OptaPlanner

Domain models and design patterns

by Geoffrey De Smet

OptaPlanner lead

Announcements

- OptaPlanner Quick Starts repository github.com/kiegroup/optaplanner-quickstarts (https://github.com/kiegroup/optaplanner-quickstarts)

Announcements

- OptaPlanner Quick Starts repository github.com/kiegroup/optaplanner-quickstarts (https://github.com/kiegroup/optaplanner-quickstarts)
- Quick Starts Showcase

Announcements

- OptaPlanner Quick Starts repository github.com/kiegroup/optaplanner-quickstarts (https://github.com/kiegroup/optaplanner-quickstarts)
- Quick Starts Showcase

DEMO

Why is modeling hard? (*)

Why is modeling hard? (*)

(*) at least the first few times

Given a text like this...

School timetabling

Optimize a school timetable of lessons
to assign teachers and students
in the best room at the best time.

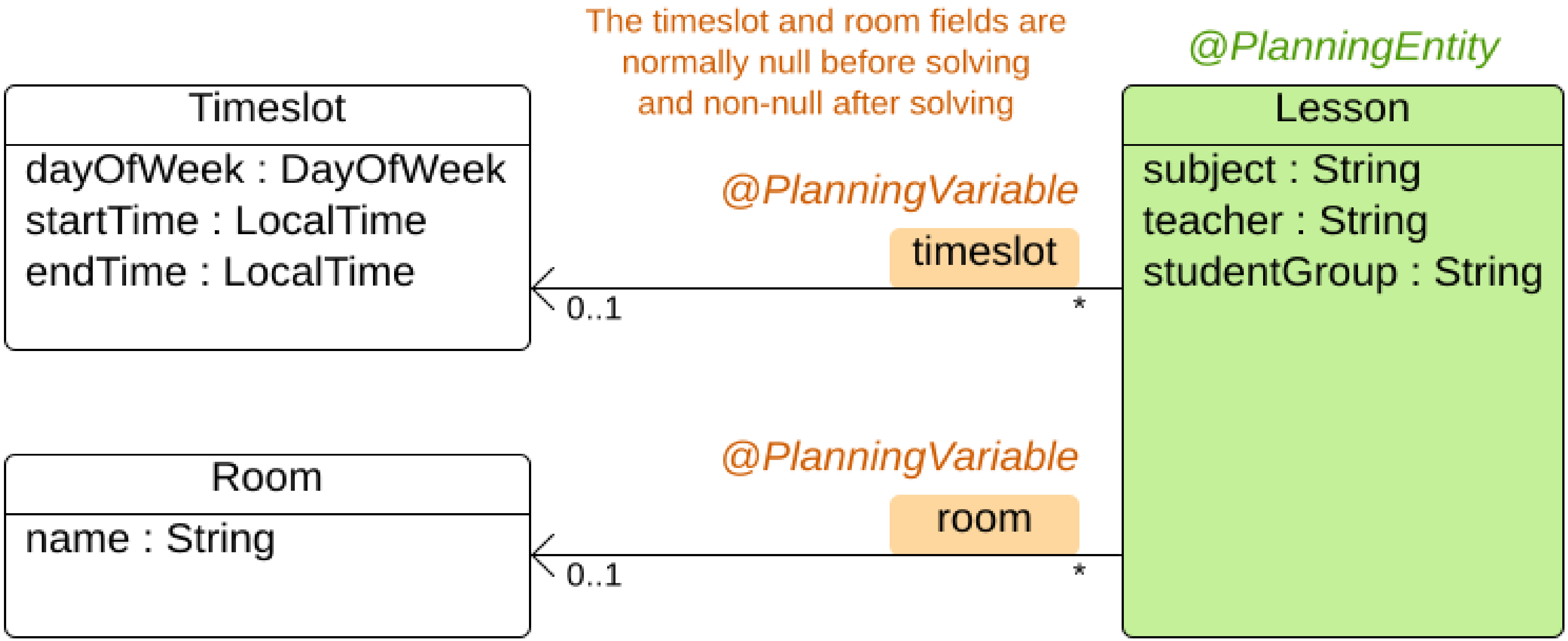
Given a text like this...

School timetabling

Optimize a school timetable of lessons
to assign teachers and students
in the best room at the best time.

Come up with a model like this...

Time table class diagram



Let's take it step by step...

School timetabling

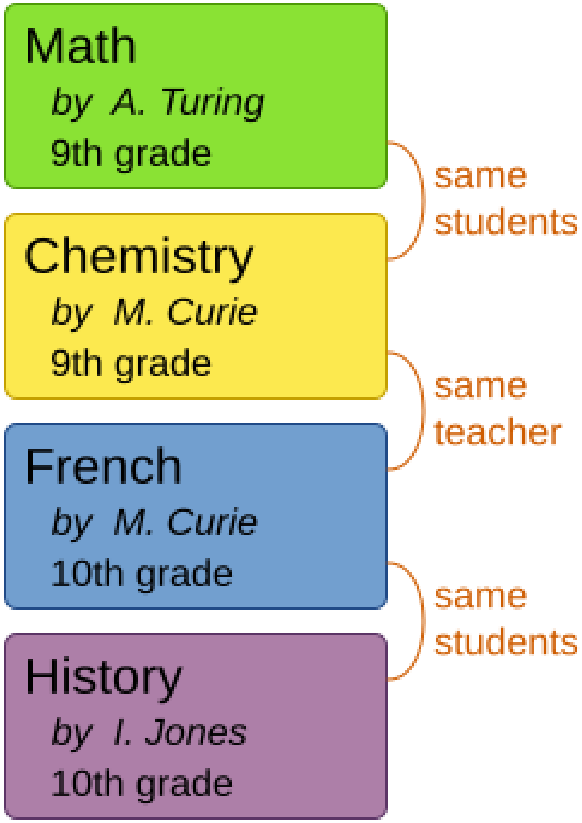
Optimize a school timetable of lessons
to assign teachers and students
in the best room at the best time.

