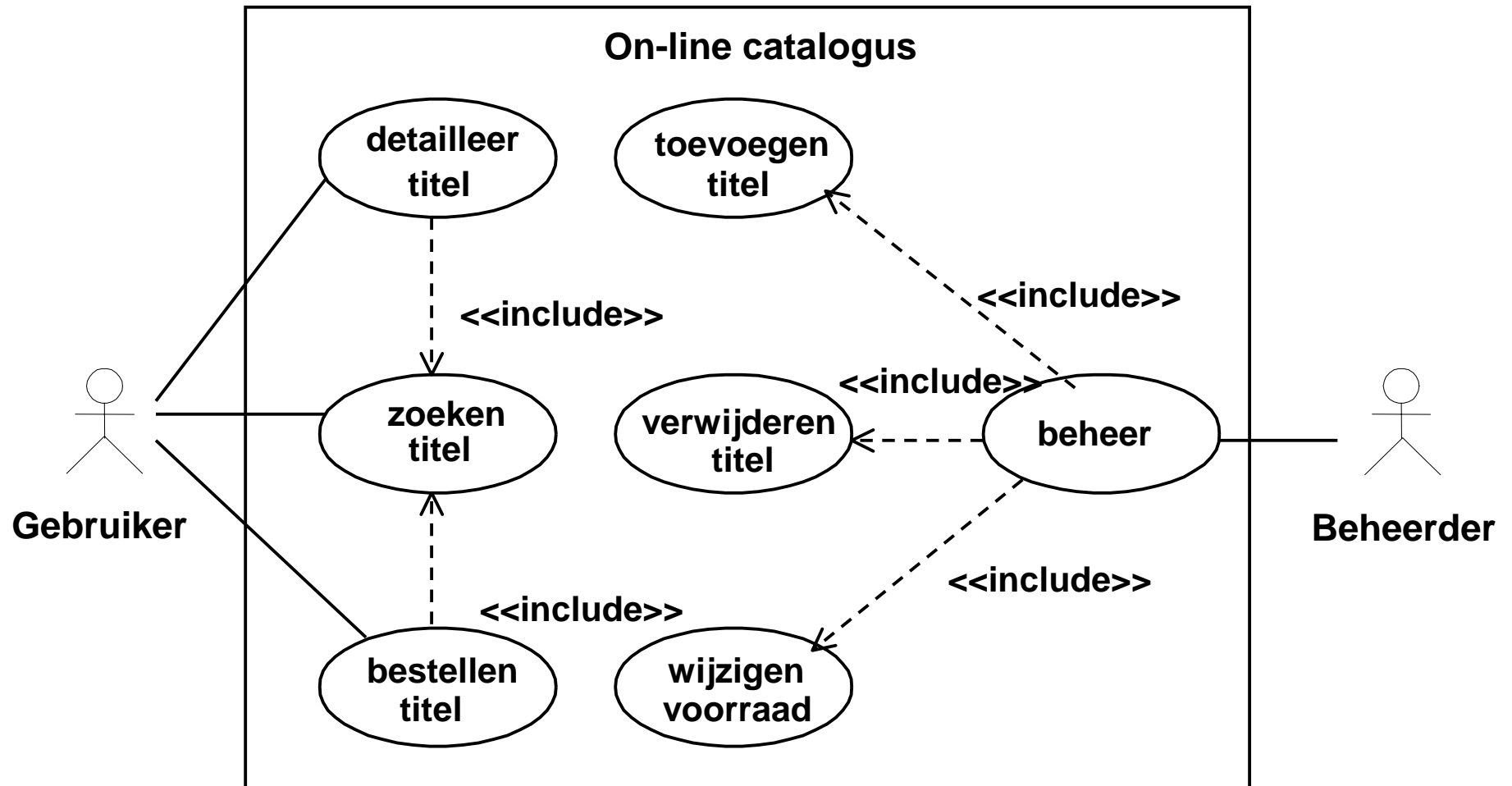
Use-case

- interactie tussen gebruiker en systeem
- levert waarneembaar resultaat op voor de gebruiker
- doelgericht
- beschreven in use-case diagram en tekst

Use-case diagram

- systeem: de te realiseren functionaliteit
- actoren: externe eenheden die met het systeem communiceren
- use cases: functionaliteit zoals gezien door de gebruiker
- relaties: relaties tussen actoren en use-cases, en use-cases onderling

Voorbeeld Use-case diagram



Use-Case relaties

Include dependency

- Een use-case maakt in zijn geheel gebruik van een andere use-case

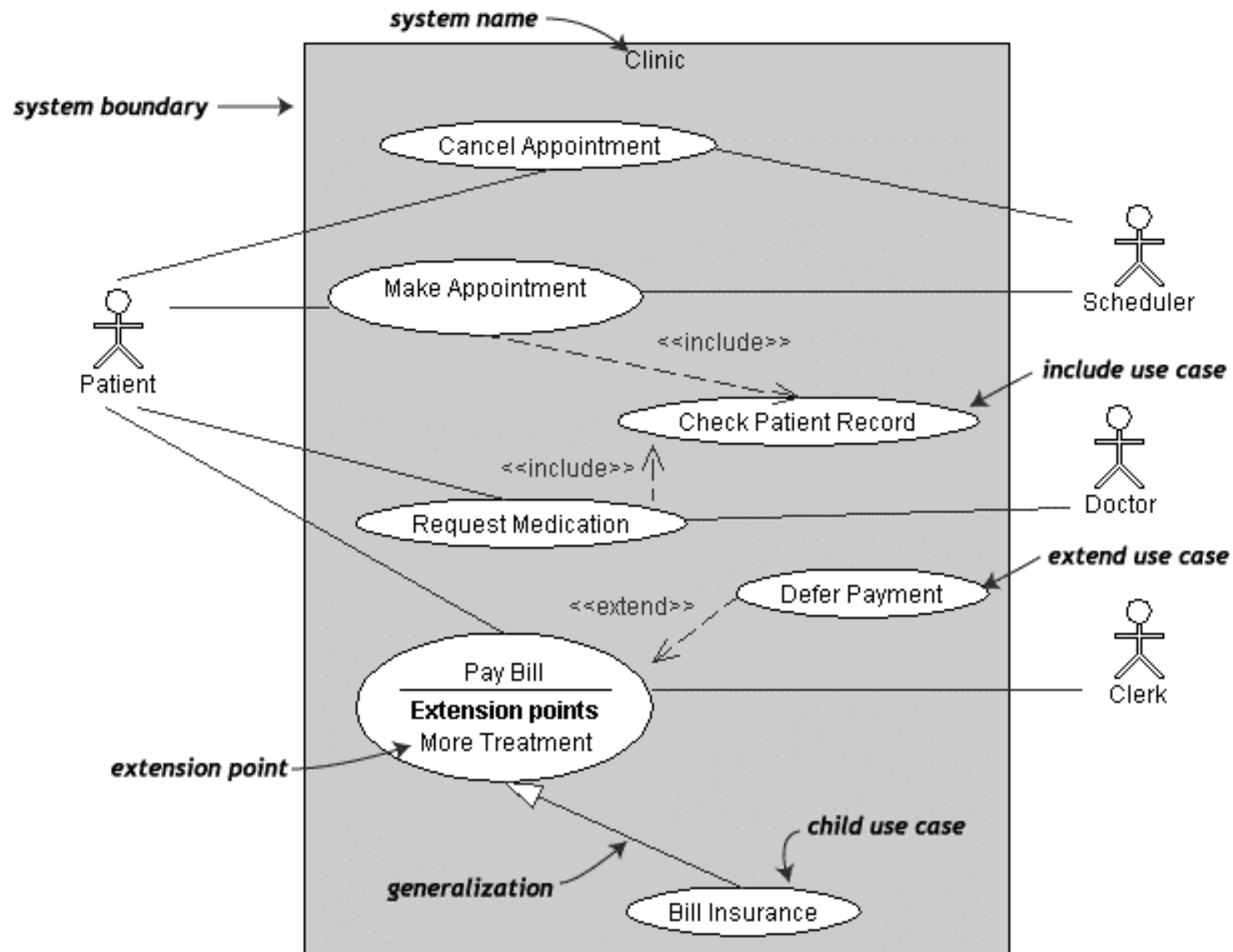
Generalization

- Een use-case is een variatie (uitbreiding) op een bestaande use-case

Extend dependency

- Een use-case is een gecontroleerde variatie (uitbreiding) op een bestaande use-case
- Extension points

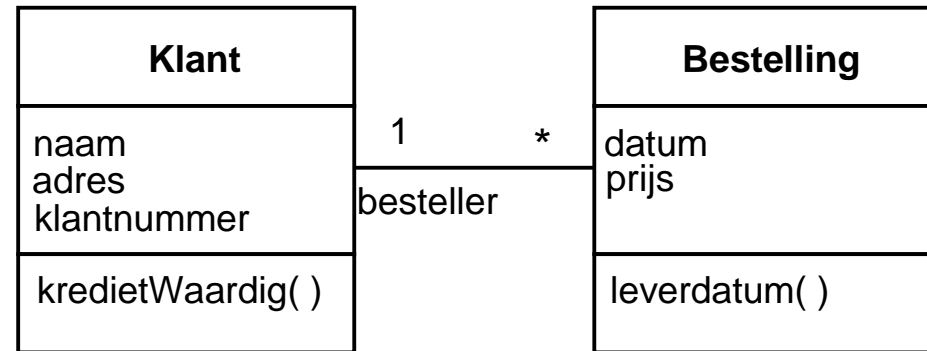
Use-Case Diagram



Class diagrammen

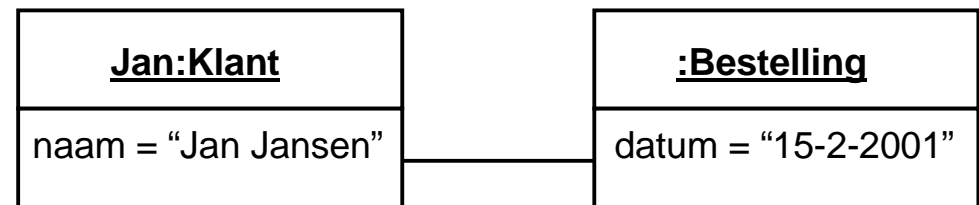
Class diagram

- Beschrijft de statische structuur van een domein of systeem in termen van klassen, hun eigenschappen en hun onderlinge relaties



(Object diagram)

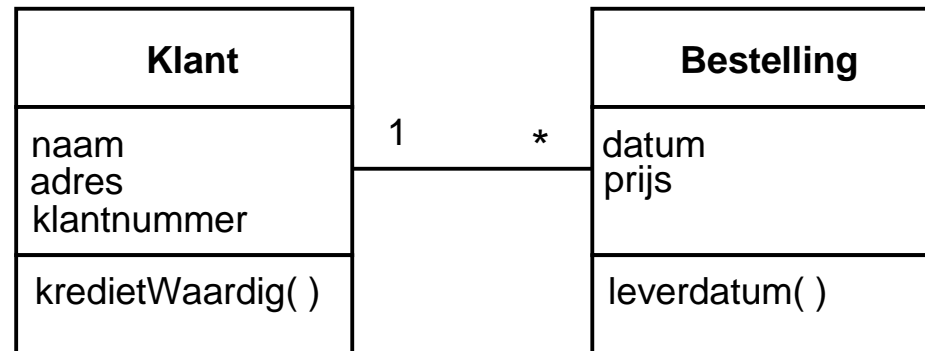
- Illustreert een clasdiagram door (voorbeeld) objecten en hun eigenschappen en relaties te tonen.



