Thesissen bij FOTS

Uitbreiding, Integratie en Gebruik van open source Modelleringstools

Pieter Van Gorp

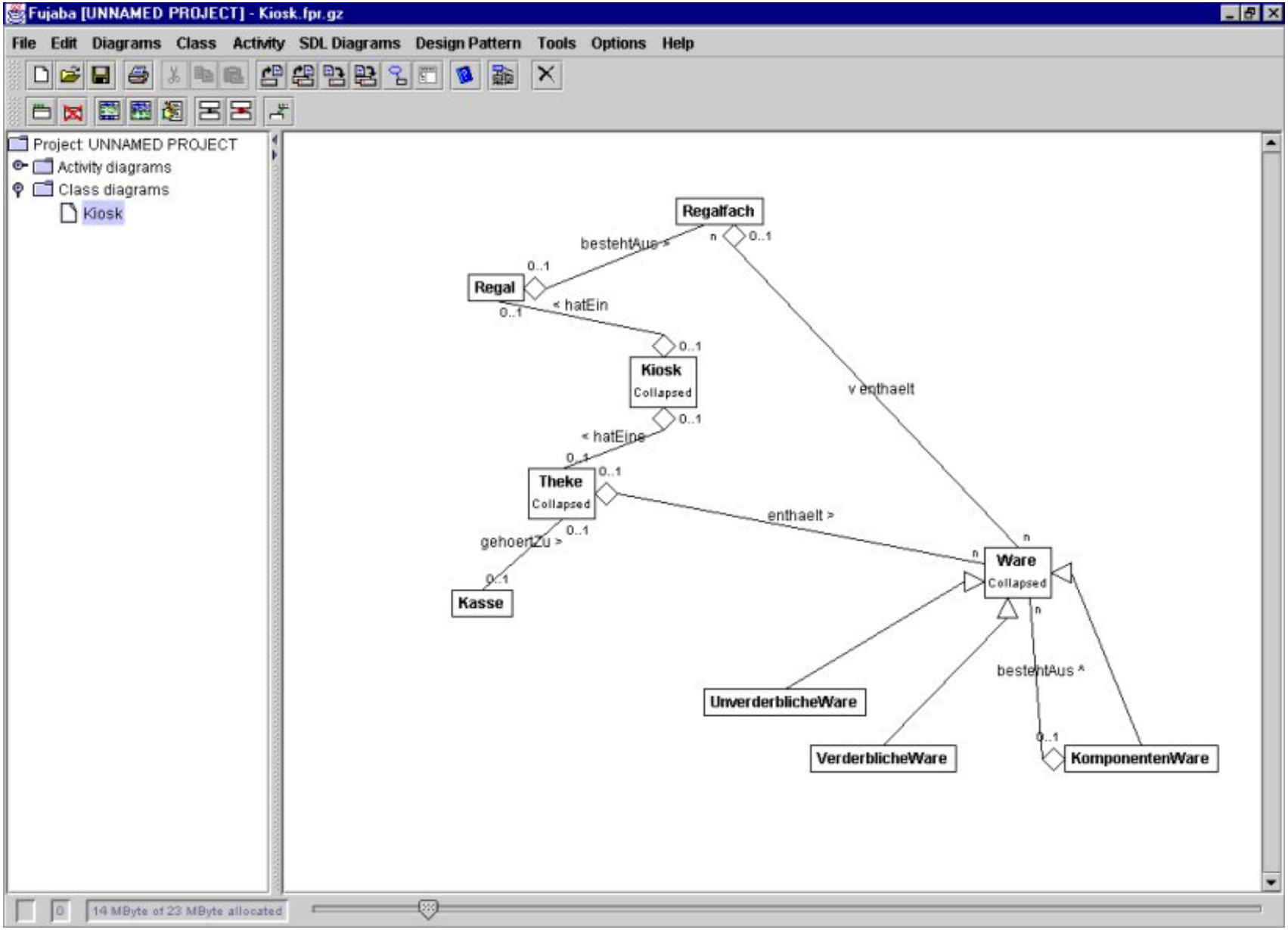
Universiteit Antwerpen

26th April 2006

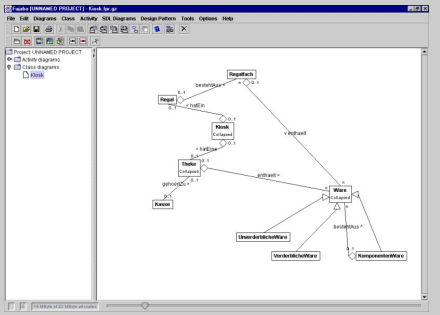
Overzicht

- DOBS= Dynamic Object Browsing System
  - visualizatie object-grafen
  - Toepassingen:
    - ★ Debugging (visualizatie testfaling),
    - ★ “Manueel” testen door object interactie
- Thesis 1:
  - Tool: eDOBS=
    Eclipse implementatie van DOBS
  - Achtergrond= Ontwikkeld binnen FOTS Jaarproject:
    MDR backend i.p.v. algemene Java Heap
  - Doel Thesis= toepassing van eDOBS in zelf te kiezen domein om onder meer debugging toepasbaarheid te evalueren.
- Thesis 2:
  - Tool: MoTMoT=
    Standaard (UML/MOF) compliant graf herschrijvingstool
  - Achtergrond= Ontwikkeld binnen FOTS thesissen, open source
  - Doel thesis= integratie van OCL in Story Diagrammen