What changes during planning?

School timetabling input/output

Assign each lesson to a time slot and a room.

Lessons



Math
by A. Turing
9th grade

Chemistry
by M. Curie
9th grade

French
by M. Curie
10th grade

History
by I. Jones
10th grade

same students

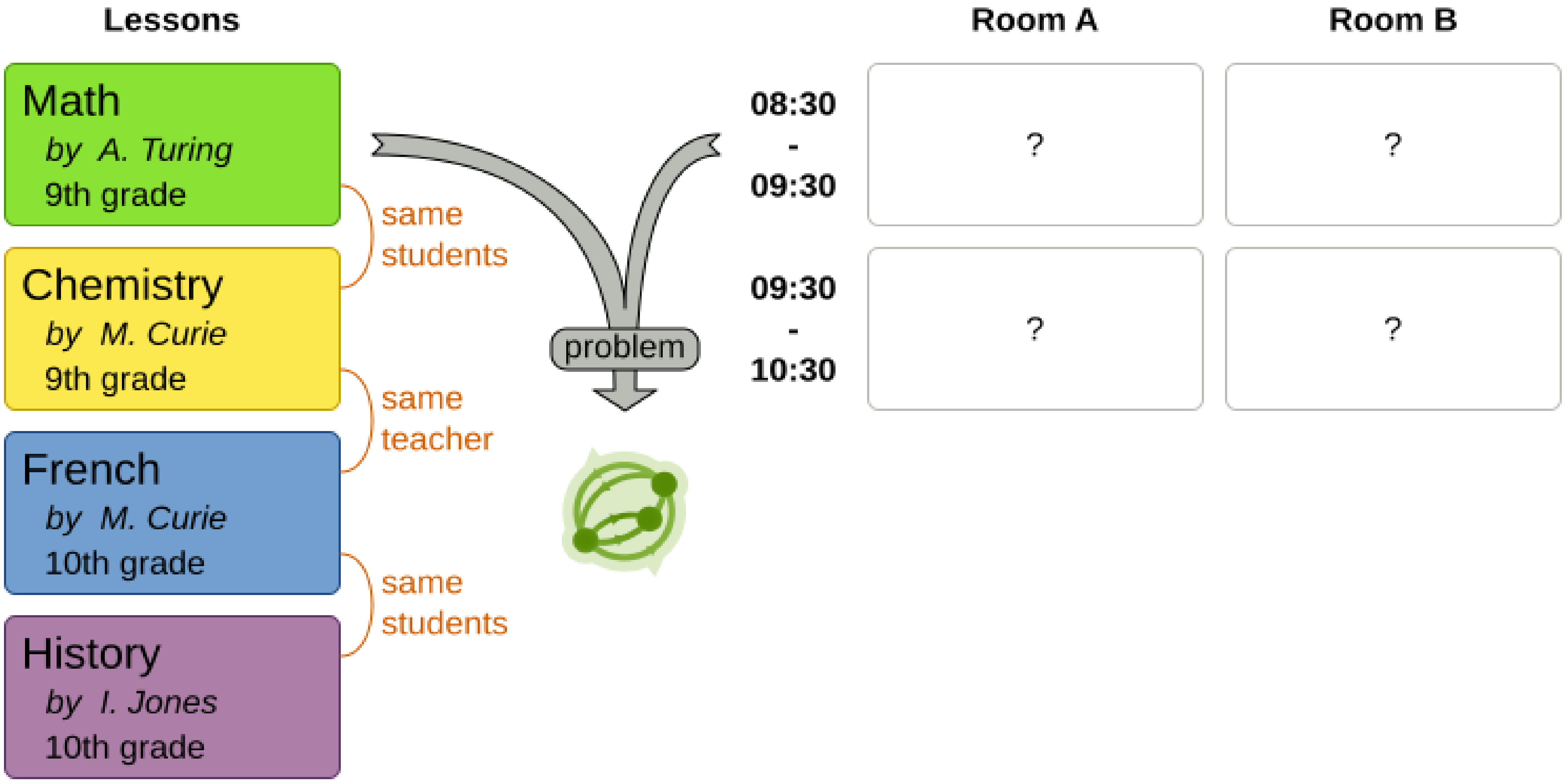
same teacher

same students

	Room A	Room B
08:30 - 09:30	?	?
09:30 - 10:30	?	?

School timetabling input/output

Assign each lesson to a time slot and a room.