Pas op

De interpretatie van een class diagram hangt af van het abstractieniveau of perspectief dat wordt gehanteerd

- Conceptueel
- Specificatie
- Implementatie



Klassen

Attributen

- *visibility name: type = initialvalue*

Operaties

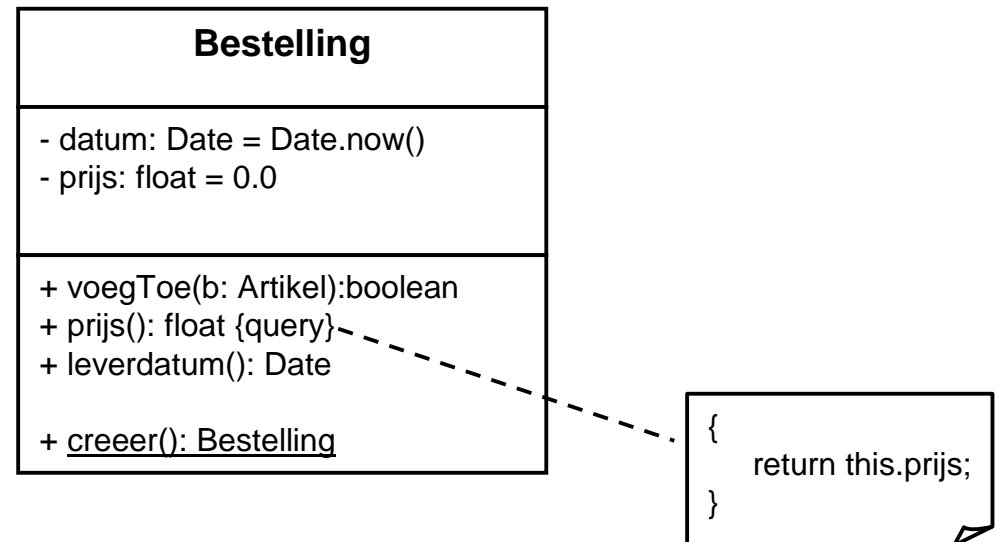
- *visibility name(params): returntype*

Visibility

- - private
- + public
- # protected
- ~ package

Scope

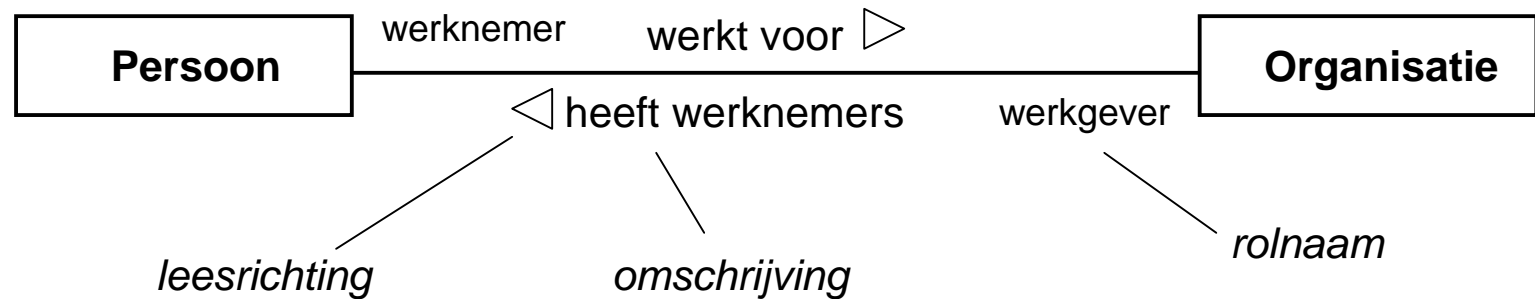
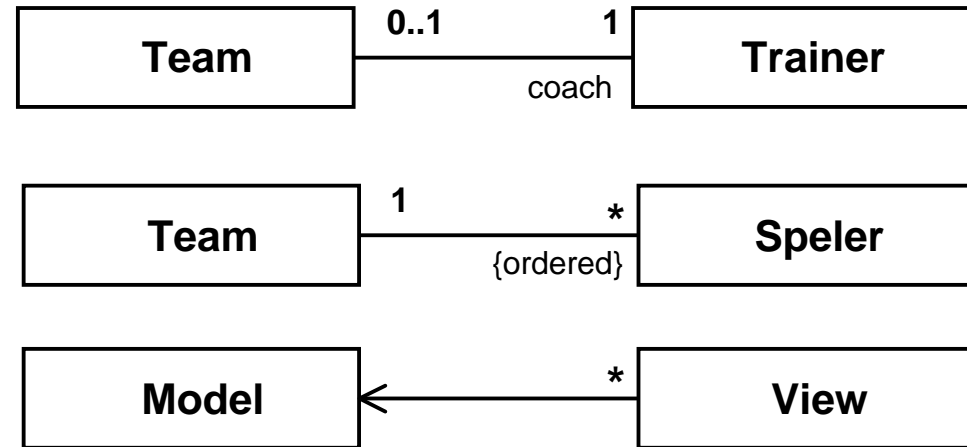
- Underline voor static/class scope



Relaties

Associaties

- Rollen
- Multipliciteit
- Navigeerbaarheid
- Constraints



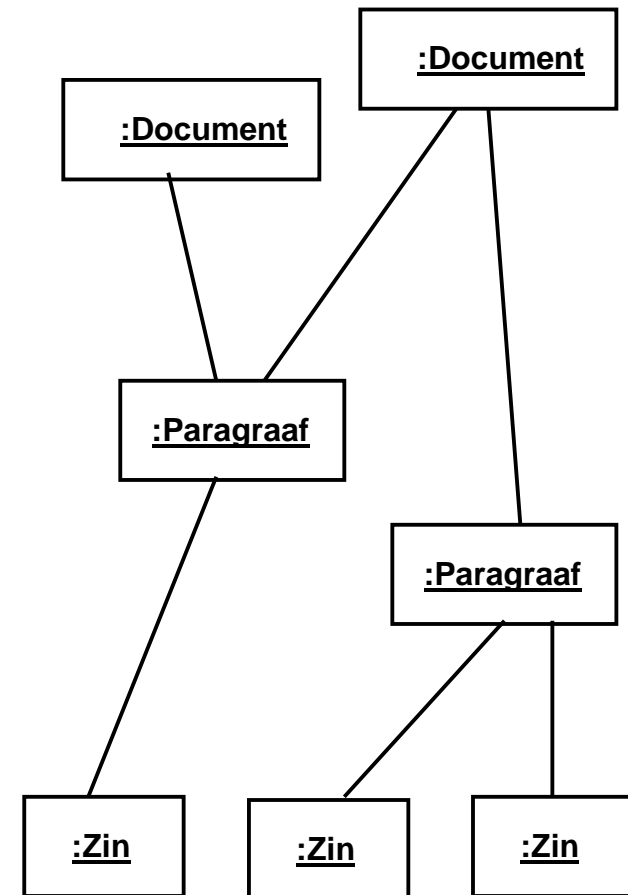
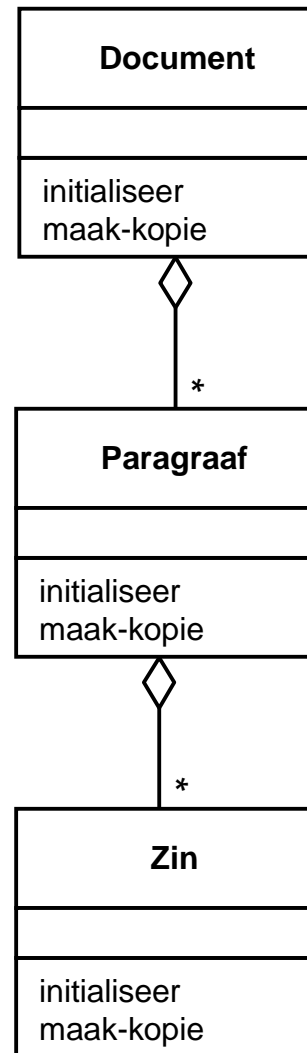
Relaties

Aggregatie

- “part of” relaties
- Bepaalde operaties propageren

Compositie

- Bestaan van onderdeel afhankelijk van geheel
- Onderdelen worden niet gedeeld
- (ruit is gevuld)



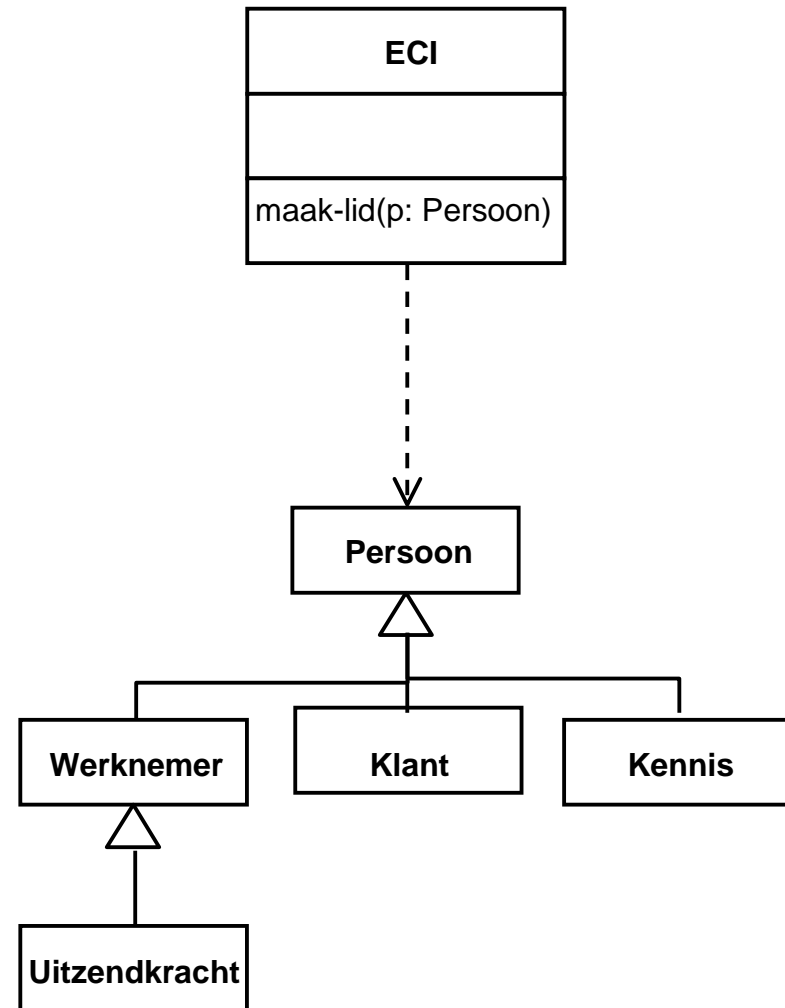
Relaties

Generalisatie

- Super-type vs. sub-type
- Substitueerbaarheid
- Sub-classing(?)

Dependency

- Afhankelijkheidsrelatie



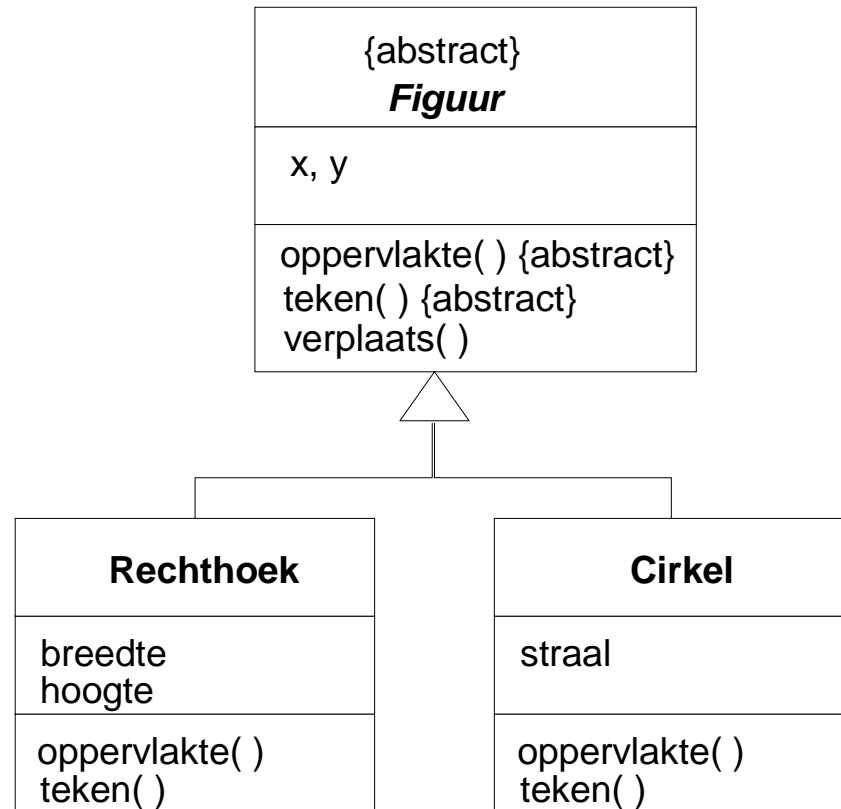
Abstracte klassen

Abstracte klasse

- Definieert interface
- Bevat geen of onvolledige implementatie
- Geen instanties
- Aangegeven met {abstract}

Concrete (implementatie) klasse

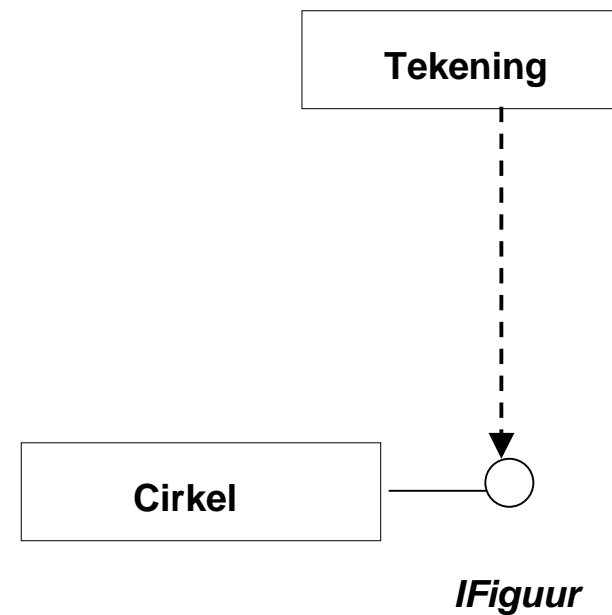
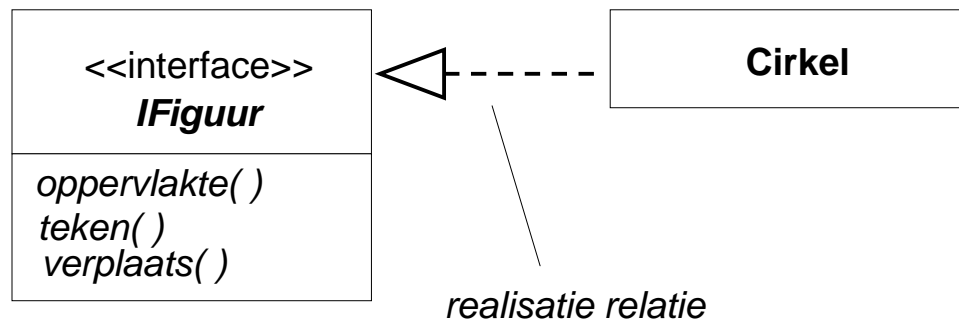
- Biedt implementatie
- Wel instanties



