

# SWOP – Iteratie 3

## Groep 21

Kenny Op De Beeck  
Jonathan Langens  
Andrey Kirillin  
Wonne Joosen

# Rollenverdeling

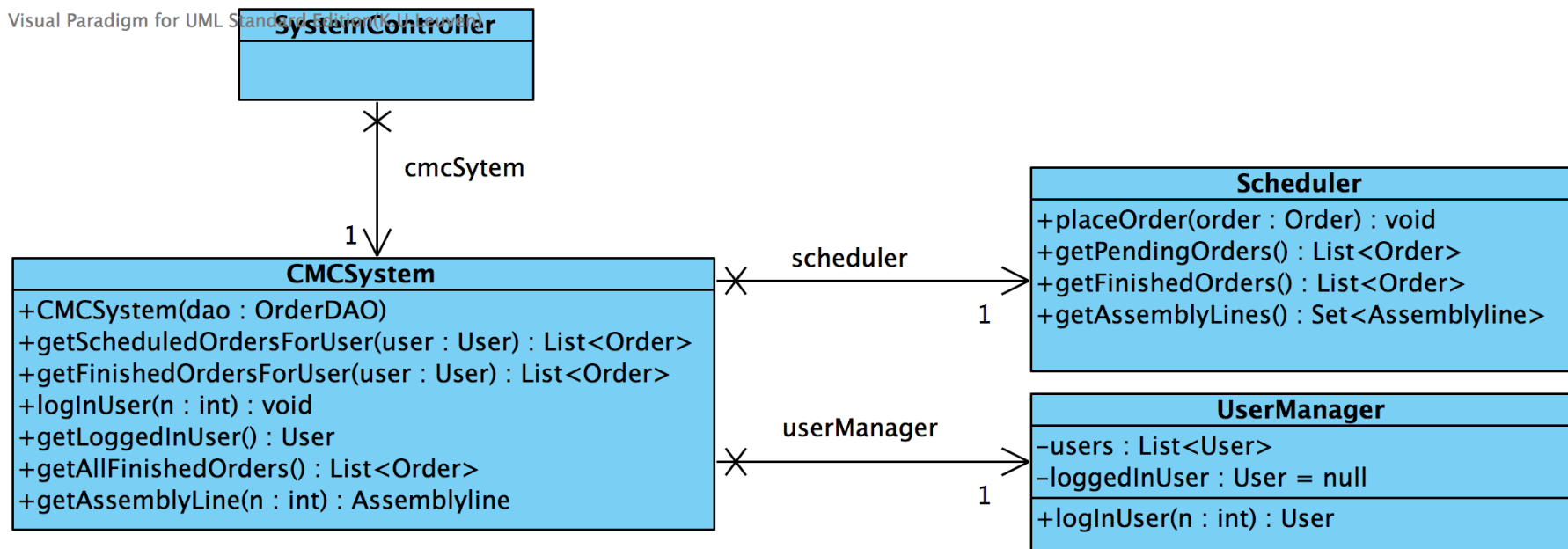
- Lead Designer = Kenny
- Lead Tester = Wonne
- Lead Domain Modeller = Jonathan

# Index

1. CMCSYSTEM
2. User
3. Tasks
4. Parts
5. Restrictions
6. Orderbuilder
7. Scheduler
8. Assembly line
9. Controllers
10. View

## 1. CMCSytem - implementatie

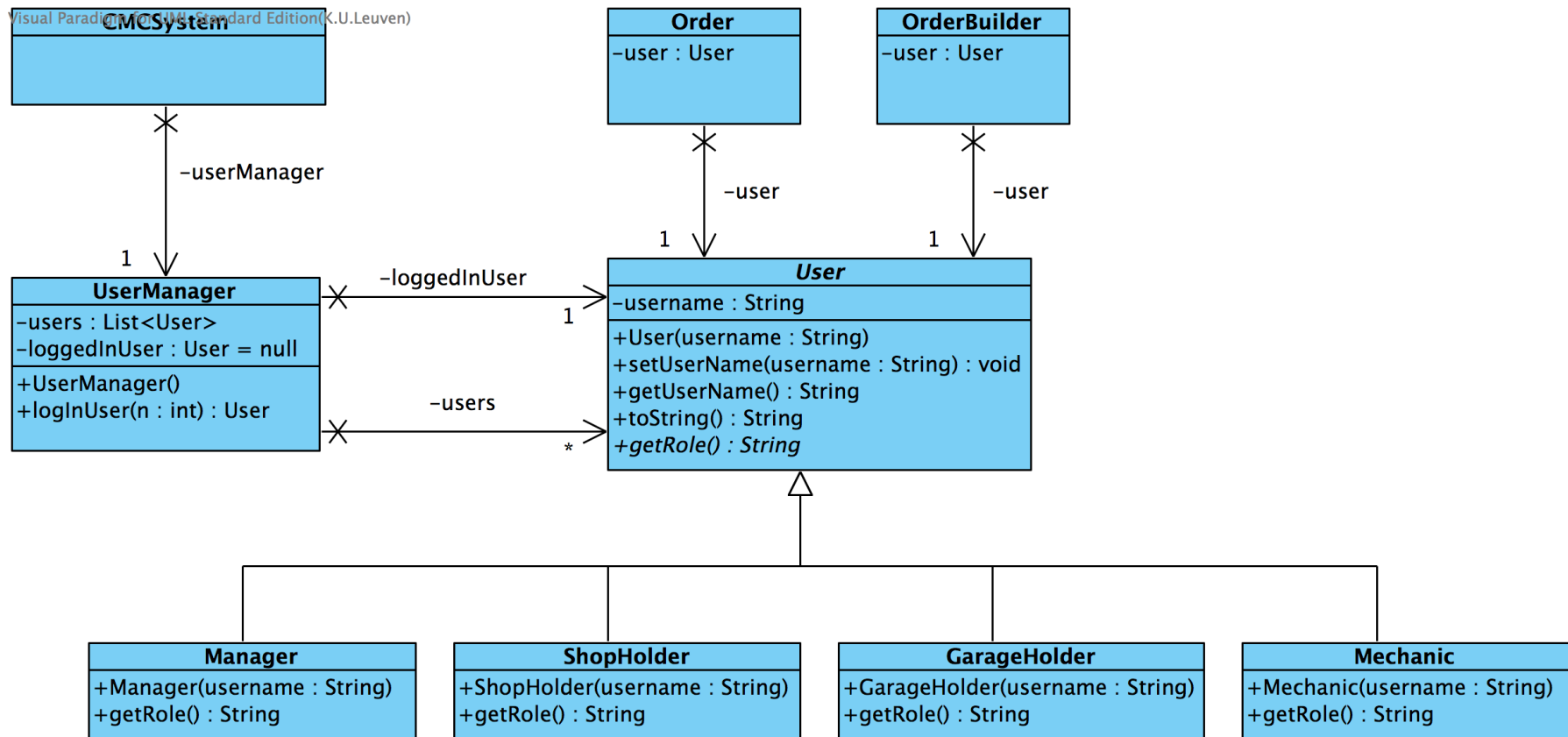
Visual Paradigm for UML Standard Edition (UML 2.0)