Lessons

Math
by A. Turing
9th grade

Chemistry
by M. Curie
9th grade

French
by M. Curie
10th grade

History
by I. Jones
10th grade

same students

same teacher

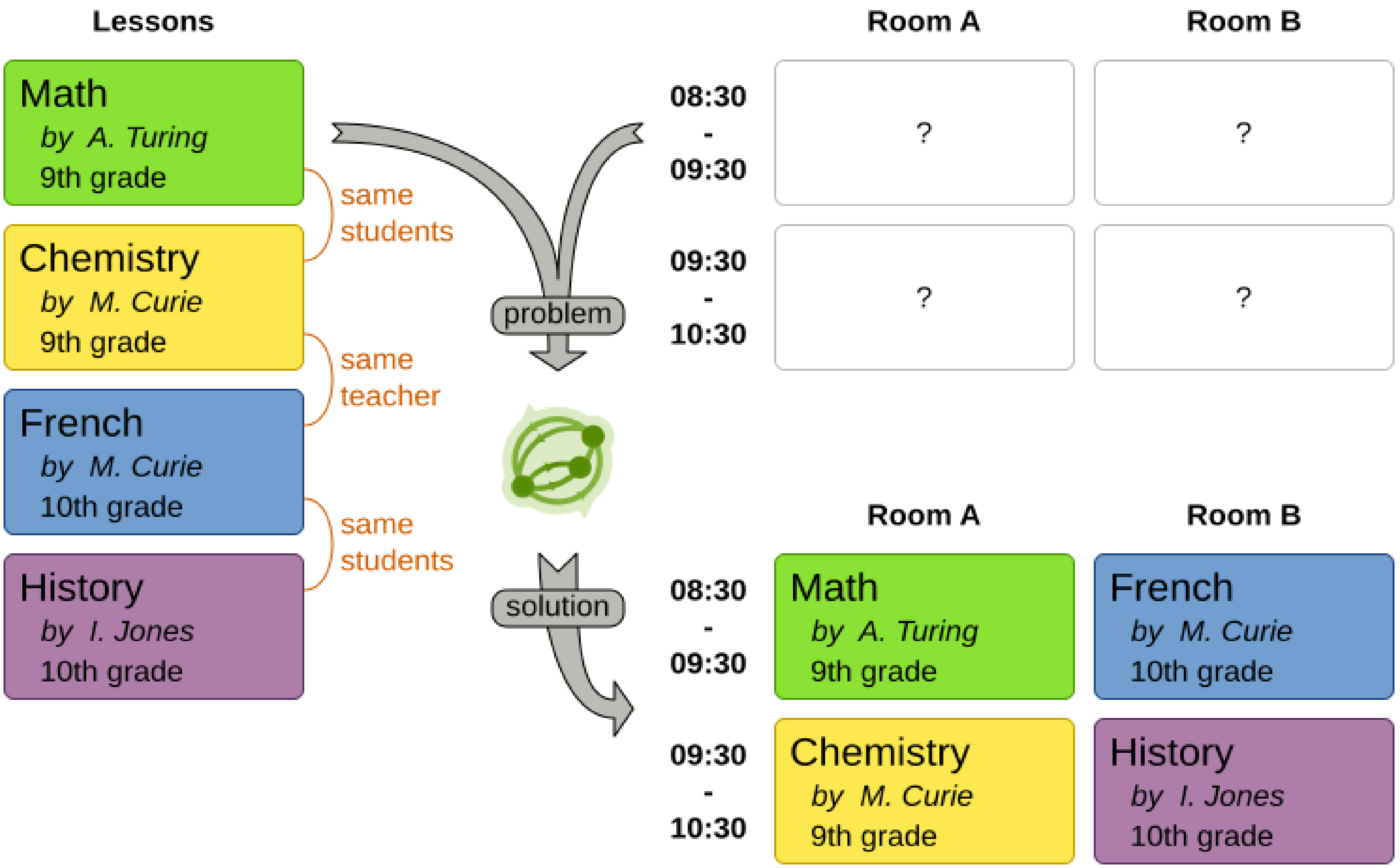
same students

problem

	Room A	**Room B**
08:30 - 09:30	?	?
09:30 - 10:30	?	?

School timetabling input/output

Assign each lesson to a time slot and a room.



Lessons

- Math *by A. Turing* 9th grade
- Chemistry *by M. Curie* 9th grade
- French *by M. Curie* 10th grade
- History *by I. Jones* 10th grade

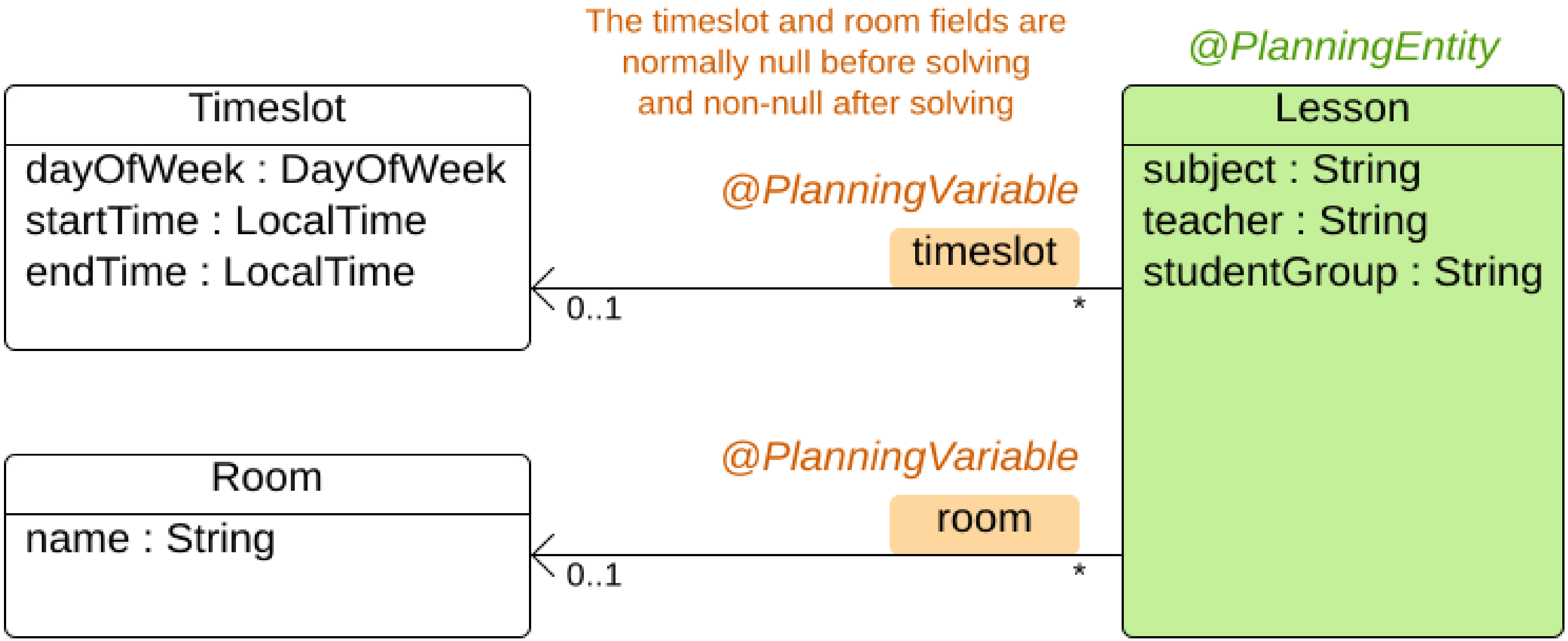
	Room A	**Room B**
08:30 - 09:30	?	?
09:30 - 10:30	?	?