Interface

Kenmerken

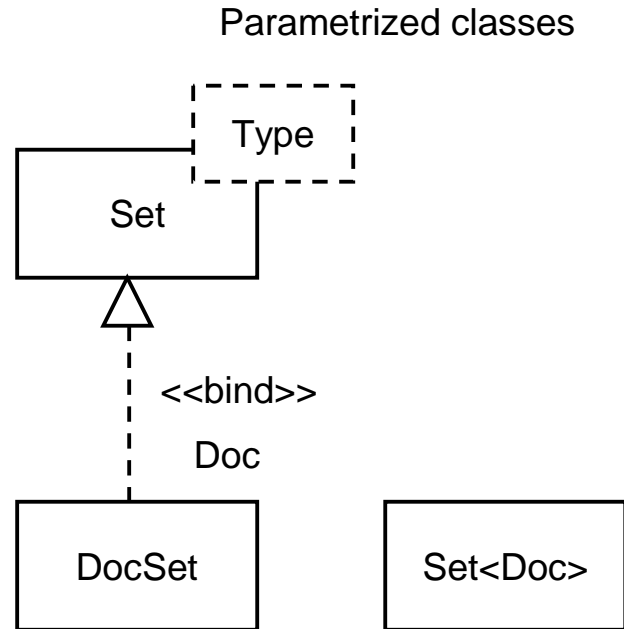
- Definieert operaties die klassen kunnen realiseren
- Vergelijkbaar met (pure) abstracte klasse
- Gemarkeerd met <<interface>> stereotype
- Gerealiseerd middels realizes relatie
- Weergegeven middels “lollypop”



Meer Klassen

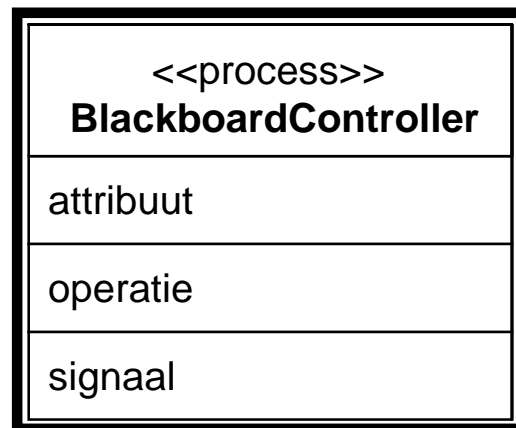
Parametrized classes

- Vgl. C++ template classes

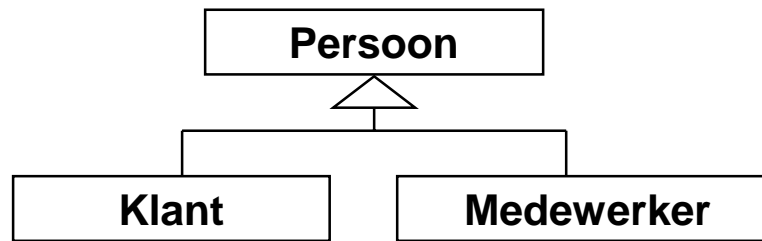


Active classes

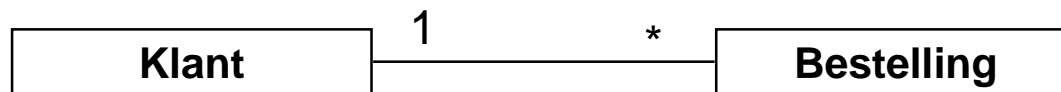
- Eigen thread of control
- <<process>>
- <<thread>>



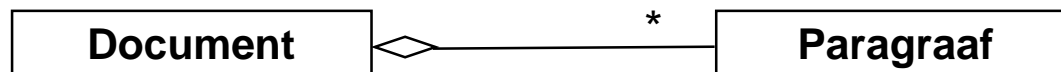
Relaties



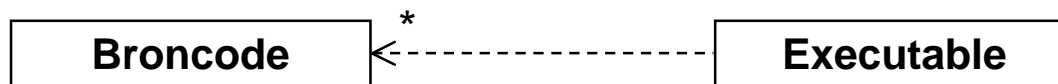
Specialisatie



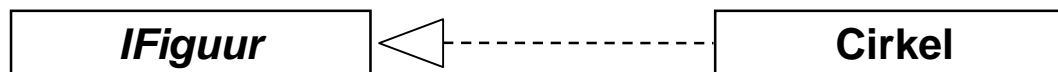
Associatie



Aggregatie



Dependency

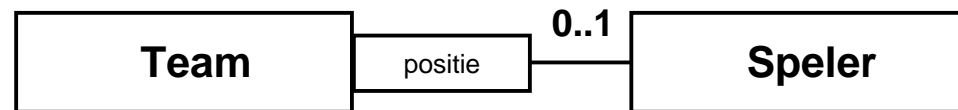


Realisatie

Meer relaties

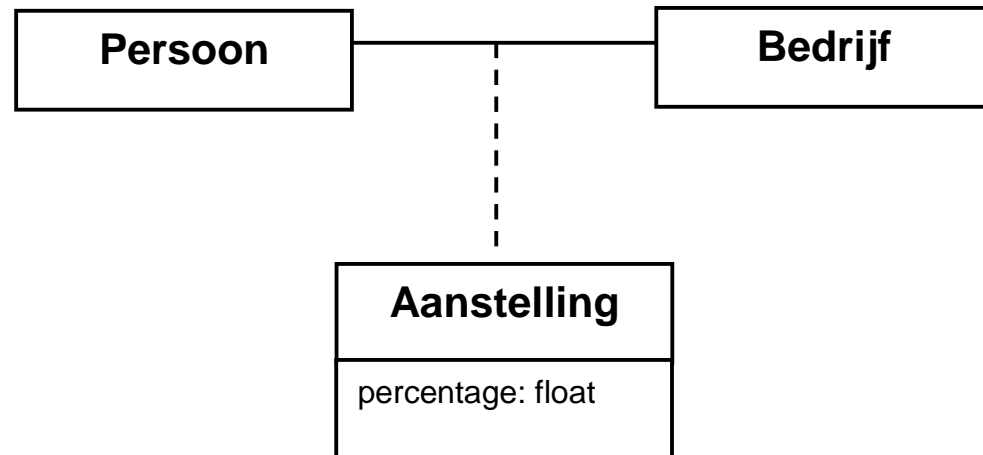
Qualified association

- Reduceert multipliciteit door extra sleutel

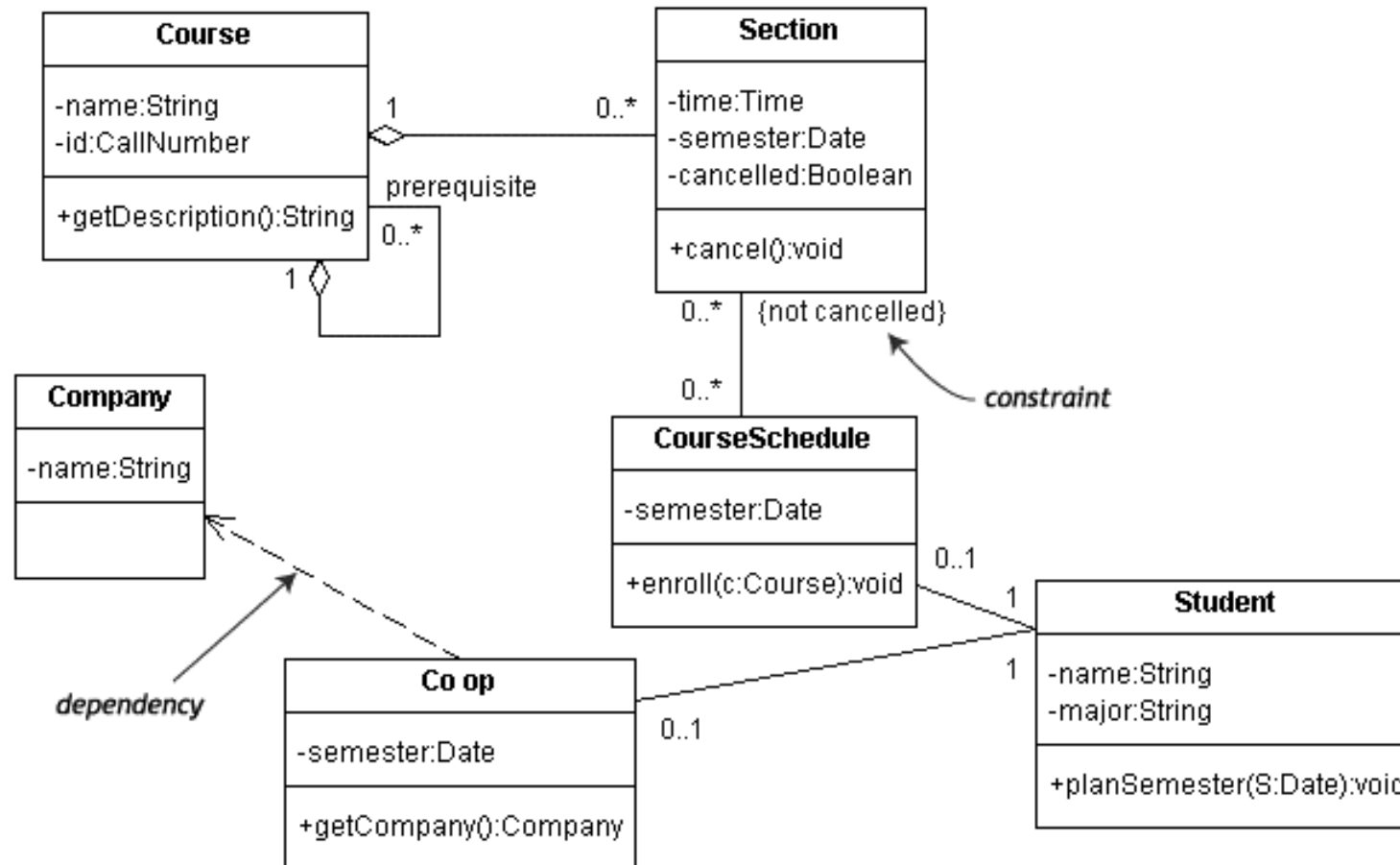


Association class

- Voeg attributen en operaties toe aan associatie



Relaties en constraints



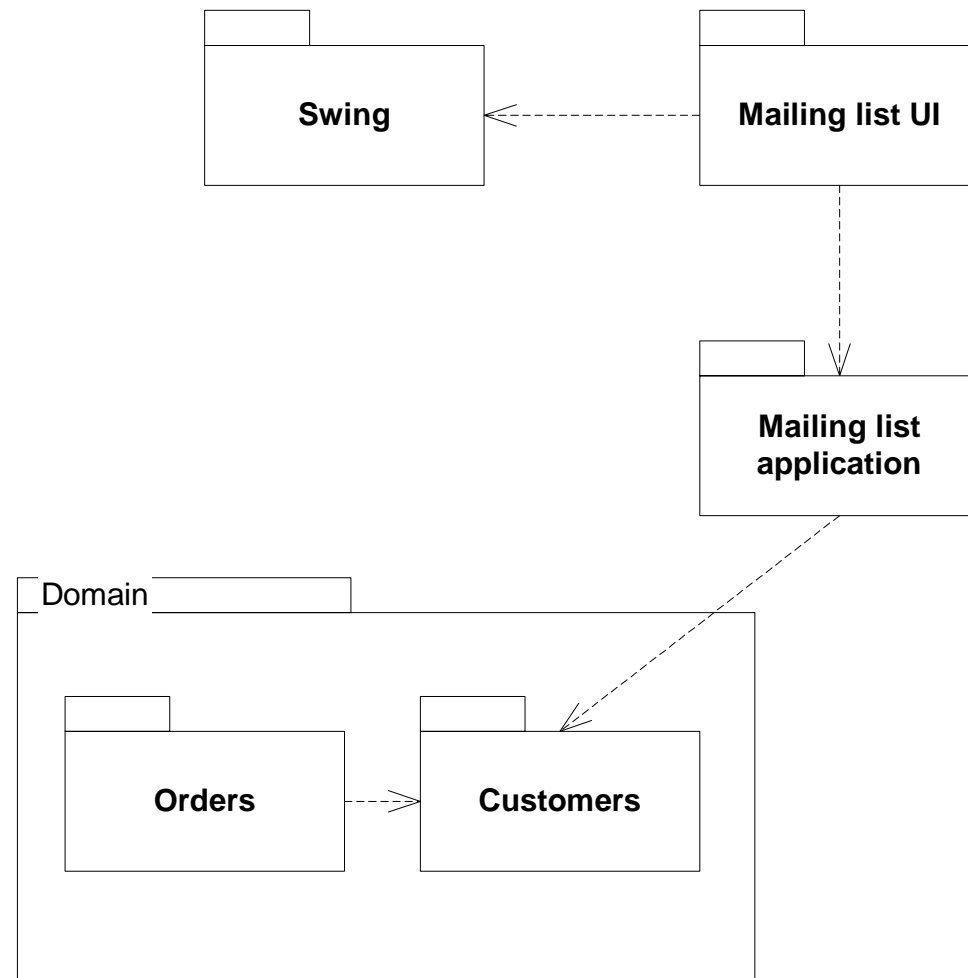
Packages

Karakteristieken

- Package is een groepering van model-elementen
- Gebruikt voor organisatie van grote modellen
- Packages kunnen genest zijn
- Een model element zit in precies 1 package

Relaties (dependencies)