## 1. CMCSysystem – Implementatie/Design

- **Verantwoordelijkheden**
  - Een algemene toegang tot het systeem bieden
  - Het systeem initialiseren
  - Het systeem 'managen'
- **GRASP**
  - Pure fabrication

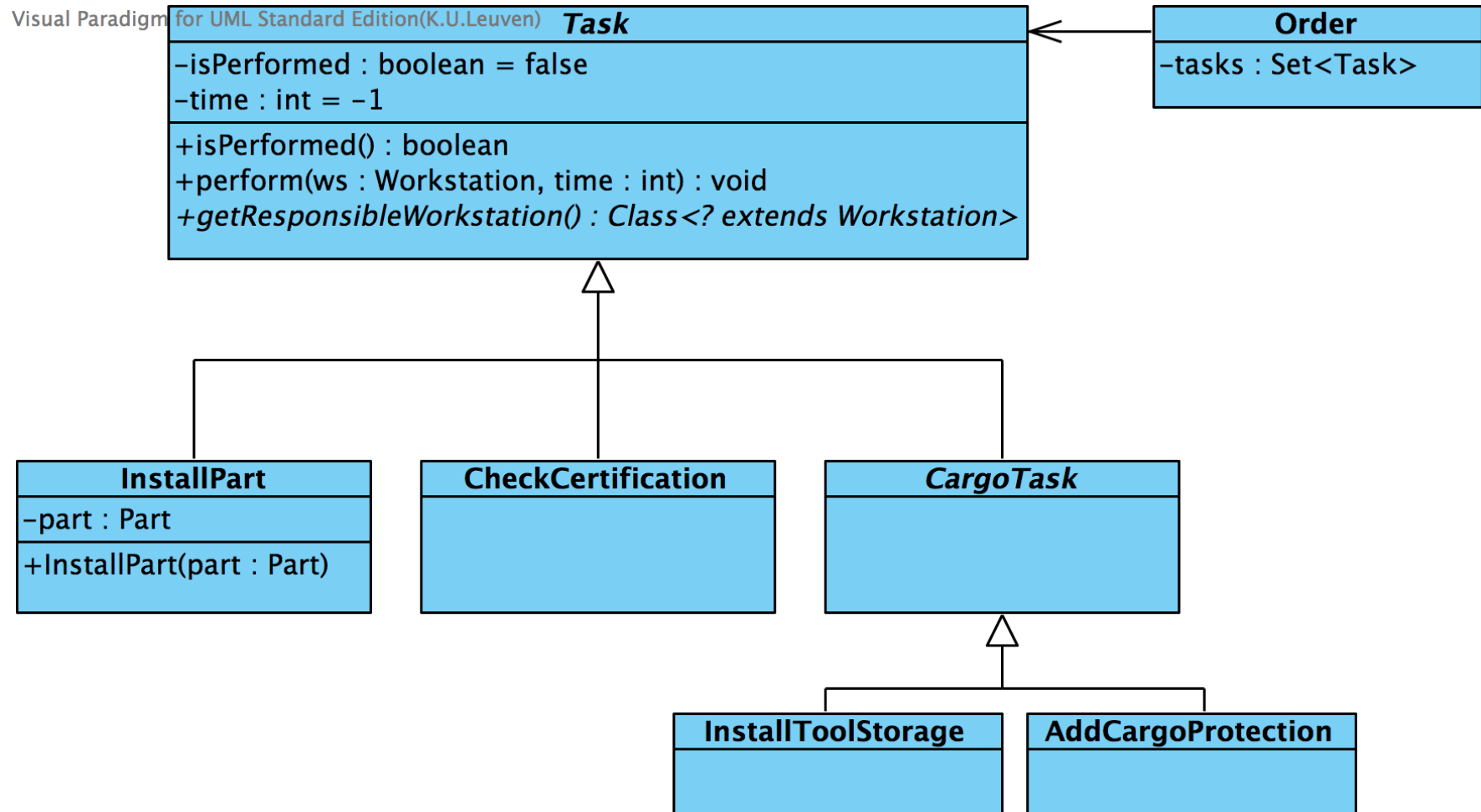
## 2. User/UserManager - Implementatie



## 2. User/UserManager - Design

- **Verantwoordelijkheden**
  - Een gebruiker laten inloggen
  - De gebruikers beheren op een gemakkelijke manier
  - Gemakkelijk vervangbaar zijn
  - Weten wie ingelogd heeft
- **Grasp**
  - Polymorphism