	Room A	**Room B**
08:30 - 09:30	Math *by A. Turing* 9th grade	French *by M. Curie* 10th grade
09:30 - 10:30	Chemistry *by M. Curie* 9th grade	History *by I. Jones* 10th grade

What changes during planning?

- The assigned timeslot of each lesson
- The assigned room of each lesson

Time table class diagram



What are the constraints?

What are the constraints?

1. Which hard constraints are build-in in our model?

What are the constraints?

1. Which hard constraints are build-in in our model?
2. Which constraints affect our planning variables?

Constraints inventory

- Each lesson must have one timeslot
- Each lesson must have one room
- No lessons in the same room together
- No teacher with lessons at the same time
- Students must be able to attend all lessons
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room
- No lessons in the same room together
- No teacher with lessons at the same time
- Students must be able to attend all lessons
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together
- No teacher with lessons at the same time
- Students must be able to attend all lessons
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together (hard)
- No teacher with lessons at the same time
- Students must be able to attend all lessons
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together (hard)
- No teacher with lessons at the same time (hard)
- Students must be able to attend all lessons
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together (hard)
- No teacher with lessons at the same time (hard)
- Students must be able to attend all lessons (hard)
- Use each teacher's time efficiently
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together (hard)
- No teacher with lessons at the same time (hard)
- Students must be able to attend all lessons (hard)
- Use each teacher's time efficiently (soft)
- Give students variety in subjects

Constraints inventory

- Each lesson must have one timeslot (hard)
- Each lesson must have one room (hard)
- No lessons in the same room together (hard)
- No teacher with lessons at the same time (hard)
- Students must be able to attend all lessons (hard)
- Use each teacher's time efficiently (soft)
- Give students variety in subjects (soft)

Which hard constraints are build-in in our model?

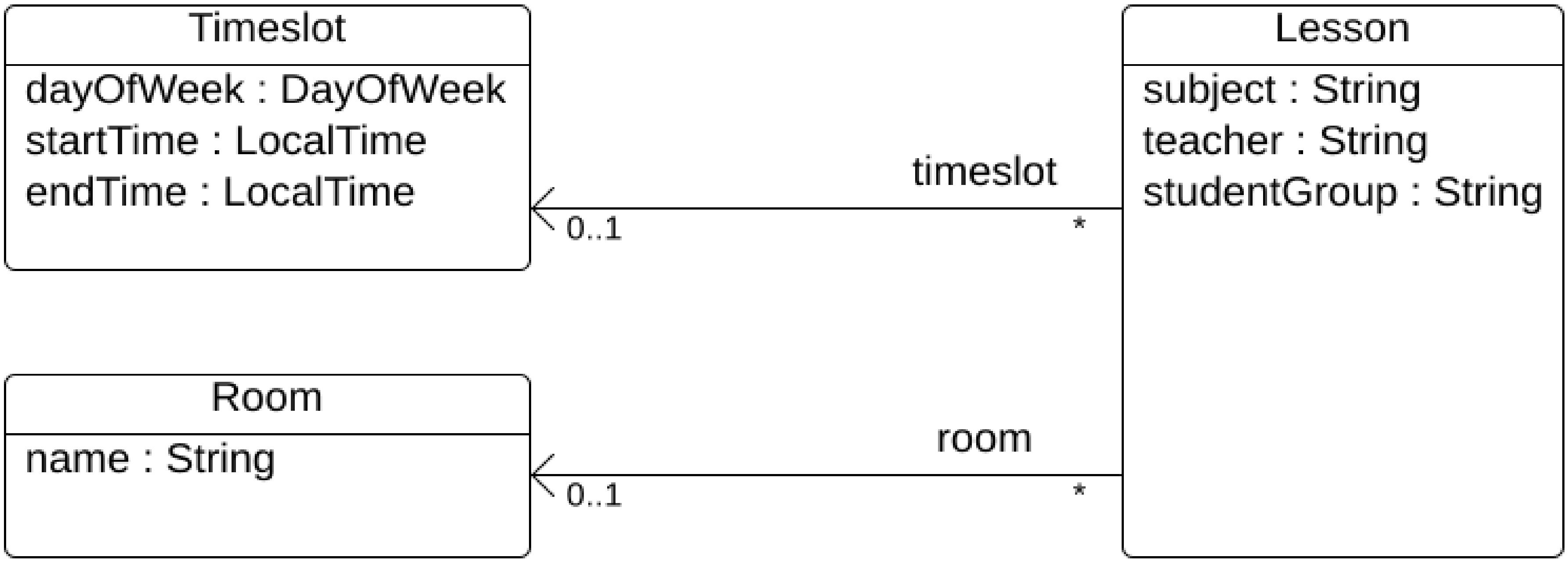
Which hard constraints are build-in in our model?