- <<access>>
- <<import>>



Dynamische diagrammen

Statechartdiagram

- toestanden, overgangen en gedrag van een object

Sequencediagram

- interactie tussen objecten, nadruk op de berichten en volgorde van berichten

Collaborationdiagram

- interactie tussen objecten, nadruk op (statische) relaties tussen communicerende objecten

Activitydiagram

- geeft de activiteiten weer die nodig zijn voor het uitvoeren van een taak, bijv. een use-case

Sequence diagrammen

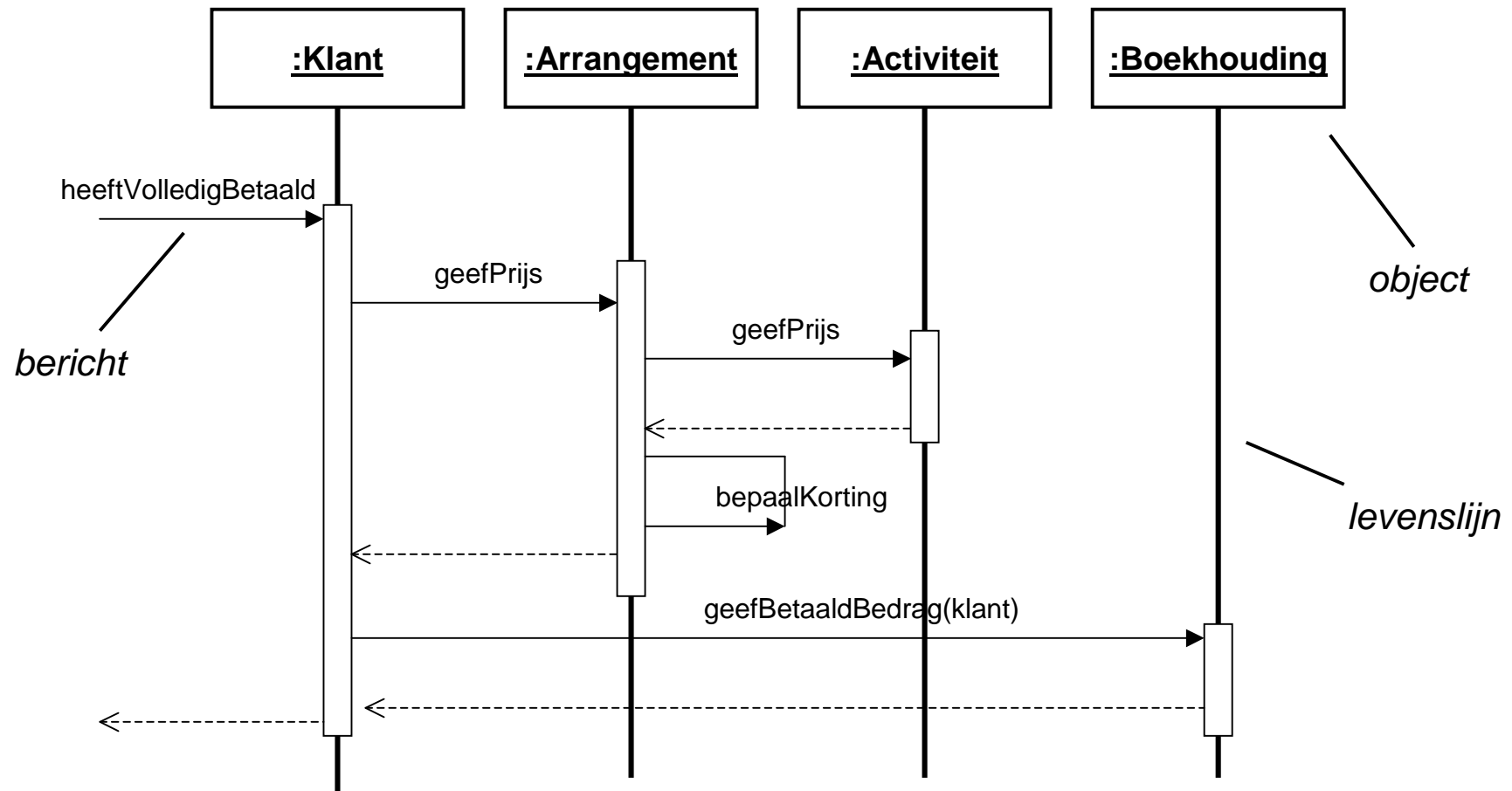
Beschrijven communicatie tussen objecten

- berichten
- volgorde van berichten

Gebruikt voor

- inzicht in gedrag van systeem
- welke objecten zijn nodig voor het uitvoeren van een dienst?
- beschrijving van scenario van use-case
- controle van 'toegangspaden'

Voorbeeld sequence diagram



Berichttypen

Simpel

- precieze betekenis ongedefinieerd
- Sinds 1.4: asynchroon!

Synchroon

- zender wacht op antwoord

Asynchroon

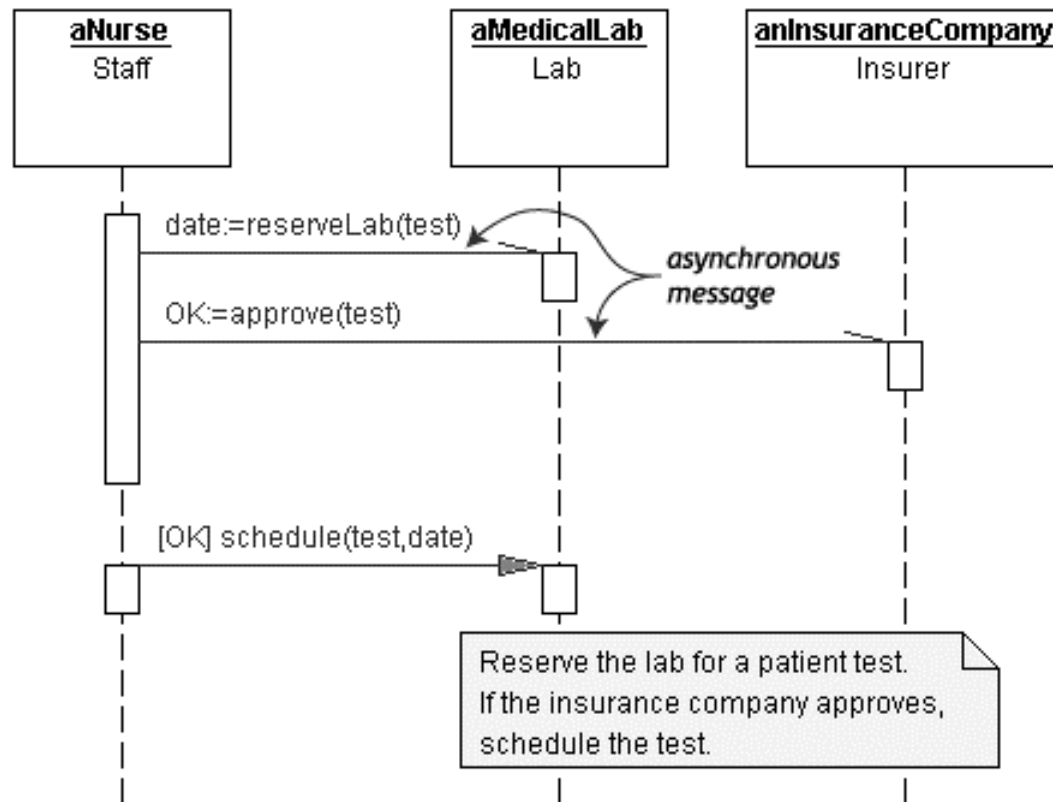
- zender wacht niet op antwoord
- Sinds 1.4: verdwenen

Terugkeer

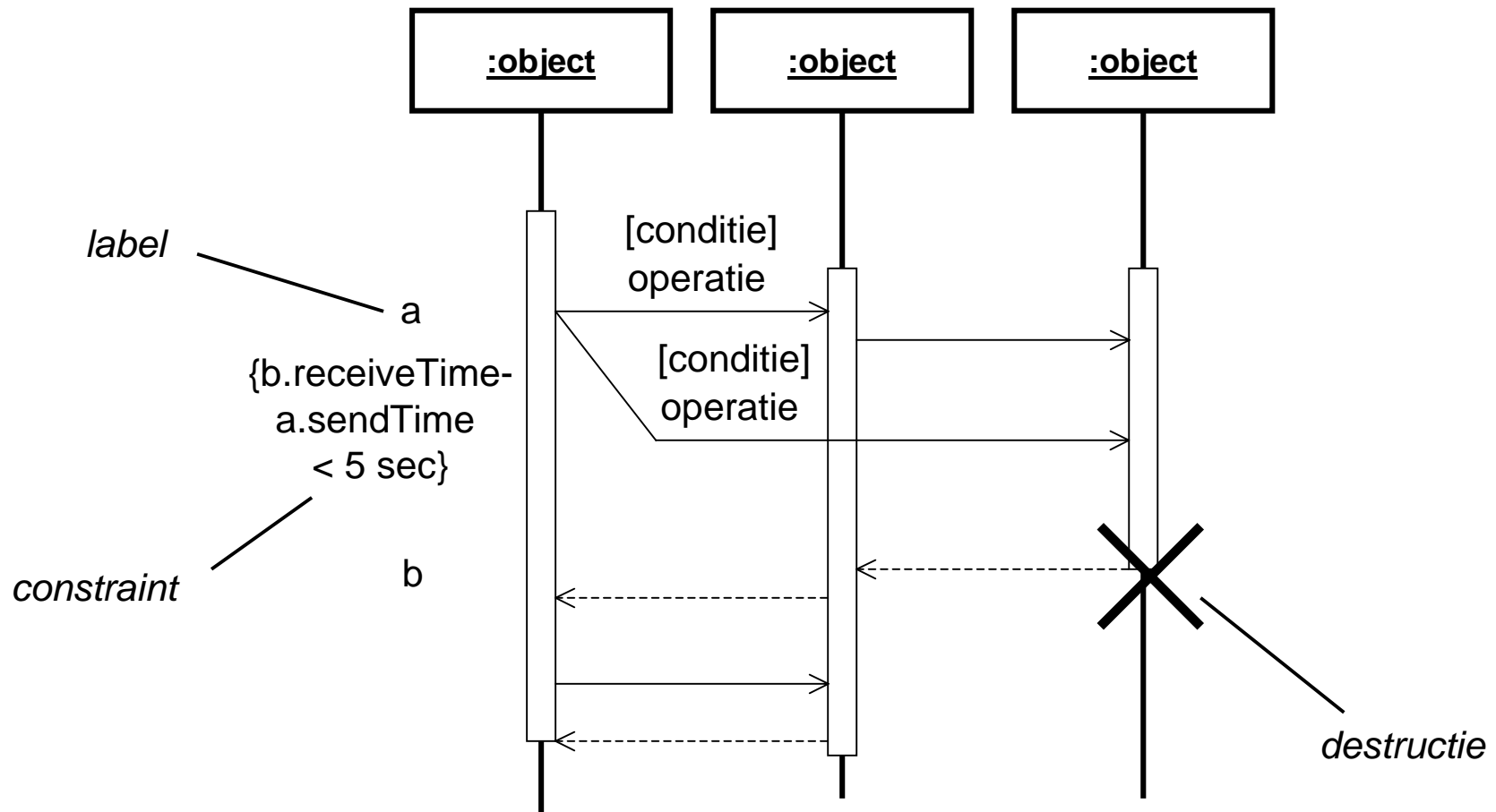
- ontvanger keert terug
- Optioneel by “synchroon”