### 3. Tasks – Implementatie



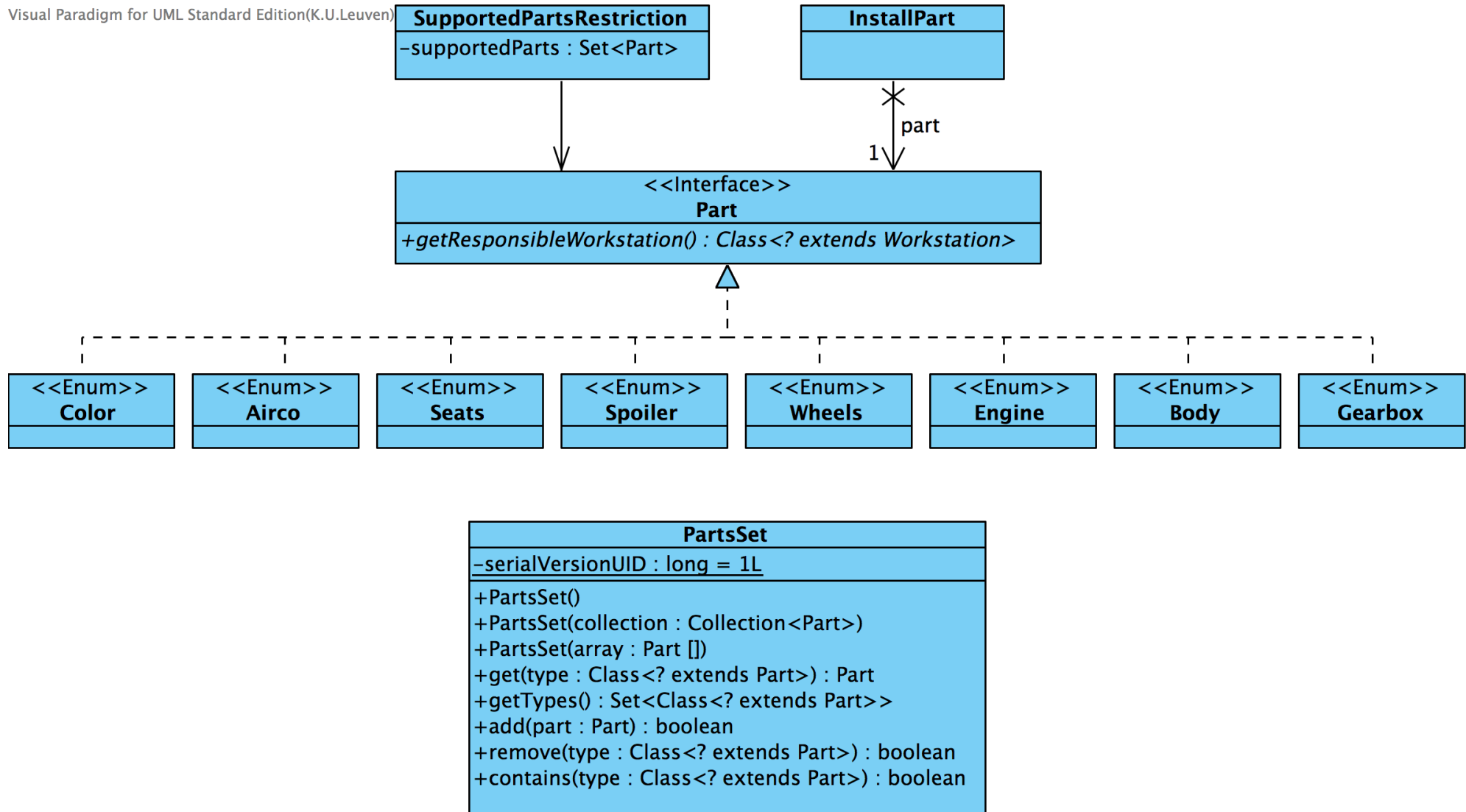


### 3. Tasks - Design

- **Verantwoordelijkheden**
  - Een Task heeft zelf een methode om uitgevoerd te worden en houdt daarna ook bij hoe lang het duurde om uitgevoerd te worden.
  - Een Task heeft een methode om op te vragen in welke workstation die kan worden uitgevoerd. In geval van InstallPart vraagt die dat dus aan de Part die hij met z'n constructor meekreeg.
- **GRASP**
  - Polymorphism
  - Protected variations
  - Indirection (voor InstallPart)

## 4. Parts - Implementatie

Visual Paradigm for UML Standard Edition(K.U.Leuven)