- Each lesson must have one timeslot (build-in hard)

Which hard constraints are build-in in our model?

- Each lesson must have one timeslot (build-in hard)
- Each lesson must have one room (build-in hard)

Time table class diagram



Hard constraints (not build-in)

- **Room conflict**: No 2 lessons in the same room at the same time
- **Teacher conflict**: No 2 lessons for the same teacher at the same time
- **Student conflict**: No 2 lessons for the same student group at the same time

Hard constraints (not build-in)

- **Room conflict**: No 2 lessons in the same room at the same time
- **Teacher conflict**: No 2 lessons for the same teacher at the same time
- **Student conflict**: No 2 lessons for the same student group at the same time

Formalize your constraints.

Soft constraints

- **Teacher time efficiently**: No gap between teacher lessons
- **Subject variety**: No sequential lessons on the same subject

Some constraints affect the planning variables.

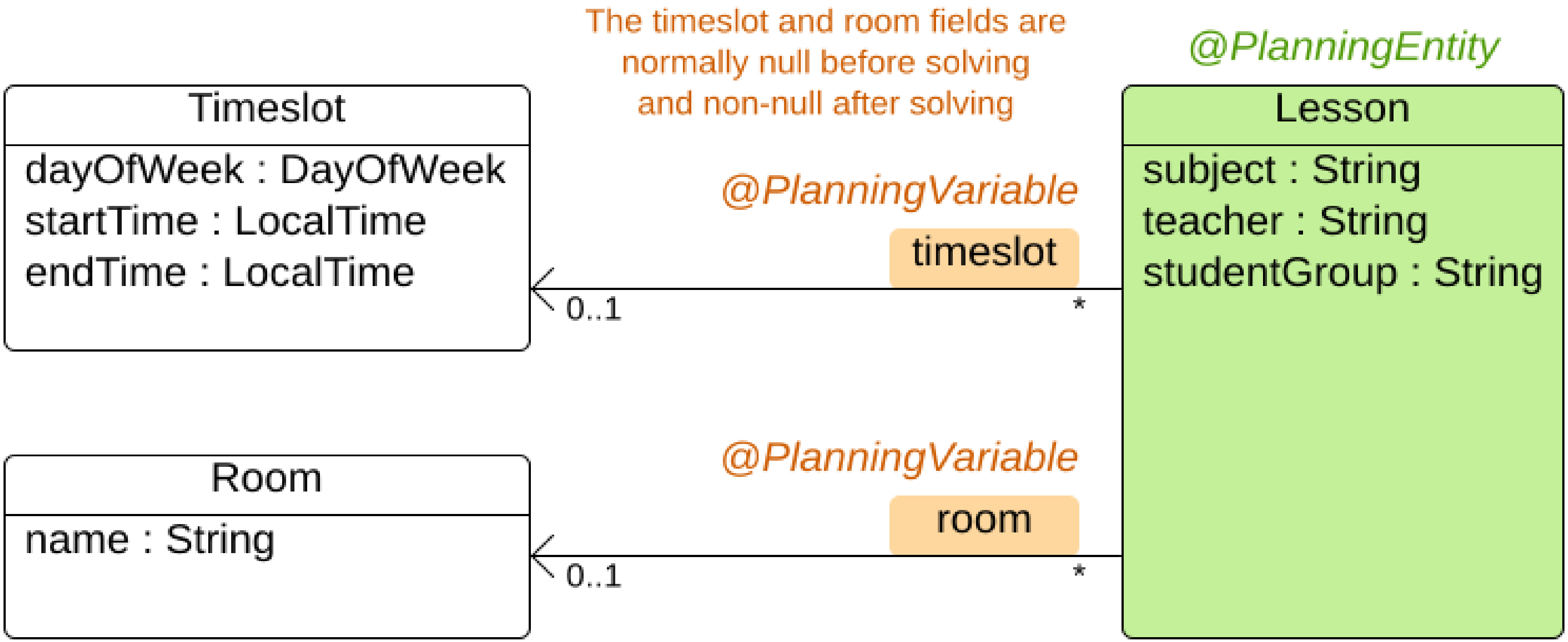
Most don't.

Some constraints affect the planning variables.

Most don't.

Learn to tell them apart:
Do these planning variables hinder any of the constraints?

Time table class diagram



Constraints that don't affect the model

- Skill requirements and affinity
- Availability and unavailability
- Fairness and load balancing
- Time windows,
- ... (many more)

Constraints that don't affect the model

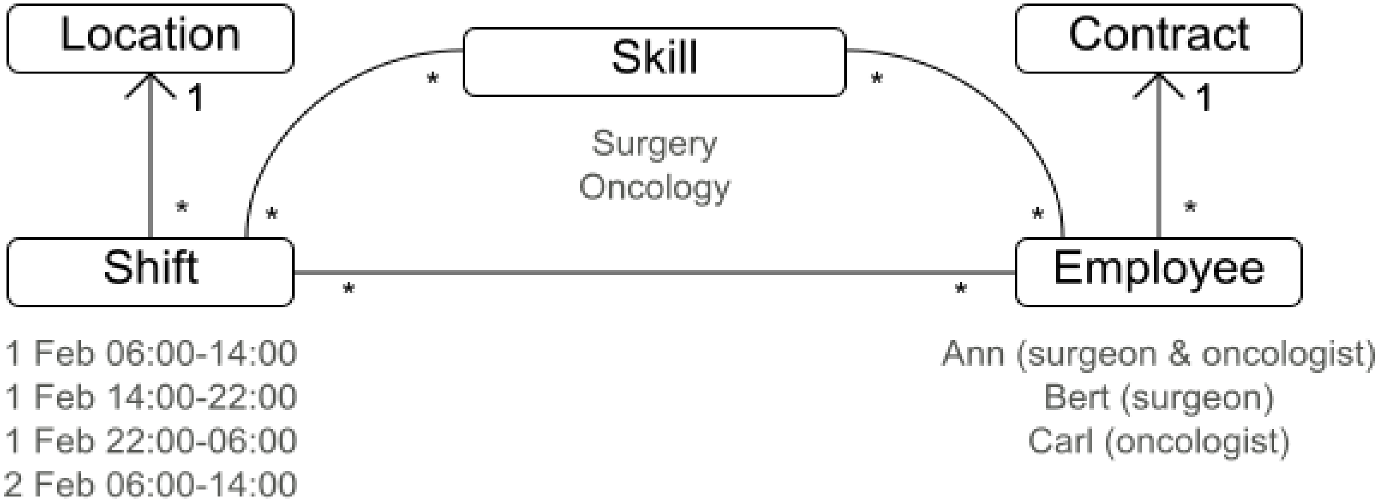
- Skill requirements and affinity
- Availability and unavailability
- Fairness and load balancing
- Time windows,
- ... (many more)