Asynchrone berichten



Geavanceerde constructies



Collaboration diagrammen

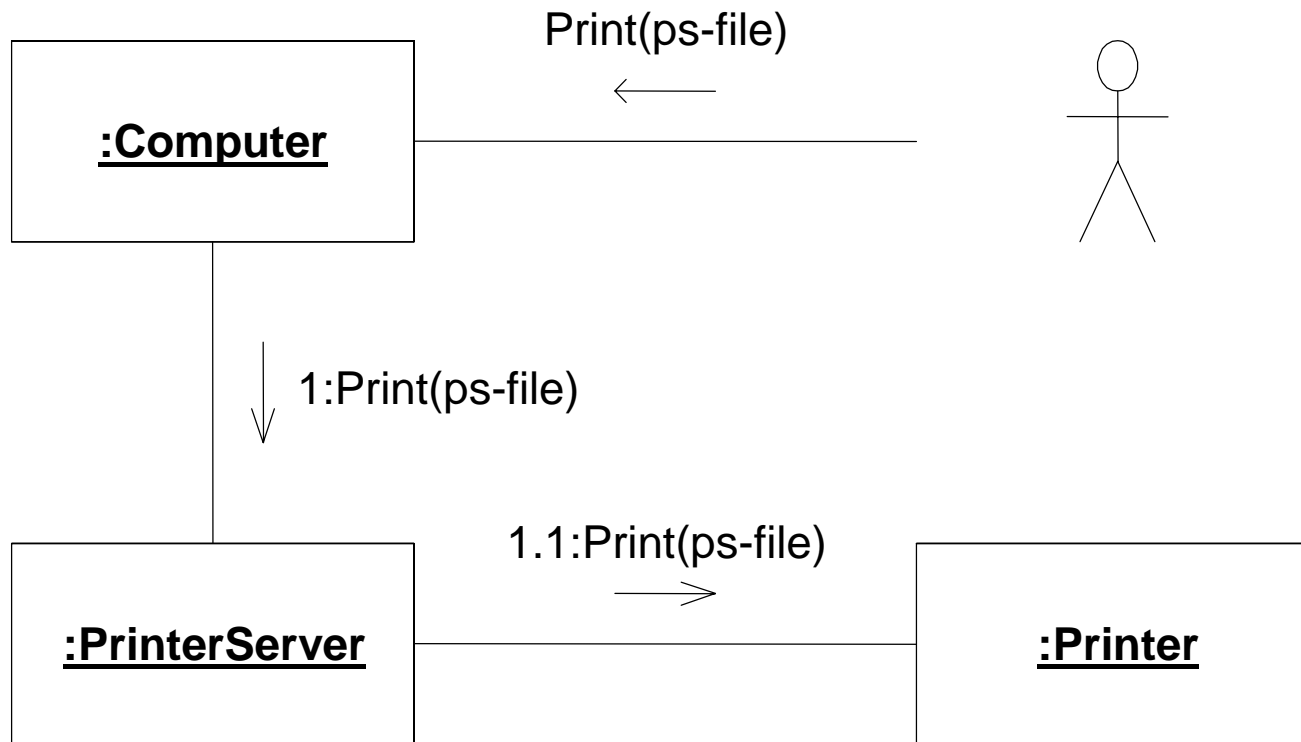
Communicatie tussen objecten

- berichten
- volgorde van berichten
- relaties tussen objecten

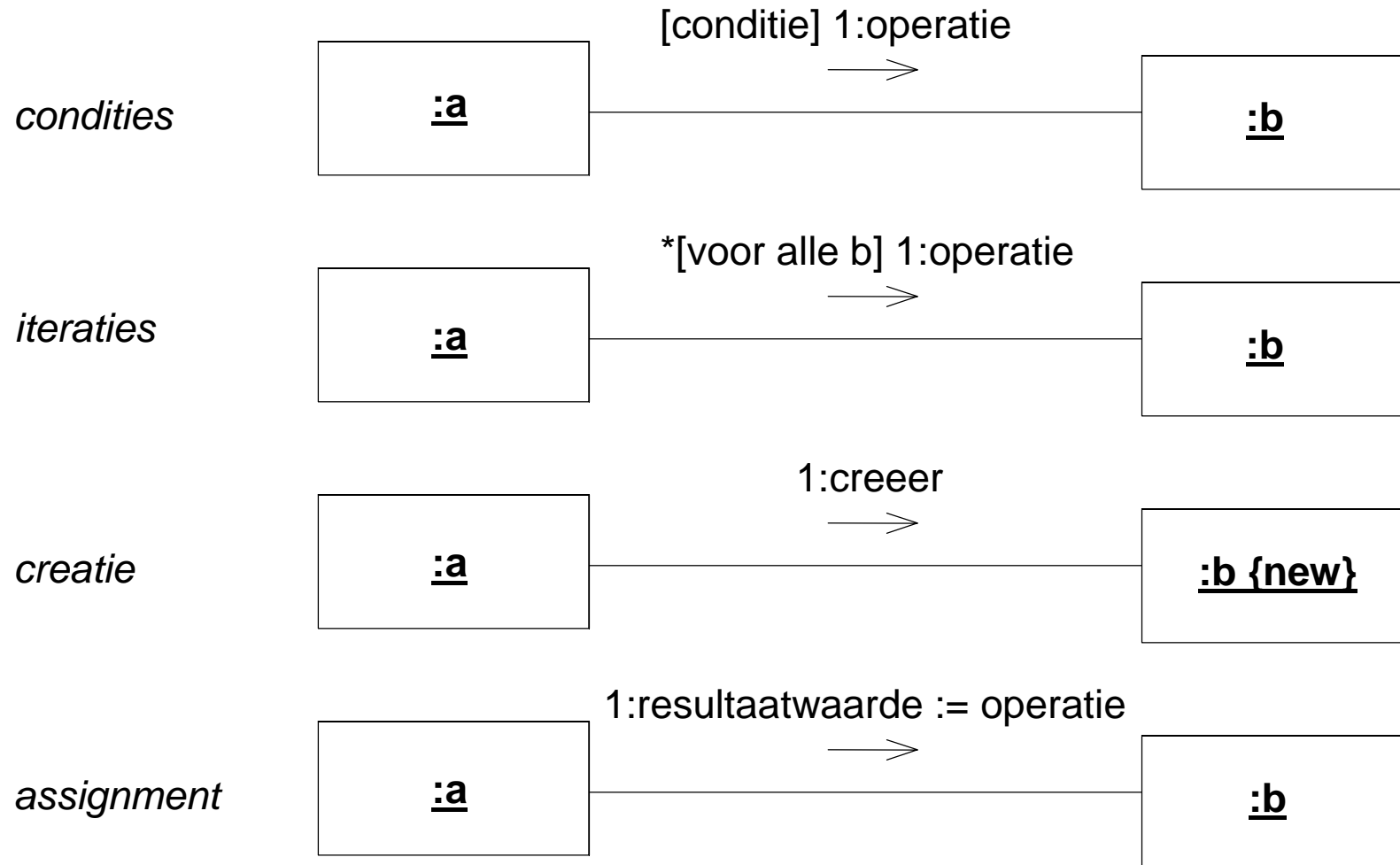
Gebruik voor

- inzicht in gedrag van systeem, nadruk op links tussen objecten
- illustreren use-case
- welke objecten zijn nodig voor het uitvoeren van een dienst?
- controle van 'toegangspaden'

Collaboration diagram



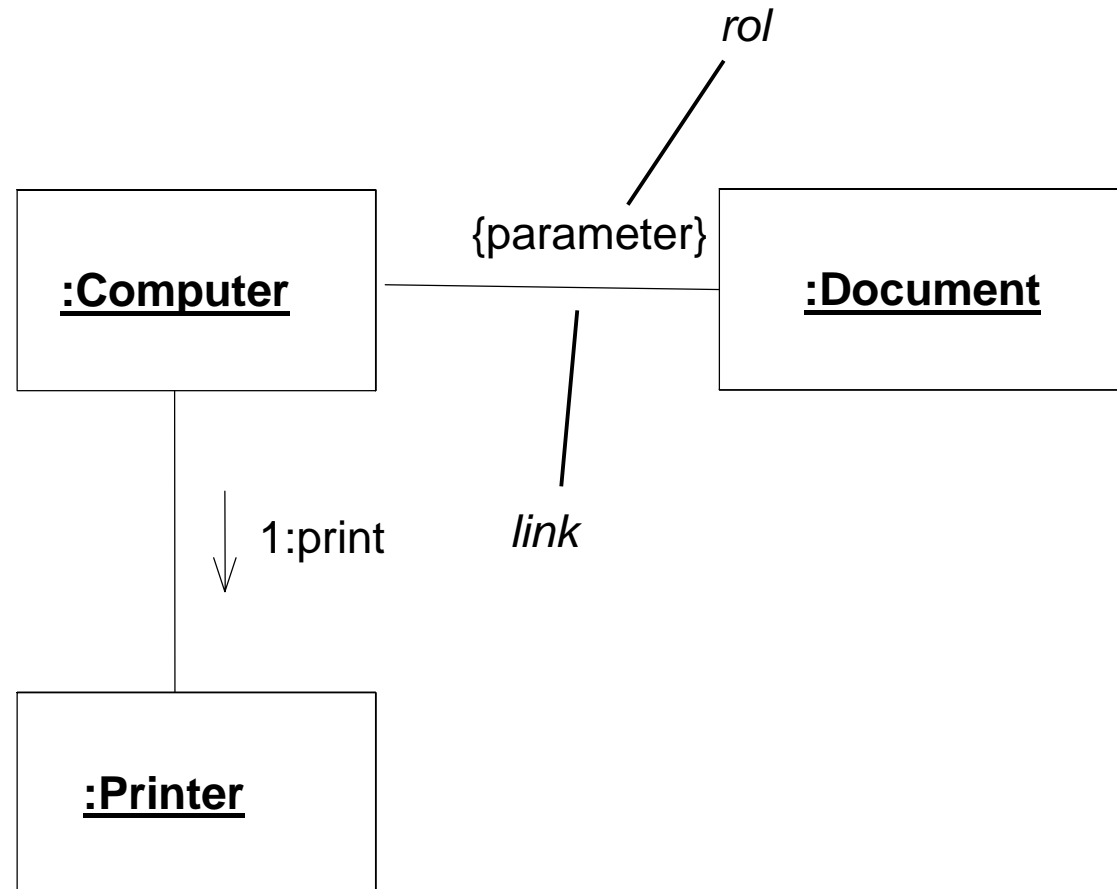
Geavanceerde constructies



Links en rollen

Link rollen

- parameter, object is een parameter in een operatie
- global, object is globaal beschikbaar
- local, object is locale variabele in operatie
- self, object kan berichten naar zichzelf sturen
- vote, antwoord wordt geselecteerd op basis van aantal stemmen
- broadcast, berichten worden niet in bepaalde volgorde afgehandeld



Statechart diagrammen

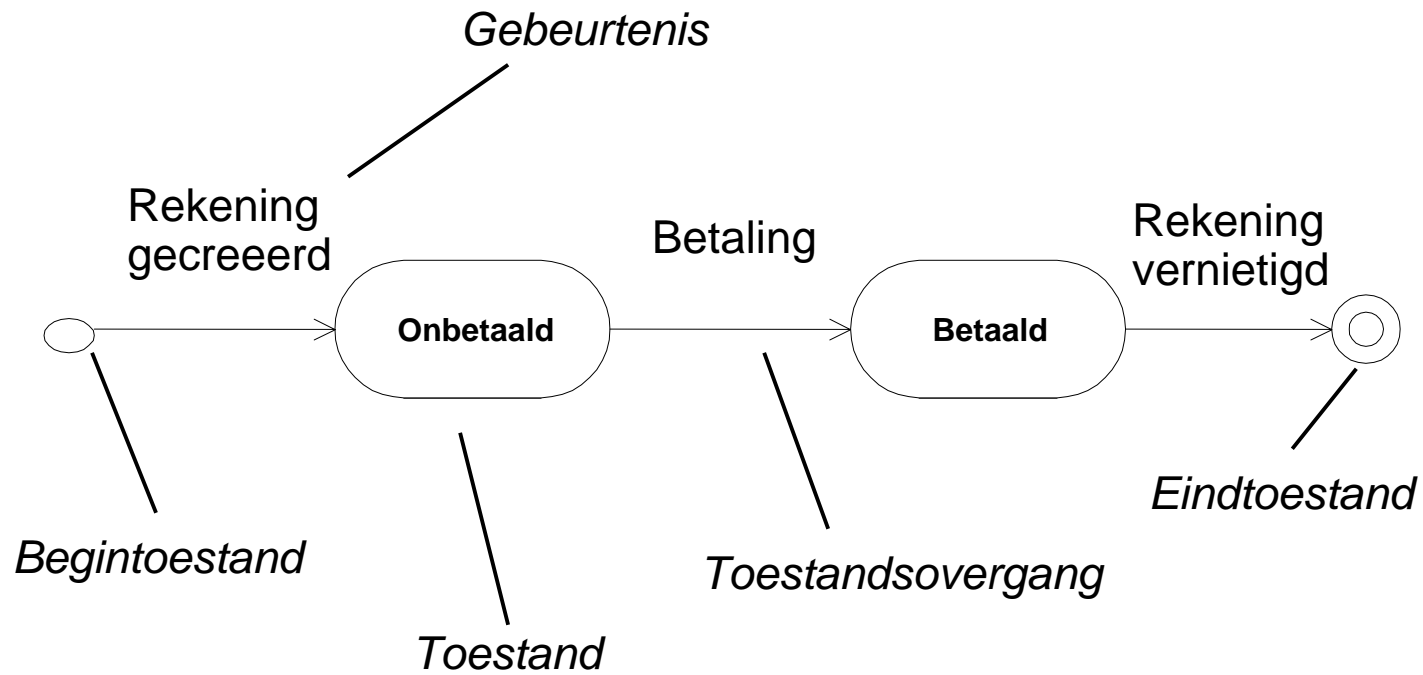
Beschrijven de toestanden van een object