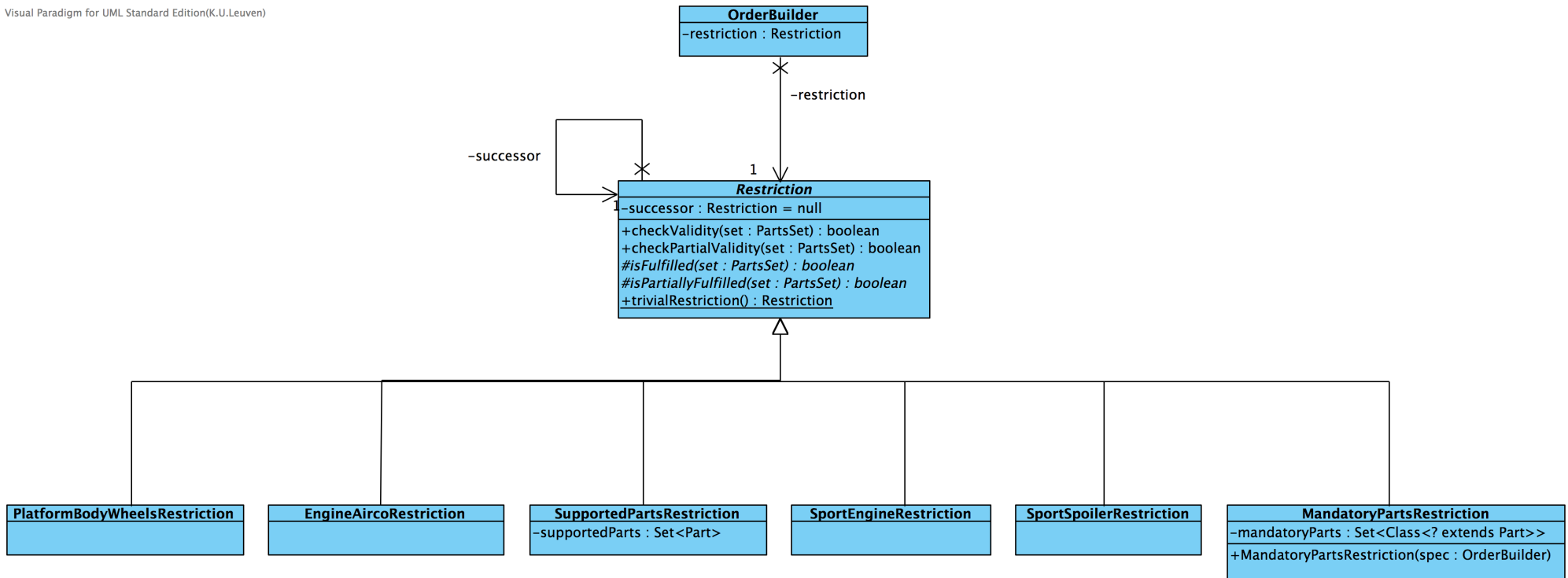


## 4. Parts - Design

- **Verantwoordelijkheden**
  - Interface Part heeft de methode `getResponsibleWorkstation`.
  - Voor elke soort is er een enum die de interface Part implementeert
  - `PartsSet` erft over van `HashSet` en geeft de extra restrictie dat er slechts één element van hetzelfde type in de set kan zijn.
- **GRASP**
  - Polymorphism

## 5. Restrictions - Implementatie

Visual Paradigm for UML Standard Edition(K.U.Leuven)

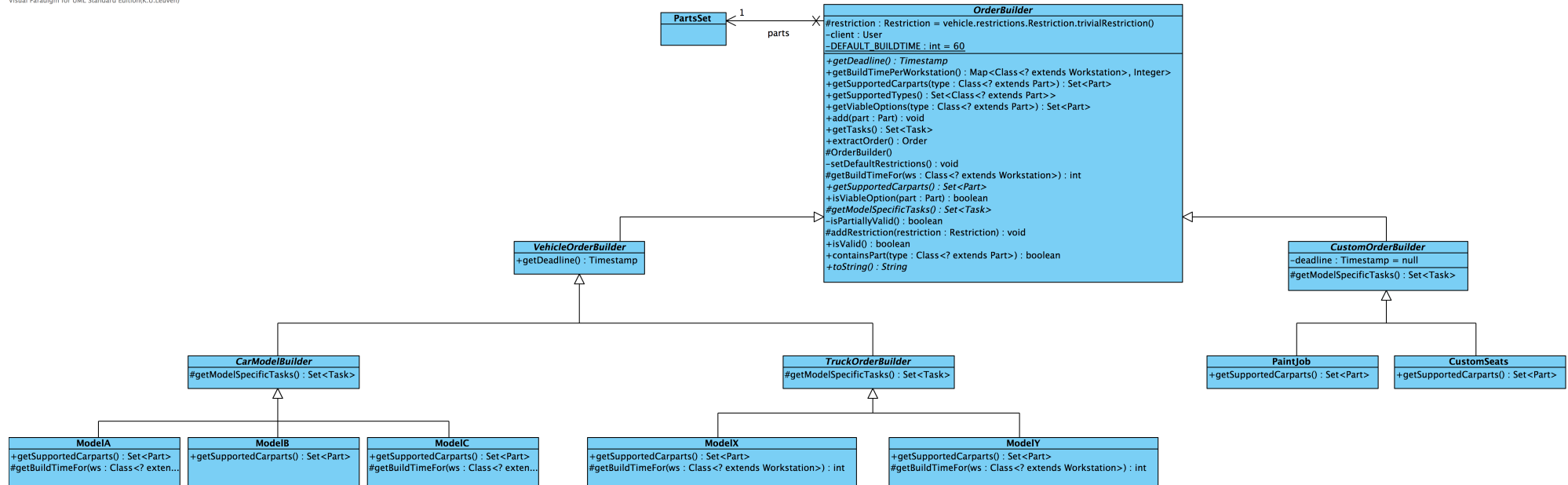


## 5. Restrictions - Design

- Verantwoordelijkheden
  - Restriction heeft methodes `checkValidity` en `checkPartialValidity`
- Design Pattern
  - Chain Of Responsibility
- GRASP
  - Protected Variations
  - Polymorphism