Structuur van klassen

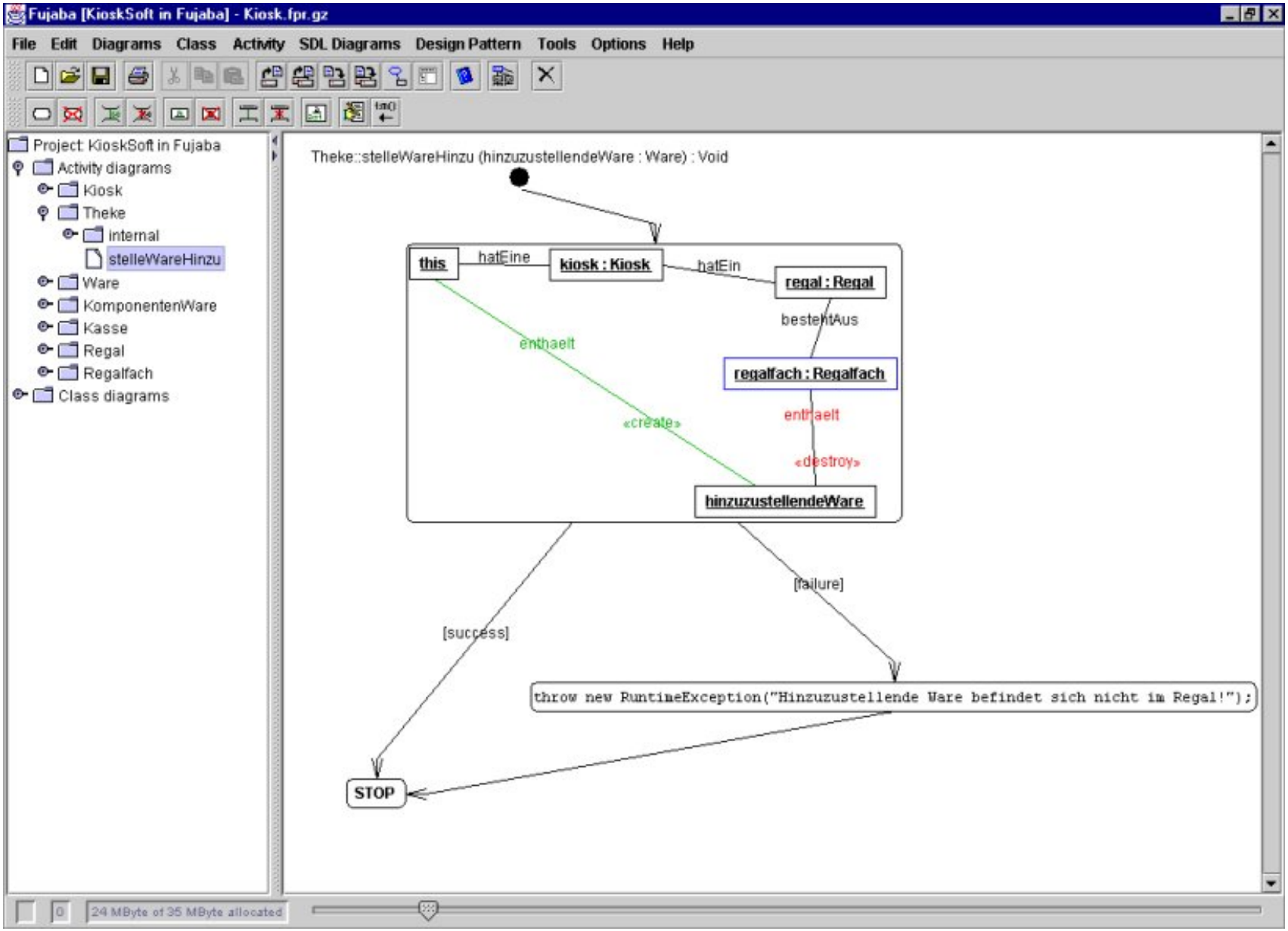


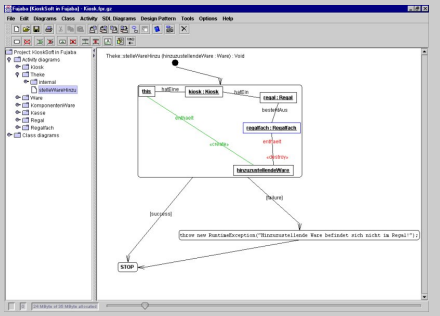
Structuur van klassen



1. Beschouw volgend structureel model van een e-commerce applicatie. Een shop (Kiosk) bestaat uit een rek (Regal) waarin in verschillende vakken (Regalfach) producten (Ware) voorgesteld worden. We beschouwen verderfelijke en niet-verderfelijke waren. De shop bevat verschillende winkelkarretjes (Theke) waarvan men de inhoud aan een kassa (Kasse) dient af te rekenen.