- Toestanden, states
- Gebeurtenissen, events
- Toestandsovergangen, transitions

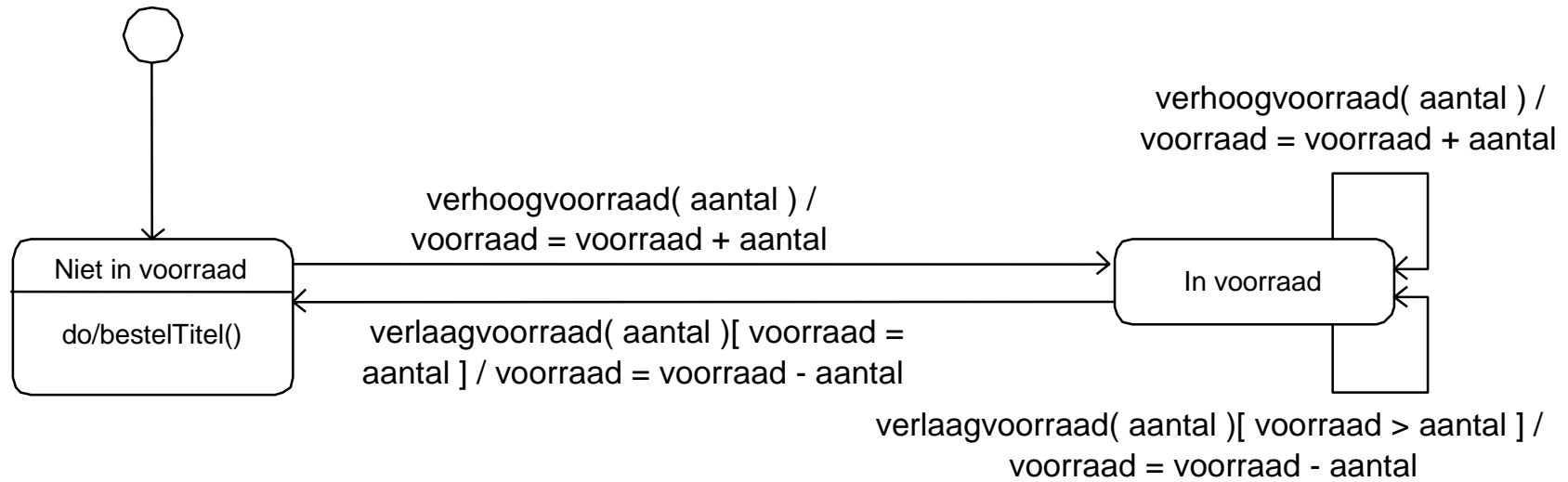
Gebruik voor

- inzicht in gedrag van een object in de tijd
- inzicht in externe stimuli

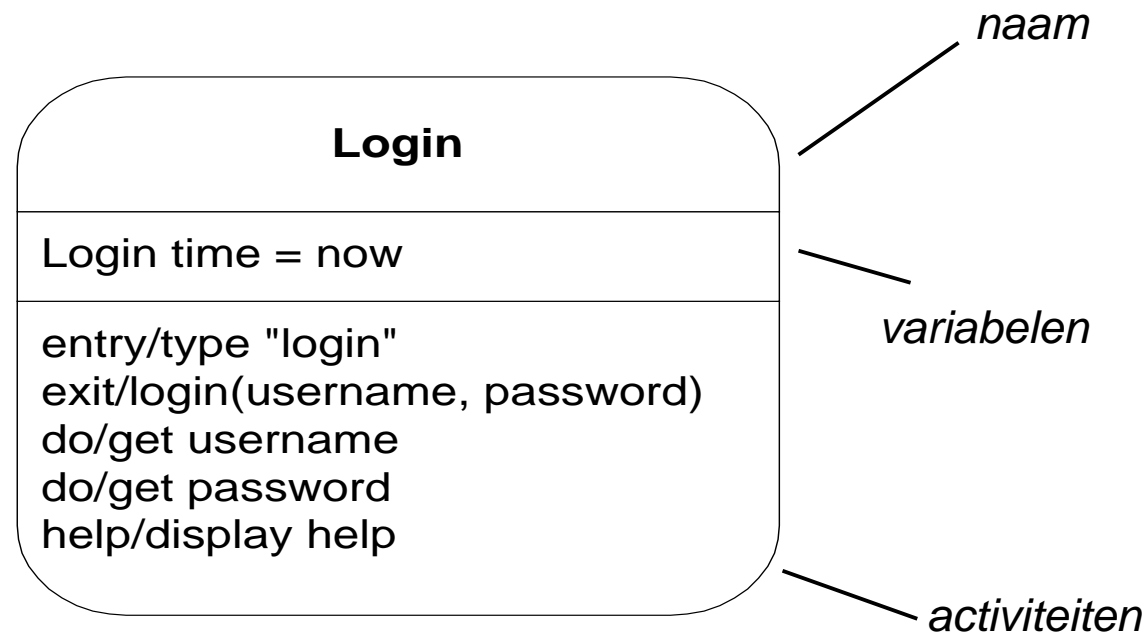
Statechart diagram



Statechart diagram



Toestanden



Activiteiten

- entry, worden uitgevoerd bij binnengaan van toestand
- exit, worden uitgevoerd bij uitgaan van toestand
- do, worden uitgevoerd in de toestand
- eigen gedefinieerde activiteiten ...

Transities

Transitie

- *event* [*condition*] / *action(s)*

Event

- Trigger voor een transitie
 - Message
 - Timeout
 - Signal
- Bijv. draw (f: figure, c: color)

Condition

- Guard op de transition
- Bijv. [color != blue]

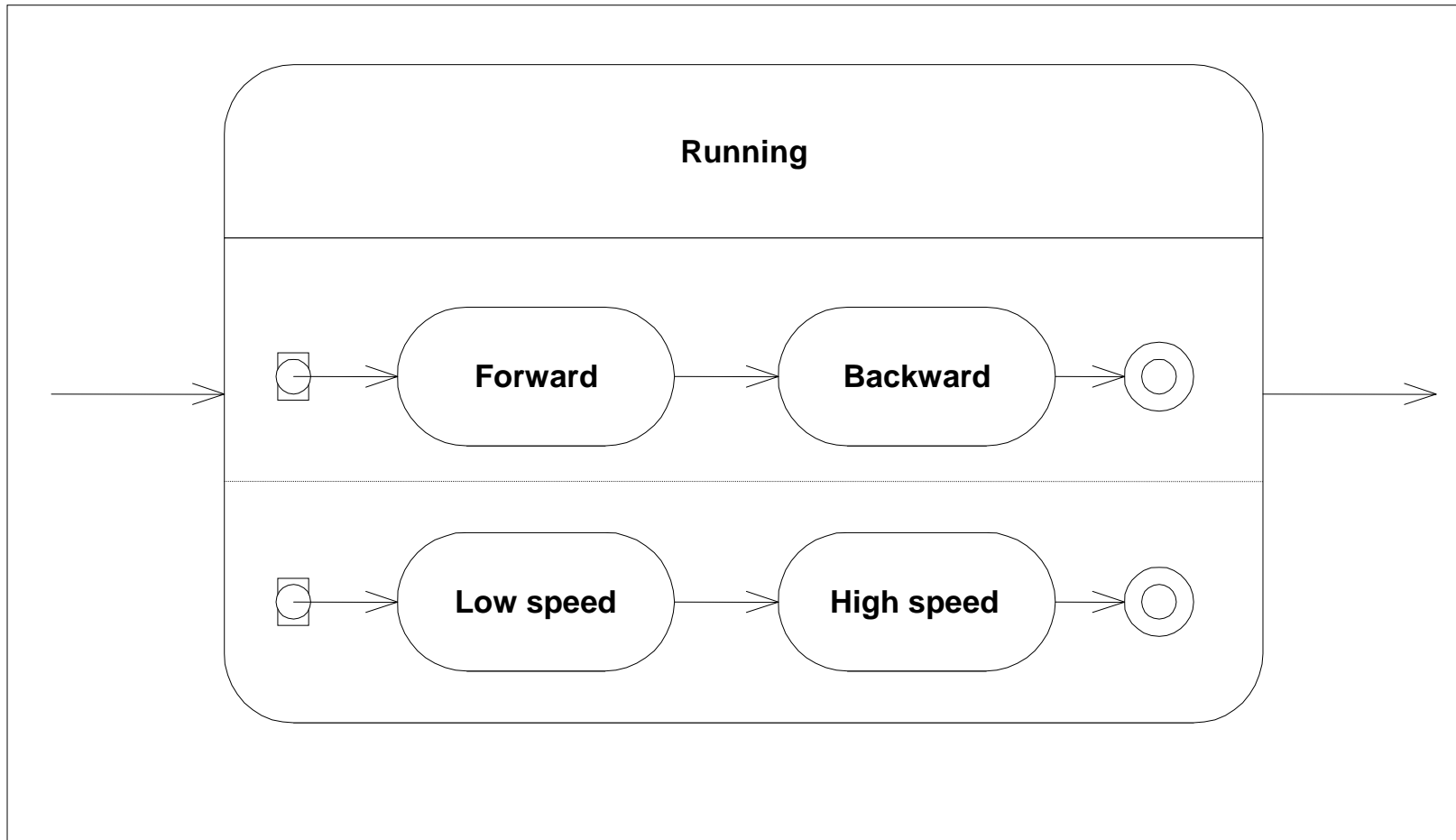
Action

- Uitgevoerd bij overgang
- Bijv. n:=n+1;
pen.setColor(color)