## 6. OrderBuilder - Implementatie

Visual Paradigm for UML Standard Edition(K.U.Leuven)



## 6. OrderBuilder - Design

- **Verantwoordelijkheden**

- Orderbuilder biedt functionaliteit om een order te bouwen door één voor één parts in te geven en deadline en user enzo.
- OrderBuilder houdt de bijhorende restrictions bij en kan alleen een Order bouwen wanneer de restrictions voldaan zijn.
- Een OrderBuilder heeft een PartSet. Waarin elke soort part slechts 1 keer kanvoorkomen.

- **GRASP**

- Pure fabrication
- Information Expert
- Creator

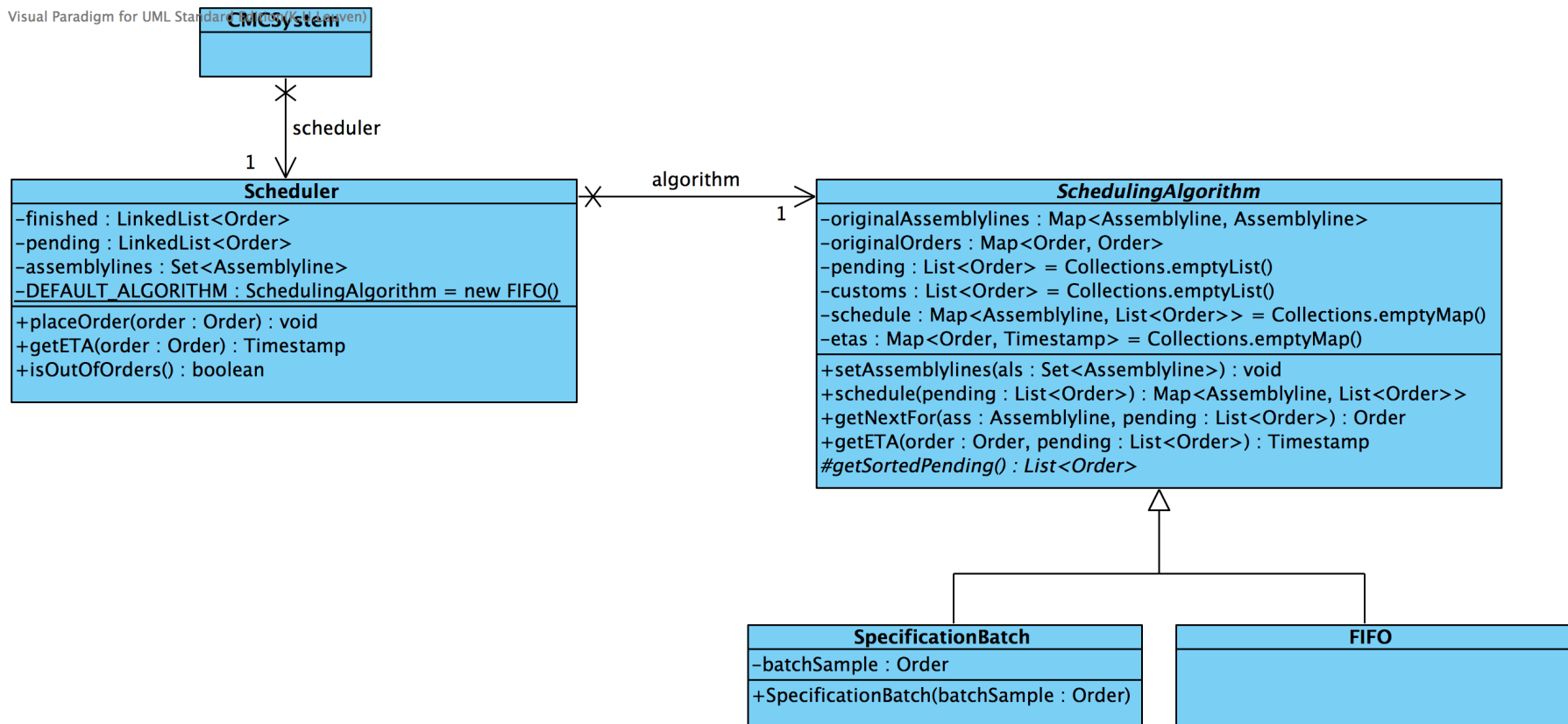
## 6. OrderBuilder - Design

- Design Pattern
  - Builder



## 7. Scheduler - implementatie

Visual Paradigm for UML Standard (© 2015 Visiogen)



## 7. Scheduler - Design

- **Verantwoordelijkheden**

- Scheduler heeft een set assemblylines en observeert deze.
- Wanneer een Assemblyline meldt dat er een verandering gebeurde, kijkt schedule wat er moet gebeuren en avancet de assemblyline indien nodig (of begint nieuwe dag indien nodig).

## 7. Scheduler - Design

- **Verantwoordelijkheden (2)**

- Schedulingalgorithm biedt methodes om een map te genereren die voor bepaalde assemblylines gegeven orders in een lijst zet.
- SchedulingAlgorithm scheduled d.m.v. simulatie. De echte assemblylines, orders, e.d. mogen natuurlijk niet gewijzigd worden. Daarom zijn Assemblyline, Workstation, Order en Task allemaal cloneable en wordt de simulatie op clones uitgevoerd.