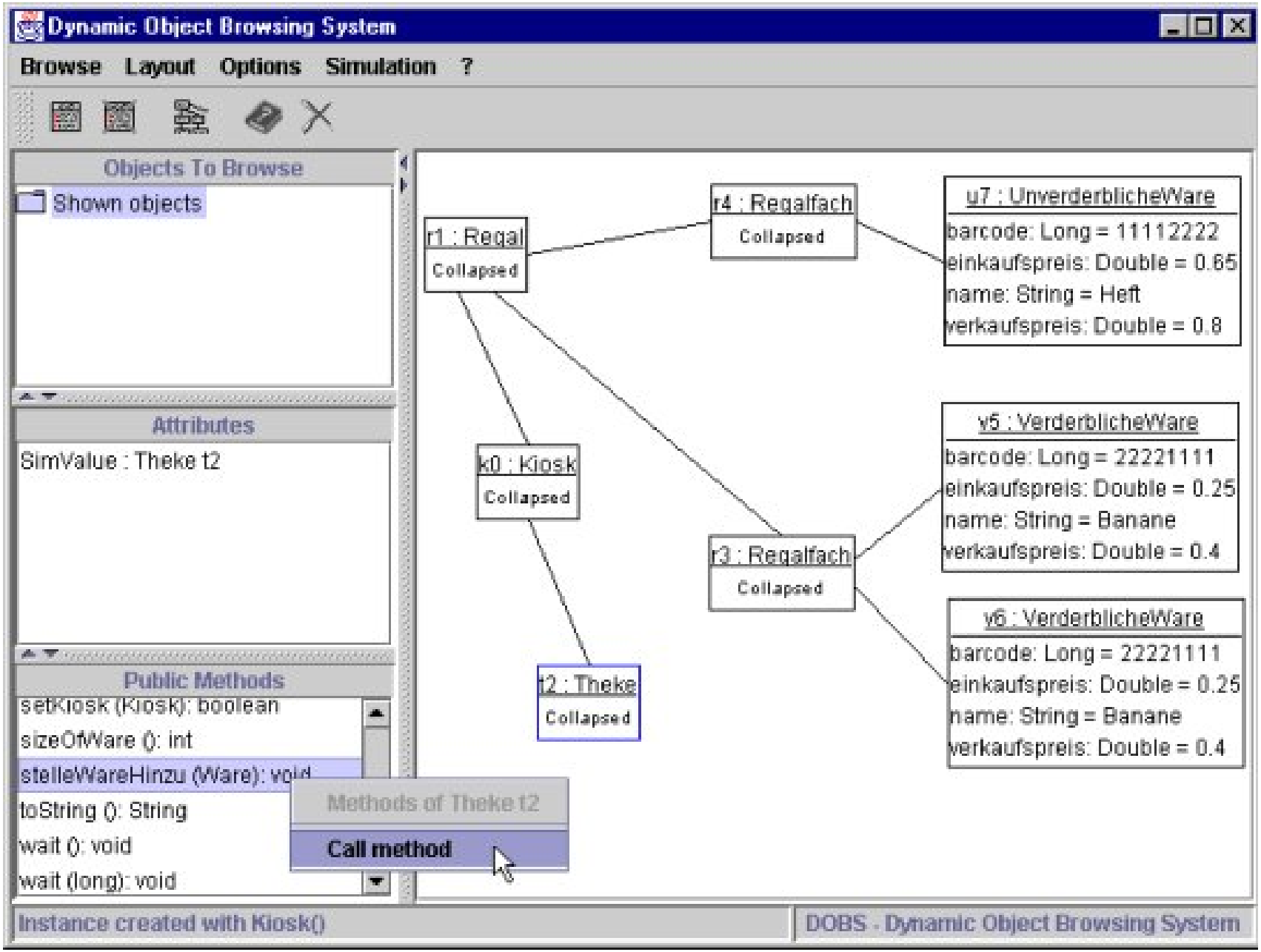
Gedrag van klassen



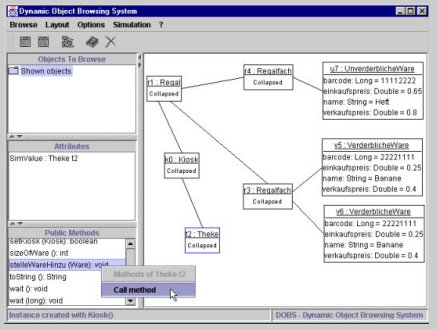


1. Beschouw nu het volgende diagram als voorbeeld van een story diagram. De graph rewrite regel (zie inleiding Dirk) in het midden specifieert hoe een product uit een rek in een winkelkar geplaatst kan worden. Het aan te kopen goed is een parameter van de *stelleWareHinzu* methode die op de Theke klasse (het winkelkarretje) gedefinieerd is. De regel gebruikt een visuele patroon specificatie om het vak op te zoeken waarin het goed in kwestie staat. Merk op dat dit vak een dele moet zijn van een rek in de winkel waarvan we een winkelkar hebben. De rode <<destroy>> markering zorgt ervoor dat he goed uit het vak genomen wordt terwijl de groene <<create>> markering ervoor zorgt dat het goed in de kar geplaatst wordt.