Most constraints do not affect the model!

The PlanningSolution is easy

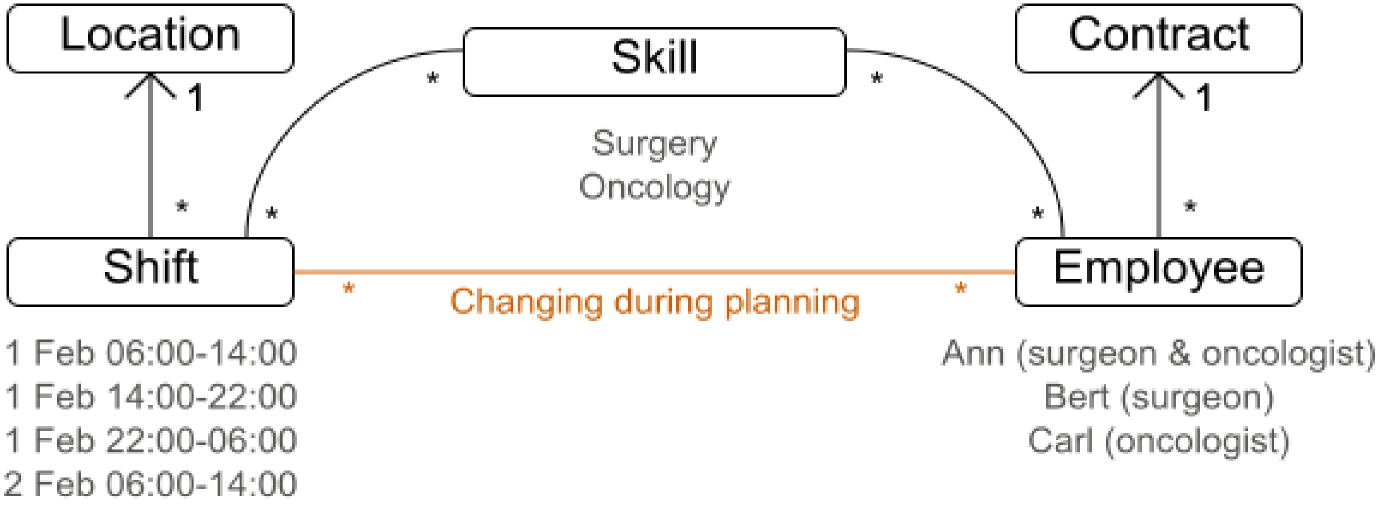
Modeling tips

Employee shift rostering modeling guide



Employee shift rostering modeling guide

What changes during planning?

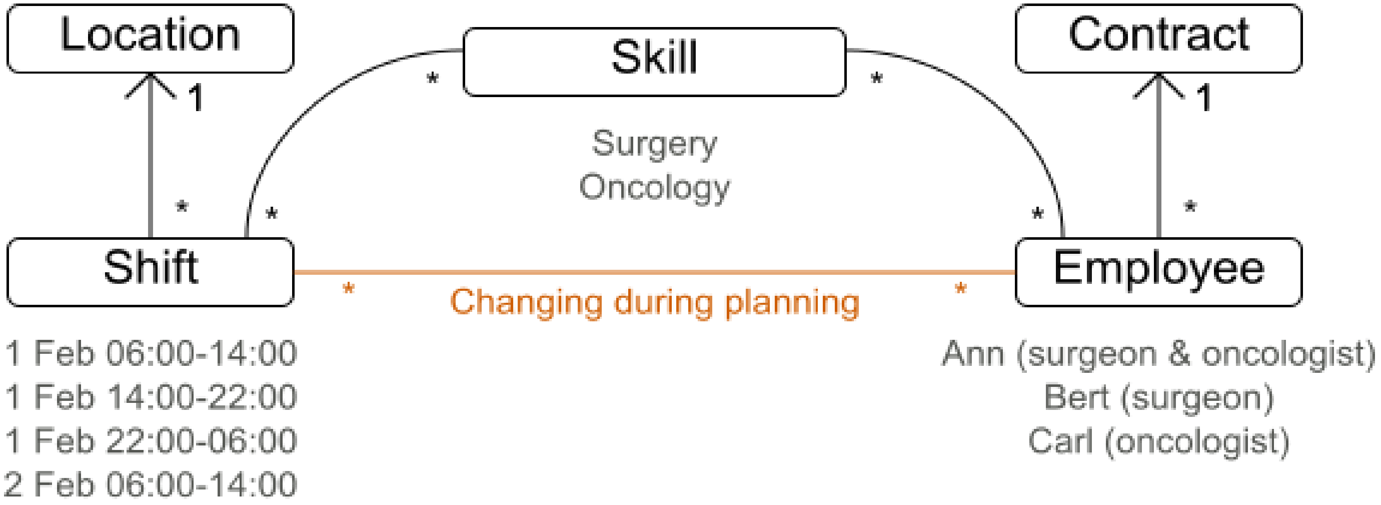


Find what changes

What is fixed in the input problem?
What can OptaPlanner change in the output solution?