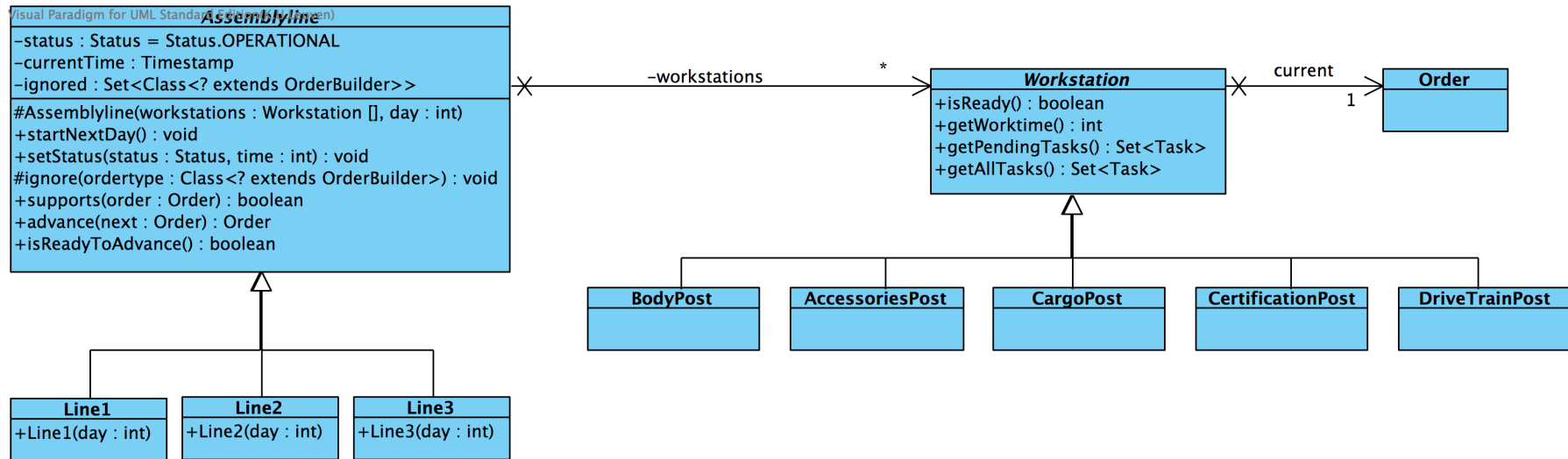
- **GRASP**

- Information expert
- High Cohesion in Scheduler
- Protected Variations (variabele planning logica in verschillende subklassen)

## 7. Scheduler - Design

- Design Pattern
  - Template Method(toegepast op SchedulingAlgorithm)
  - Observer (Scheduler 'observeert' Assemblyline)

## 8. Assembly Line - Implementatie



## 8. Assembly Line - Design

- **Verantwoordelijkheden**

- Workstation kan een Order bijhouden en heeft methodes om alle pending tasks op te vragen die kunnen uitgevoerd worden in die workstation.
- Assemblyline houdt zelf bij wat zijn status is (enum) en wat de currenttime is (voor die specifieke assemblyline)
- Heeft een ignore-list met types orders die geweigerd worden op deze assemblyline (wordt vanuit subklasse bepaald).