Snapshot van objecten in “Mr. DOBS”

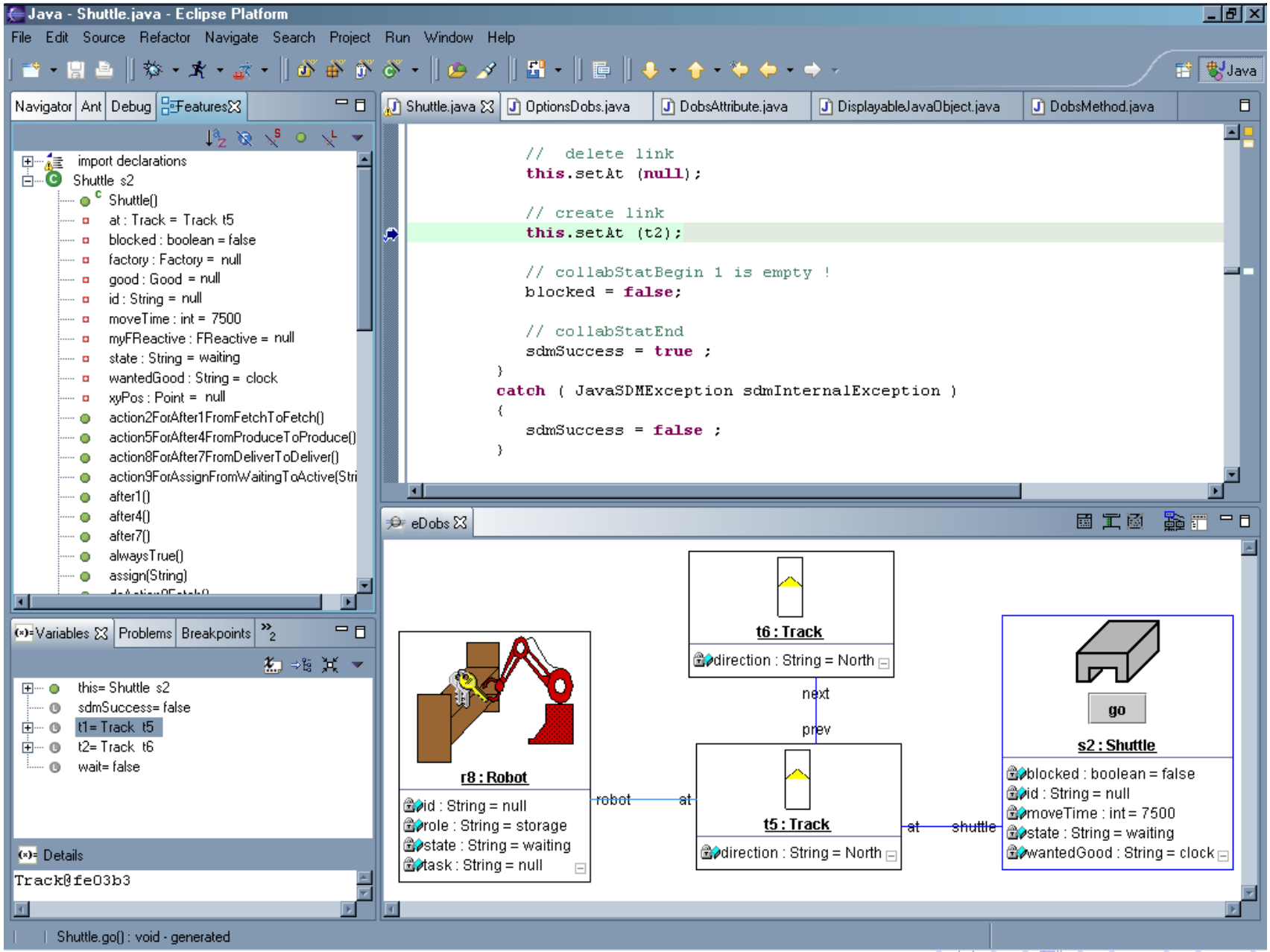


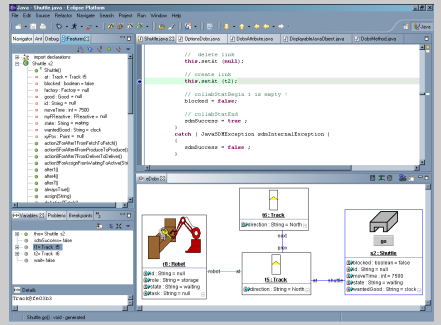
Snapshot van objecten in “Mr. DOBS”