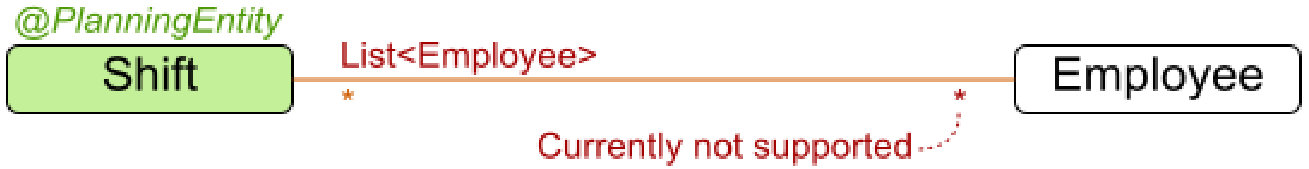
Employee shift rostering modeling guide

What changes during planning?



Find what changes

What is fixed in
the input problem?
What can
OptaPlanner change
in the output solution?

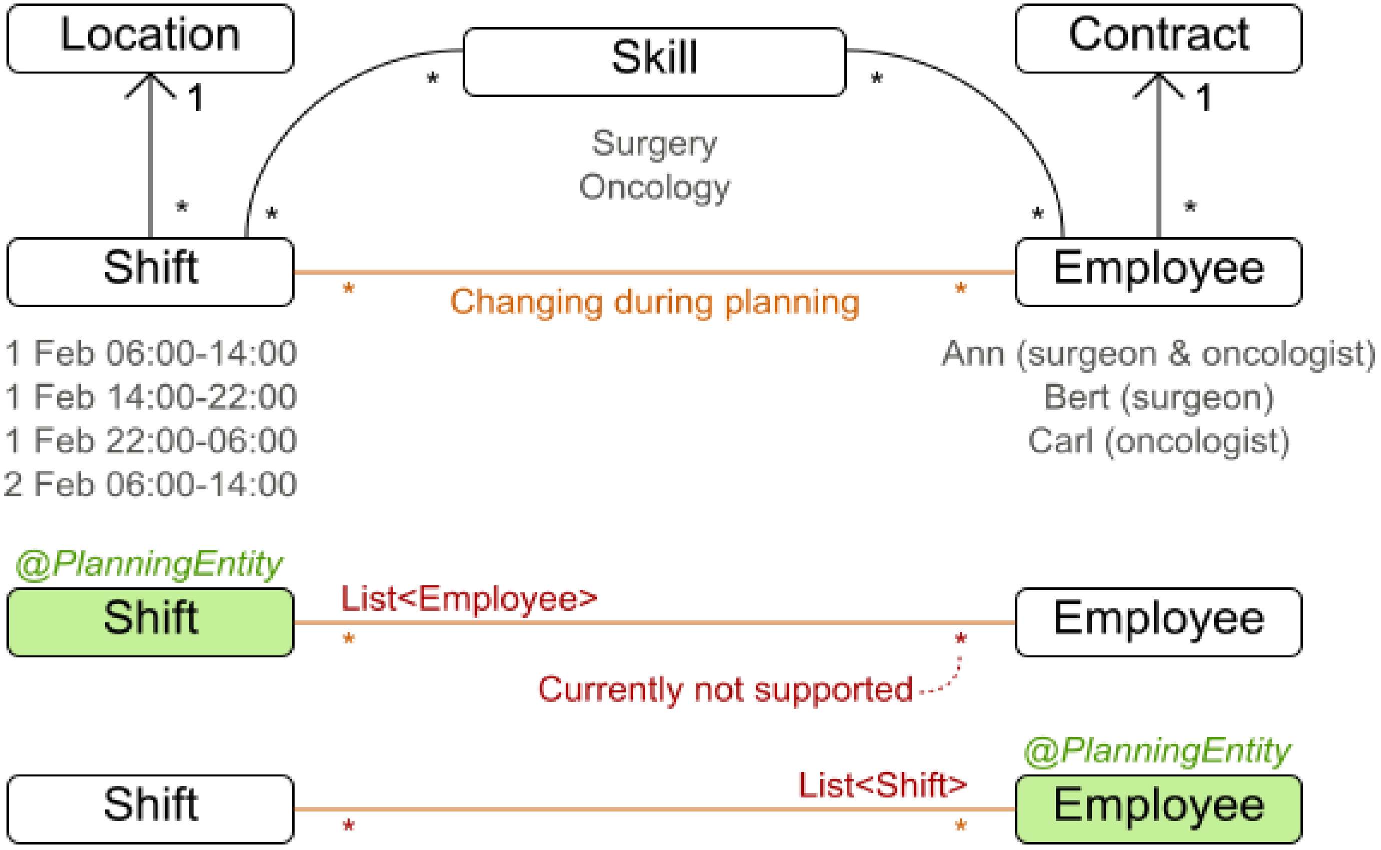


Bad model

Planning variable is a
one to many relationship

Employee shift rostering modeling guide

What changes during planning?



Find what changes

What is fixed in the input problem?
What can OptaPlanner change in the output solution?