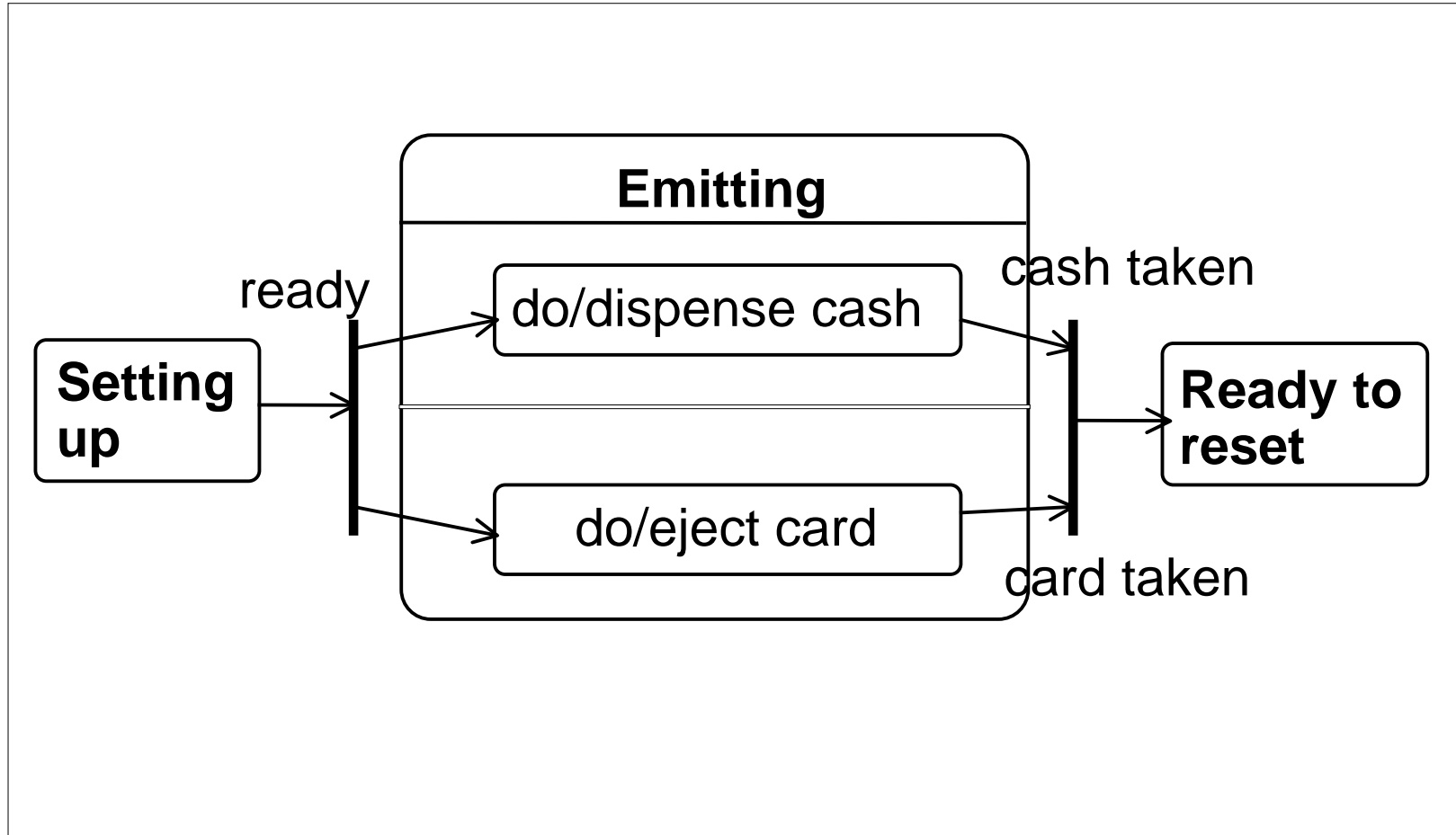
Lege transitie

- Als activiteiten in brontoestand klaar zijn

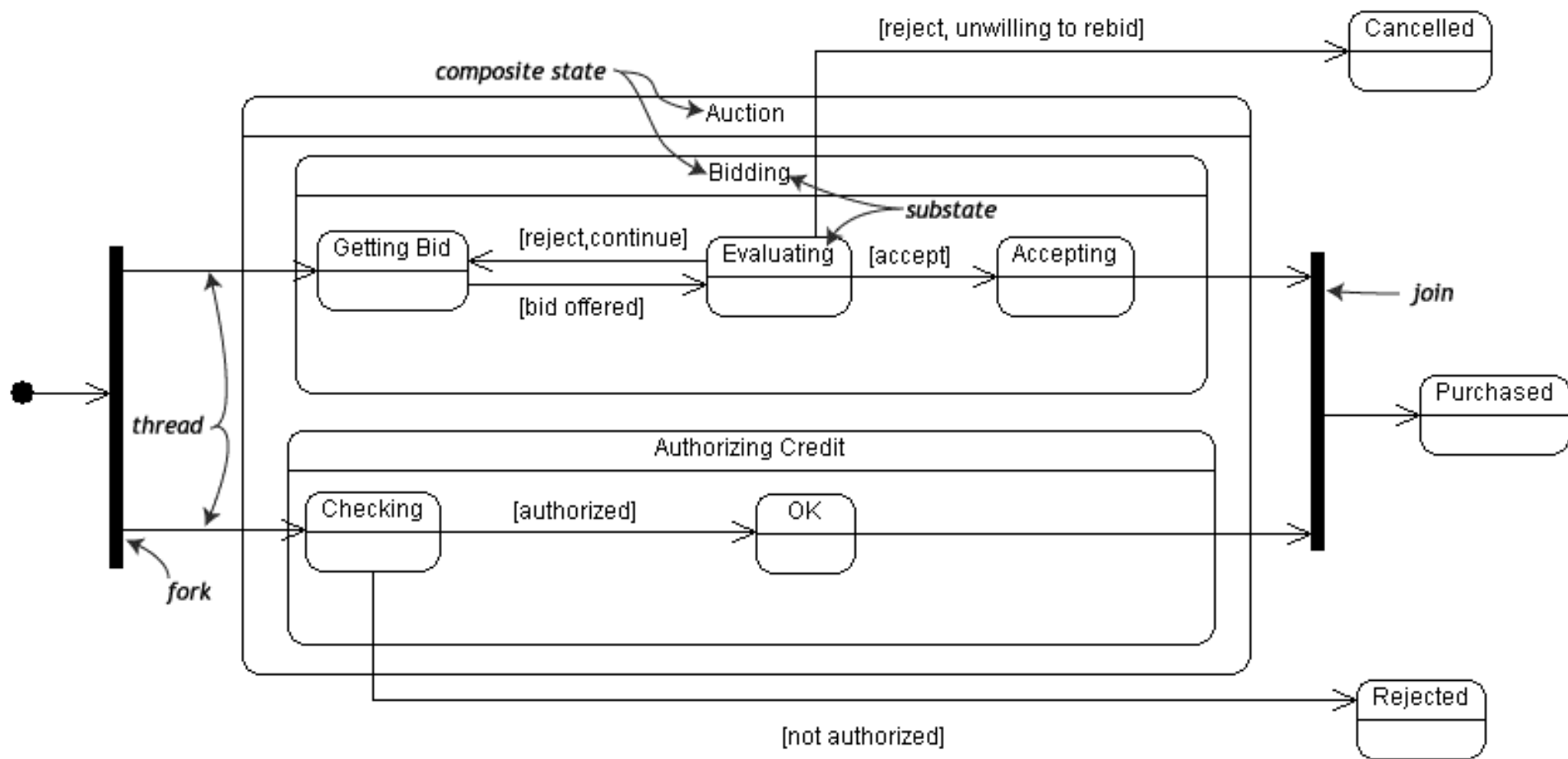
Subtoestanden



Synchronisatie



State Chart Diagrammen



Activity diagrammen

Beschrijven acties en hun resultaten

- operaties
- activiteiten in een use-case

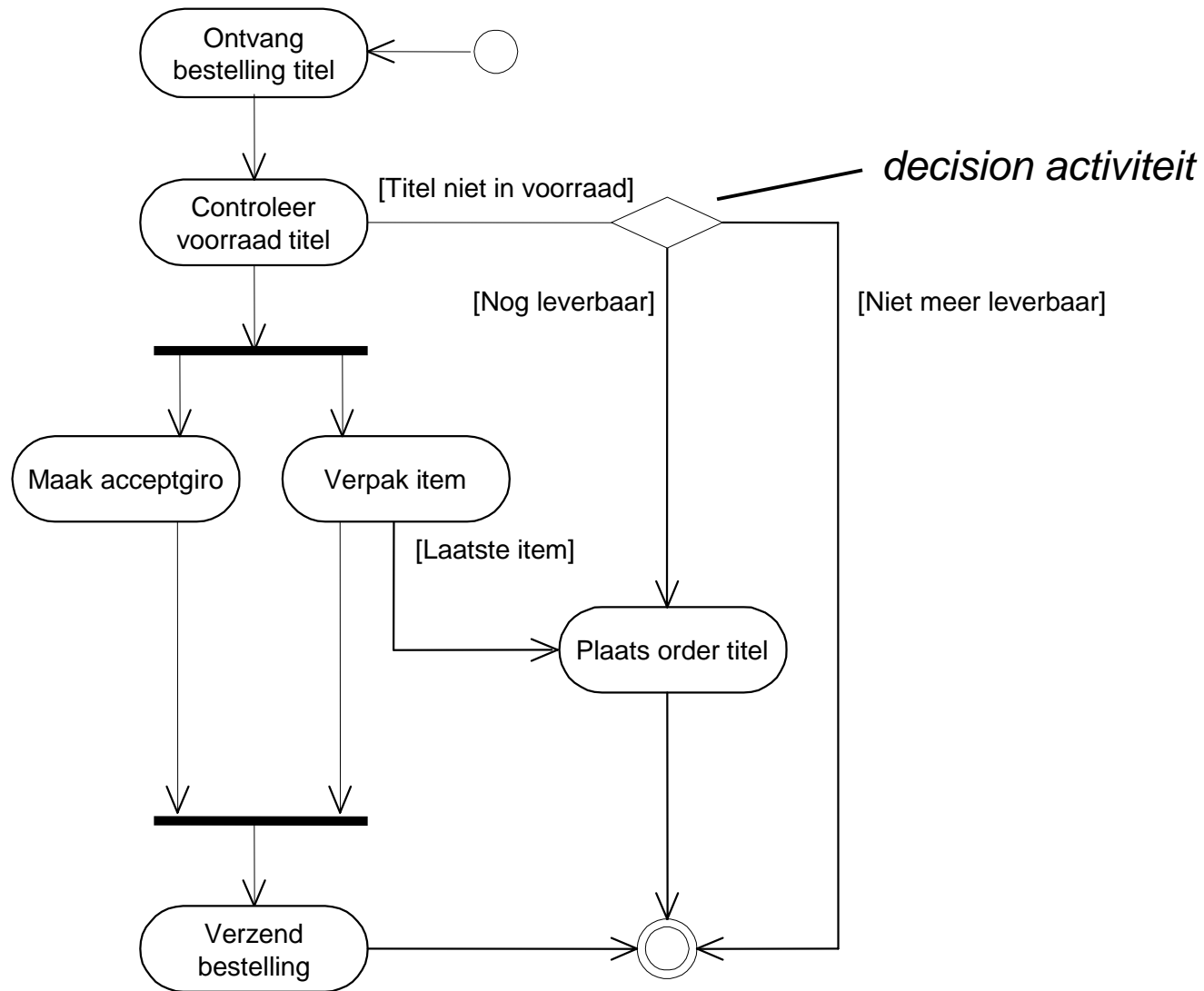
Gebruik voor

- inzicht in werk dat wordt uitgevoerd in een operatie
- inzicht in hoe een object werkt
- inzicht in acties en hun invloed op objecten
- illustreren use-case

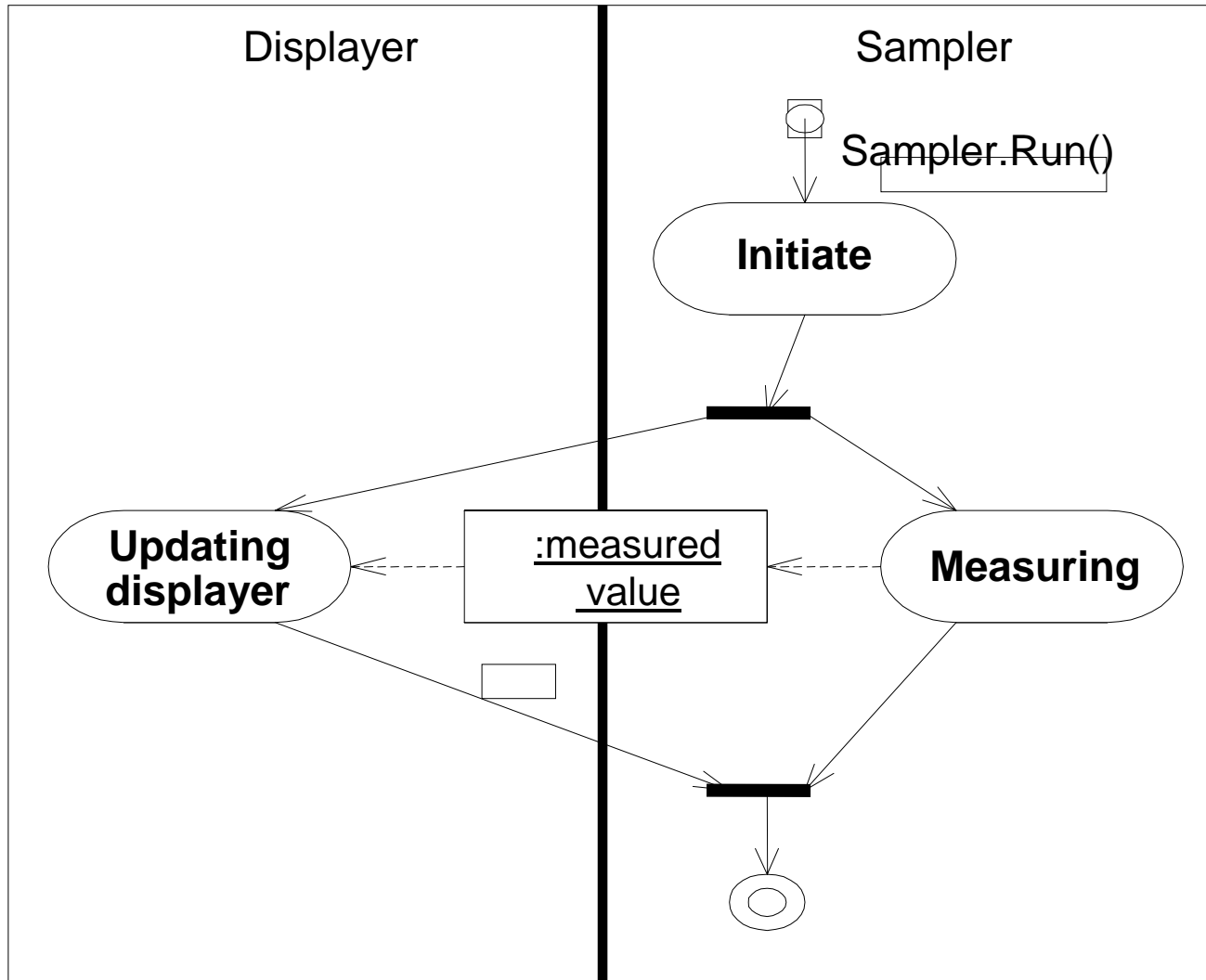
Let op

- Activity diagrammen zijn een speciaal soort statechart diagrammen
- De toestanden zijn acties die naar de volgende toestand gaan na het afronden van de actie

Voorbeeld Activity Diagram



Swimlanes



Component diagrammen

Component

- Een fysiek en vervangbaar onderdeel van een systeem dat conformeert aan, en de realisatie biedt van een aantal interfaces
- Omvat model elementen (<<reside>> relatie) die de component functionaliteit realiseren
- (geïmplementeerd als/met artifacts)

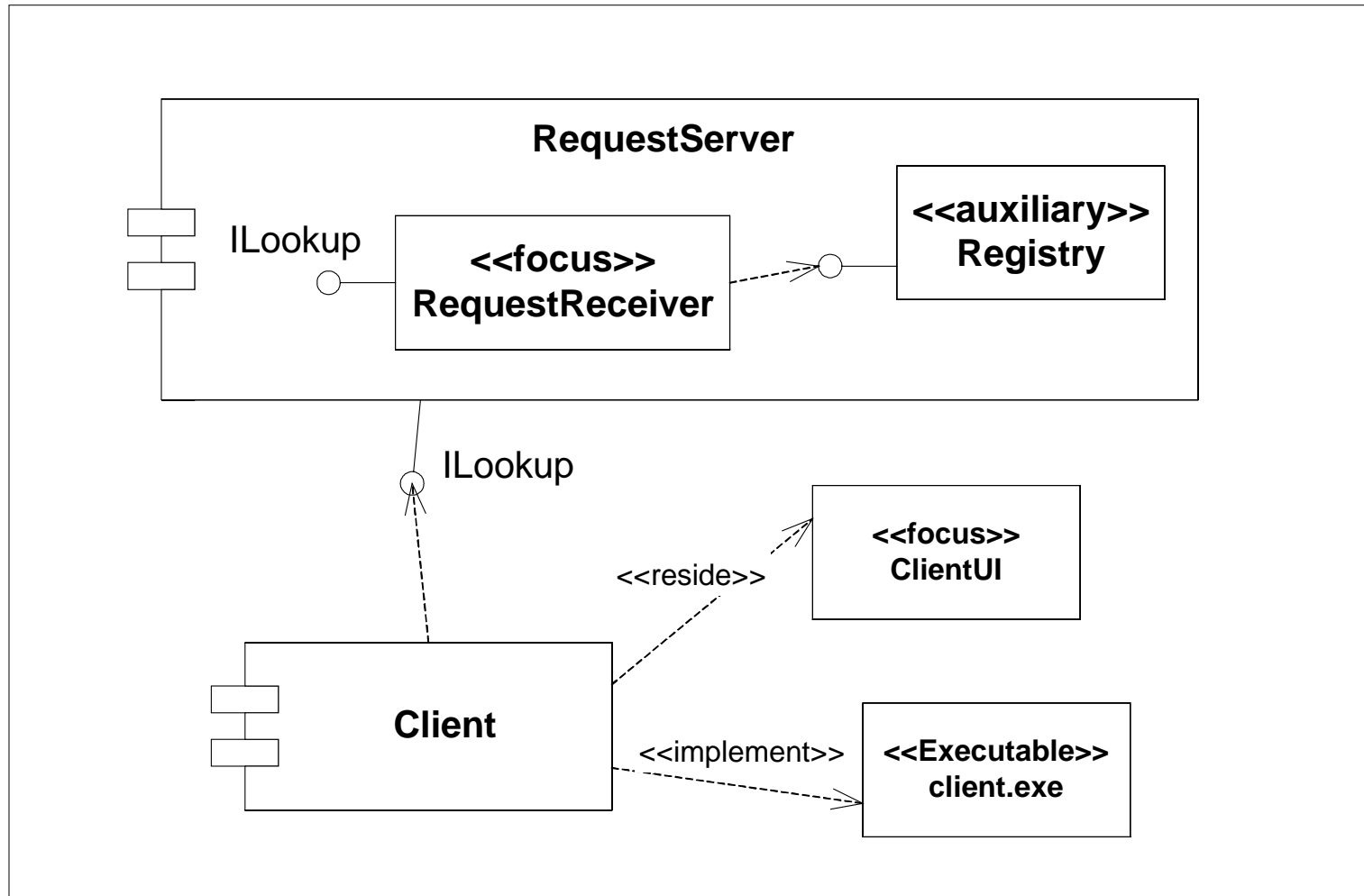
Interface

- Een interface beschrijft de diensten die een object aanbiedt in de vorm van operaties