- **Design Pattern**

- Observer

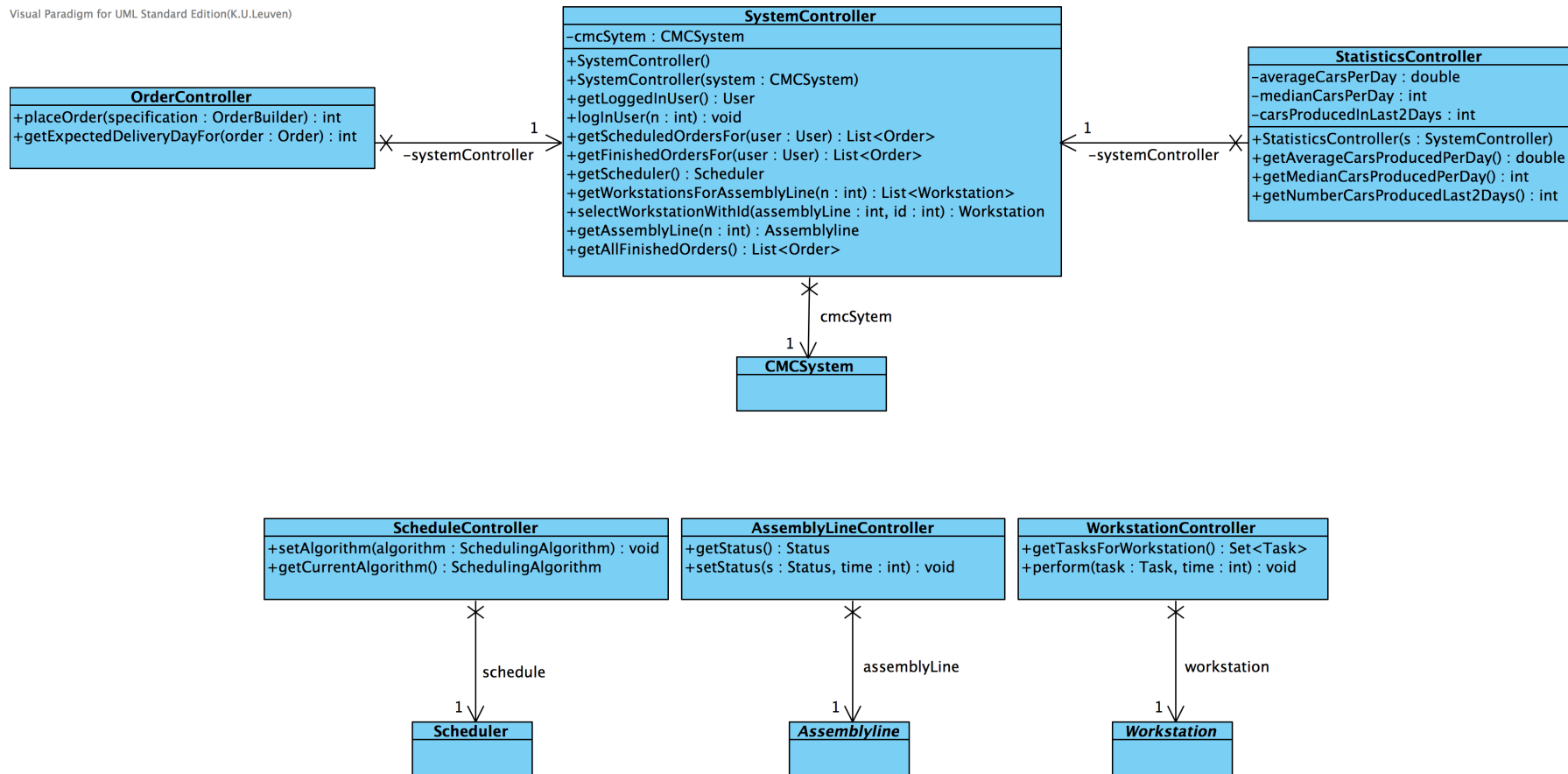
## 8. Assembly Line - Design

- **GRASP**

- Polymorphism (voor Assemblyline & Workstation)
- Protected Variations
- High Cohesion (van de verschillende Workstation subtypes)

## 9. Controllers - Implementatie

Visual Paradigm for UML Standard Edition(K.U.Leuven)





## 9. Controllers – Design

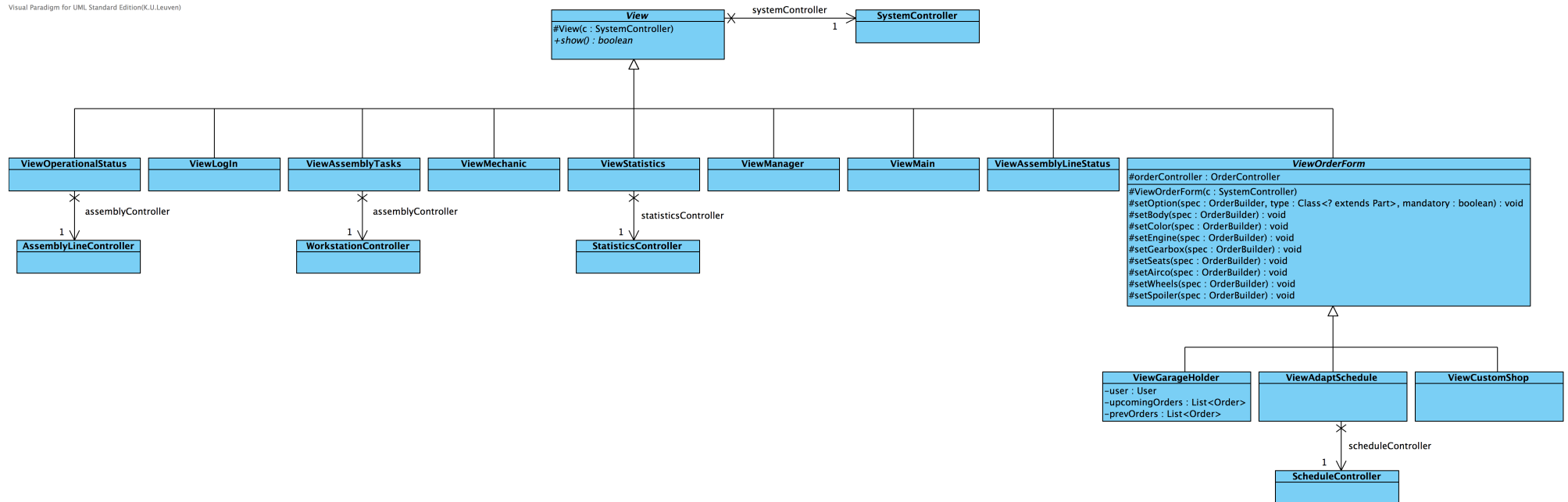
- **Verantwoordelijkheden**
  - Moet een 'dun' laagje bovenop de domein laag zijn
  - Biedt toegang en functionaliteiten
- **GRASP**
  - Controller
  - Indirection
  - Low Coupling

## Opgesplitst in 1 controller per functionaliteit

- Systemcontroller biedt algemene functionaliteiten aan
- Statisticscontroller heeft functies om bepaalde systeemstatistieken te berekenen
- OrderController laat toe om een order te plaatsen
- Workstation-, AssemblyLine- en Schedulercontroller laten toe om hun dedicated object te manipuleren