1. Deze slide geeft een toepassing van DOBS weer. Zonder een applicati specifieke GUI ontwikkeld te hebben, wordt hier een aankoop van een goed gesimuleerd aan de hand van de modellen die op de vorige slide gepresenteerd werden. Concreet wordt voor een instance graph met twee vakken, waarop 3 producten staan, de *stelleWareHinzu* methode opgeroepen op een winkel karretje. Dit zal een popup triggeren waarbij de user 1 van de 3 goederen kan selecteren als argument.

eDOBS


Java - Shuttle.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Navigator Ant Debug Features
import declarations
Shuttle s2
Shuttle()
at : Track = Track t5
blocked : boolean = false
factory : Factory = null
good : Good = null
id : String = null
moveTime : int = 7500
myFReactive : FReactive = null
state : String = waiting
wantedGood : String = clock
xyPos : Point = null
action2ForAfter1FromFetchToFetch()
action5ForAfter4FromProduceToProduce()
action8ForAfter7FromDeliverToDeliver()
action9ForAssignFromWaitingToActive(Stri
after1()
after4()
after7()
alwaysTrue()
assign(String)
Variables Problems Breakpoints
this= Shuttle s2
sdmSuccess= false
t1= Track t5
t2= Track t6
wait= false
Details
Track@fe03b3
Shuttle.java OptionsDobs.java DobsAttribute.java DisplayableJavaObject.java DobsMethod.java
// delete link
this.setAt (null);
// create link
this.setAt (t2);
// collabStatBegin 1 is empty !
blocked = false;
// collabStatEnd
sdmSuccess = true ;
}
catch (JavaSDMException sdmInternalException)
{
sdmSuccess = false ;
}
eDobs
t6 : Track
direction : String = North
next
prev
t5 : Track
direction : String = North
r8 : Robot
id : String = null
role : String = storage
state : String = waiting
task : String = null
robot
at
at
shuttle
go
s2 : Shuttle
blocked : boolean = false
id : String = null
moveTime : int = 7500
state : String = waiting
wantedGood : String = clock
Shuttle.go() : void - generated



1. Volgende slide geeft 3 aspecten aan. *Enerzijds* wordt geillustreerd dat het gepresenteerde concept toepasbaar is voor elke applicatie die object-georienteerd gemodelleerd kan worden. De thesis kan zich dus baseren op case studies die gaan van web e-commerce toepassingen tot assembly applicaties met een onderliggende Lego MindStorm infrastructuur. *Anderzijds* geeft de slide aan hoe eDOBS kan toegepast worden in een debugging context. Er werd namelijk een Eclipse breakpoint geplaatst op de lijn code die een shuttle van een track verwijdert nadat een robot er goederen op gezet heeft. *Tenslotte* geeft de slide aan dat objecten een visualizatie kunnen toegewezen krijgen. Instanties van de klasse Robot zien er bijvoorbeeld heel anders uit dan die van klasse Shuttle.

Concreet: de thesissen

- Thesis 1:
  - Evalueer de toepasbaarheid van eDOBS in een bepaald ontwikkelingsproces.
- Thesis 2:
  - Integreer OCL in Story Diagrammen
  - Maw: breid de visuele taal die gebruikt wordt om gedrag te specificeren uit met een tekstuele, side-effect-free query taal.
  - Motivatie: beide approaches hebben hun voordelen
    - ★ compactheid van tekst,
    - ★ begrijpbaarheid van diagrammen,
    - ★ ...
  - Toegepast op het e-commerce voorbeeld kan in de *stelleWareHinzu* methode het pad tussen this en regalfach vervangen worden door:

Example

where regalfach= this.kiosk.regal.regalfach->*select*(r | r.hinzustellendeWare->*contains*(hinzustellendeWare))->*first*()