Artifact

- “Fysiek” stuk data, bijv. <<file>>, <<executable>>, <<library>>

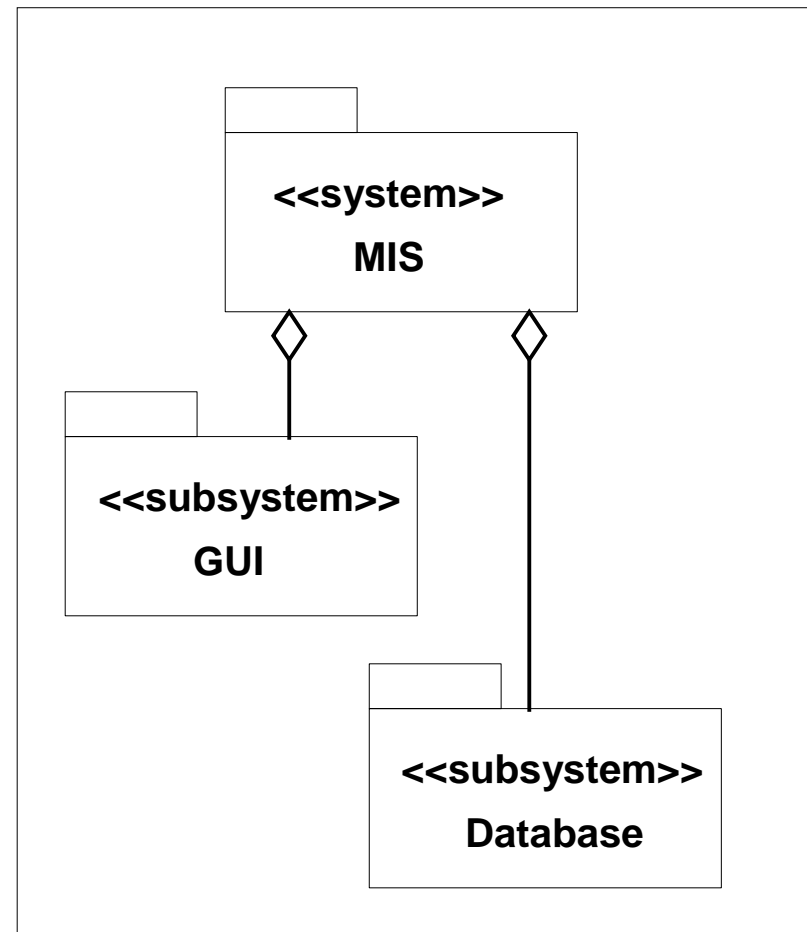
Component diagram



Systemen en subsystemen

Vergelijk met packages

- Een subsysteem definieert een apart onderdeel van het systeem (ontwikkeling, runtime)
- Een (sub)systeem heeft een specificatie-deel en een realisatie deel
- Een subsysteem biedt interfaces
- Systemen en subsystemen kunnen worden weergegeven als package met overeenkomstig stereotype
- Aggregatie kan worden weergegeven als nesting of middels relaties



Deployment Diagram

Node

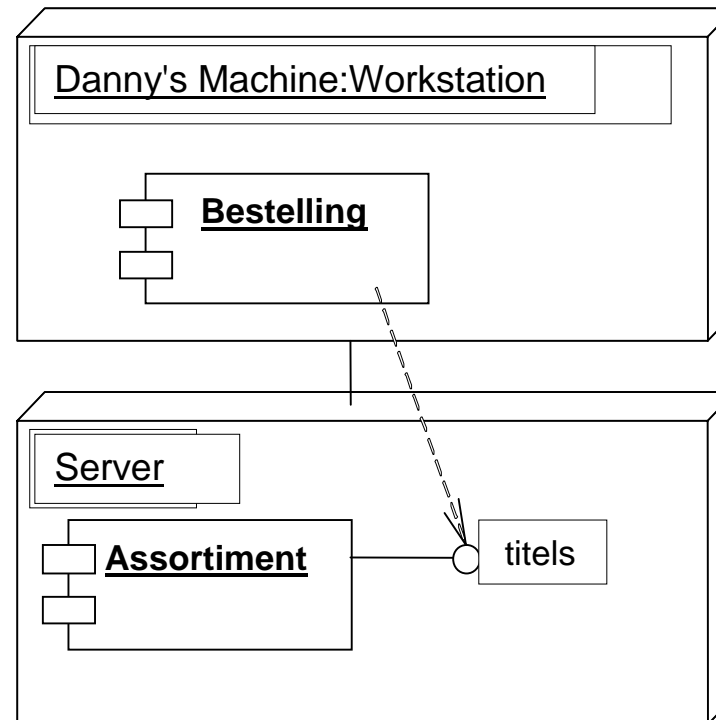
- Een fysiek element dat bestaat tijdens runtime en dat een computationele bron representeert (bijv. computer, printer, router)
- Weergegeven als een 3d rechthoek

Connection

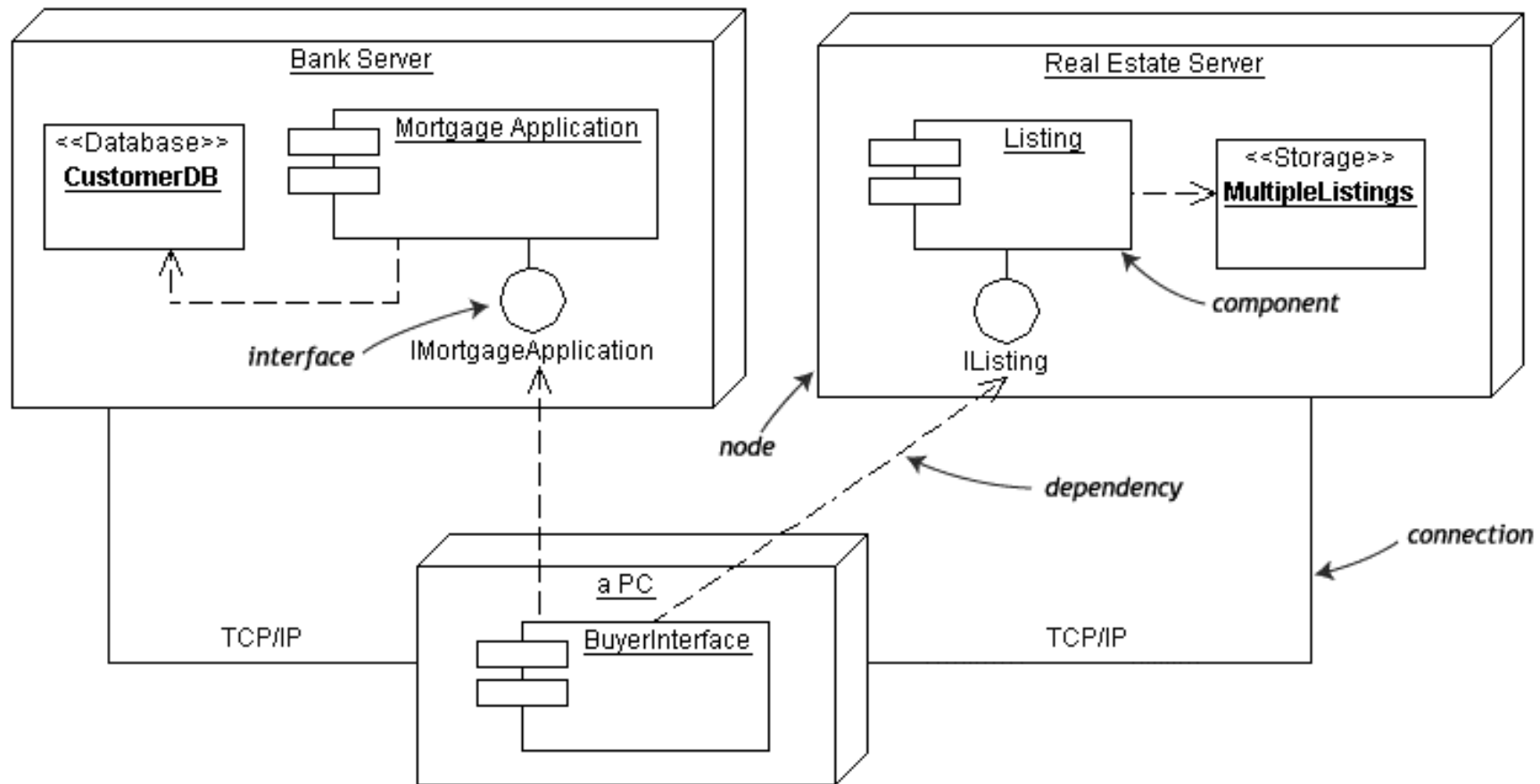
- Een fysieke verbinding tussen nodes (bijv. ethernet verbinding)

Componentallocatie

- Componenten die executeren op een node kunnen weergegeven worden met afhankelijkheidsrelaties <<reside>>

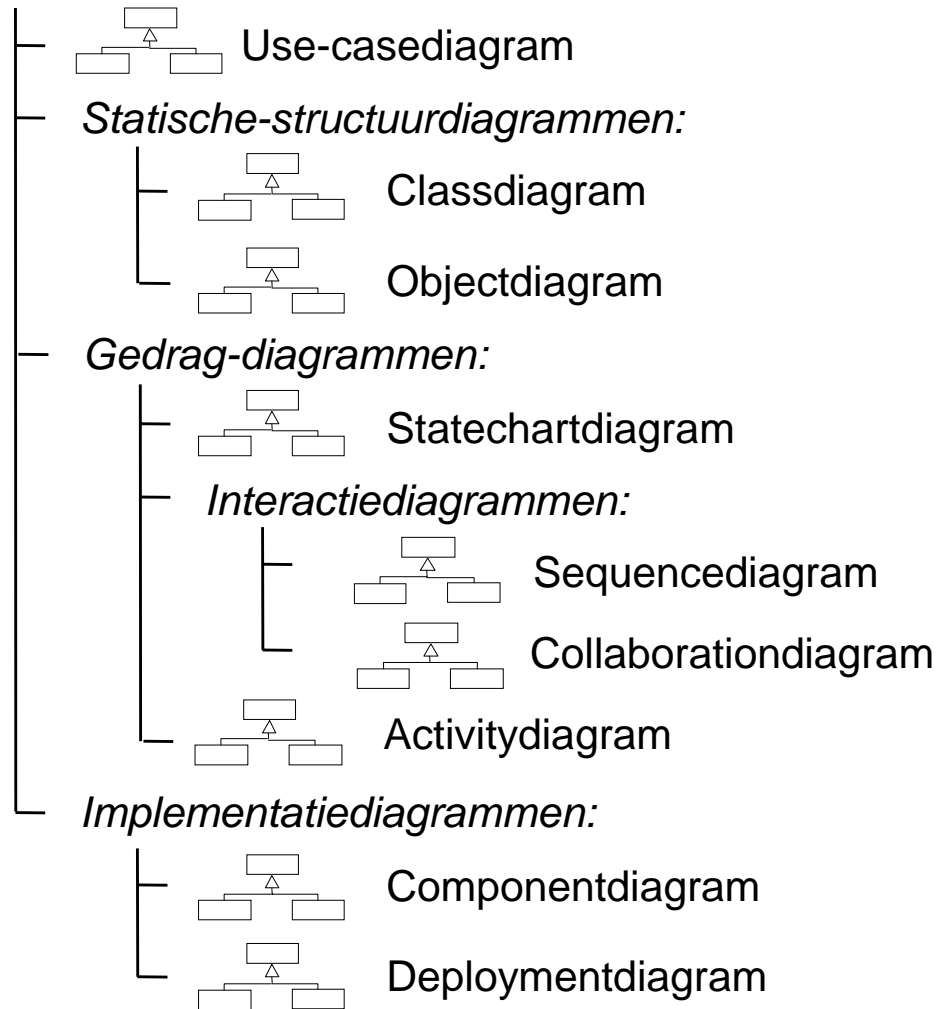


Deployment Diagram

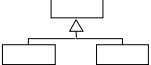


UML Diagrammen

UML-diagrammen:



cursief : Diagramsoort

 : Concreet UML-diagram

Conclusies

UML

- Rijke diagramtechnieken
- Uitbreidbaar naar eigen wensen
- Overzichtelijkheid?
- Volledige ondersteuning?

Gebruik

- Rational Rose, Together, Visio, ...
- Overdaad schaadt

Referenties

- Martin Fowler, UML Distilled, 2nd edition, Addison Wesley
- <http://www.omg.org>
- http://www.togethersoft.com/services/practical_guides/umlonlinecourse/index.html