## 10. View - Implementatie

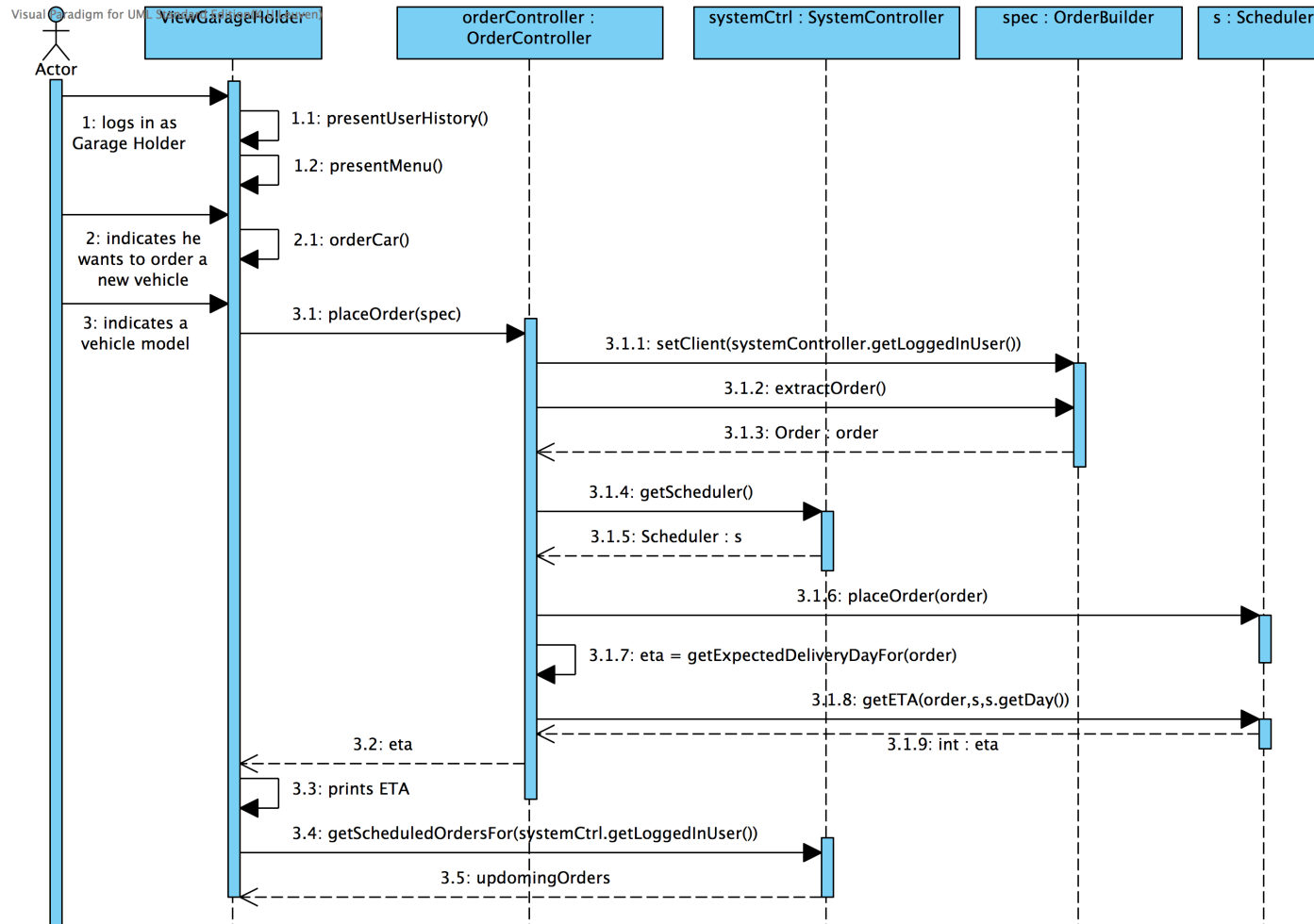
Visual Paradigm for UML, Standard Edition(R.U.Leuven)



## 10. View - Design

- **Verantwoordelijkheden**
  - Alle UI logica zit in de View klassen. De gebruiker interageert enkel met deze View objecten
  - Via verschillende Controllers worden taken naar het systeem gedelegeerd
  - Maakt het mogelijk om UI zaken en domain layer zaken van elkaar gescheiden te houden
  - Elke specifieke View-klasse implementeert een interface View
- **GRASP**
  - Controller
  - Low Coupling














## Design – Order new car example



## Testing

- Unit testing
  - Demo in Eclipse

AssemAssist (May 28, 2014 9:54:46 PM)

Element	Coverage	Covered Instructions	Missed Instructions ▼	Total Instructions
▼ AssemAssist	 83.3 %	4,098	819	4,917
▼ src	 83.3 %	4,098	819	4,917
▶ company.schedule	 41.6 %	469	659	1,128
▶ vehicle.parts	 93.7 %	674	45	719
▶ company.assemblylines	 92.2 %	482	41	523
▶ vehicle.order	 96.3 %	947	36	983
▶ company	 91.0 %	132	13	145
▶ company.workstations	 93.4 %	127	9	136
▶ user	 93.5 %	100	7	107
▶ vehicle.assemblytasks	 95.5 %	107	5	112
▶ vehicle.restrictions	 98.3 %	225	4	229
▶ controllers	 100.0 %	233	0	233
▶ dao	 100.0 %	602	0	602

## Testing

- **Scenario tests**

- We hebben voor alle use cases scenario tests geschreven die controleren dat aan alle scenarios die in de usecases beschreven worden de verwachte afloop wordt gegeven.
- Demo in Eclipse