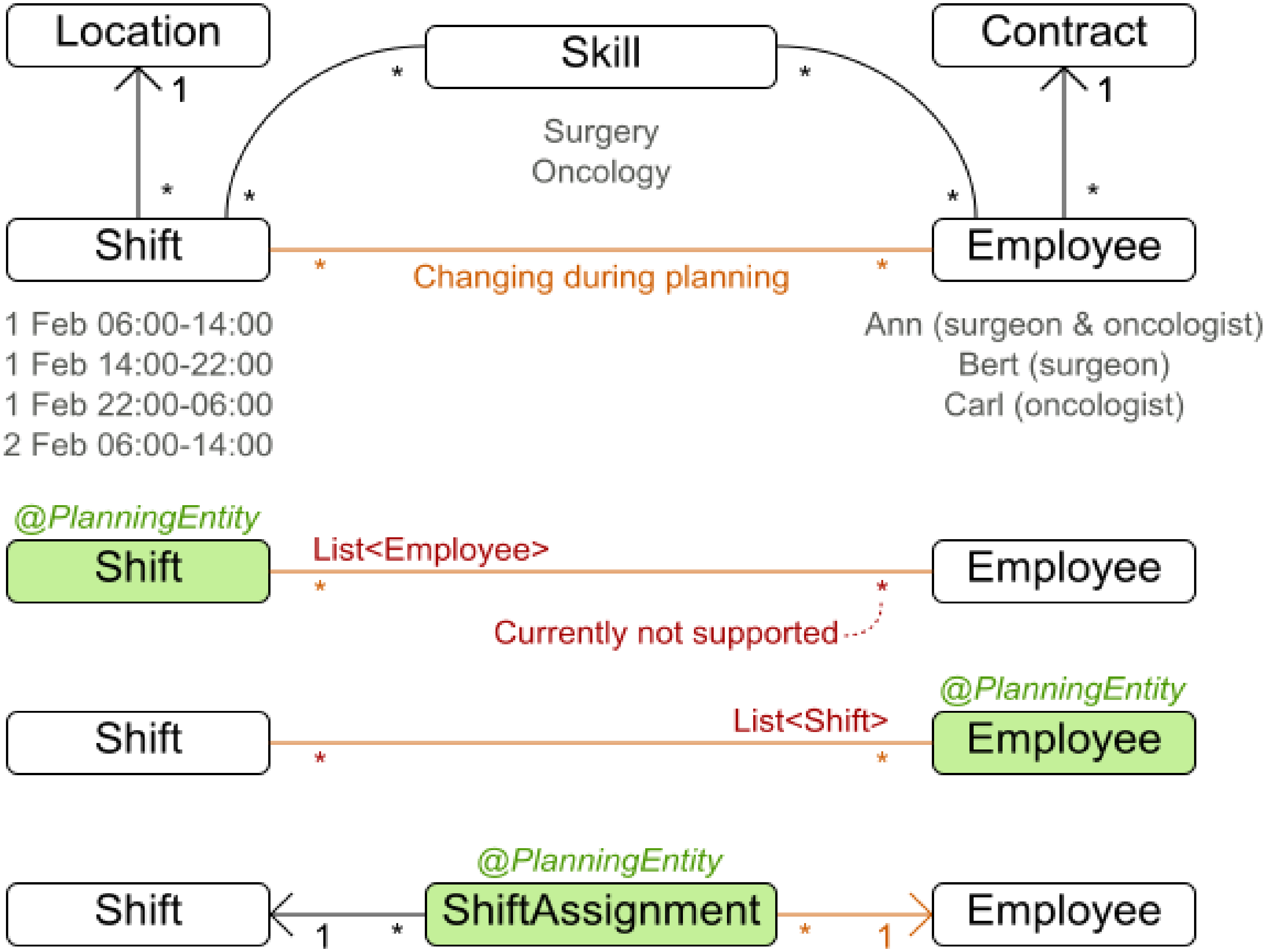
Bad model

Planning variable is a one to many relationship

Bad model

Planning variable is a one to many relationship

Employee shift rostering modeling guide



Find what changes

What is fixed in the input problem? What can OptaPlanner change in the output solution?

Bad model

Planning variable is a one to many relationship

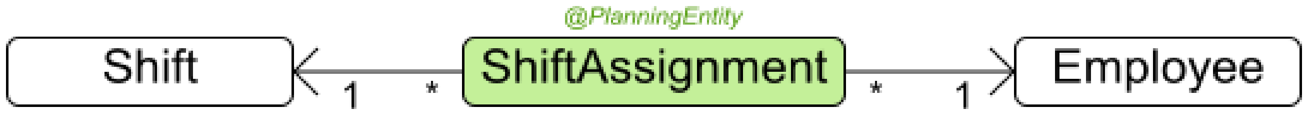
Bad model

Planning variable is a one to many relationship

Good model

Planning variable is a many to one relationship

Employee shift rostering modeling guide



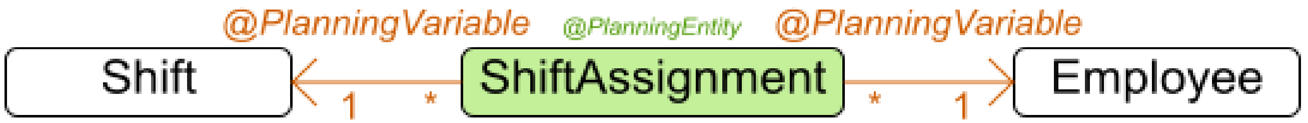
@PlanningEntity

Shift 1 * ShiftAssignment * 1 Employee

1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

Ann
Bert
Carl

Employee shift rostering modeling guide



@PlanningVariable

@PlanningEntity

@PlanningVariable

Shift

ShiftAssignment

Employee

1

*

*

1

1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

(0) → 1 Feb 06:00 + Ann
(1) → 1 Feb 06:00 + Bert
(2) → 1 Feb 14:00 + Ann
(3) → 1 Feb 14:00 + Carl

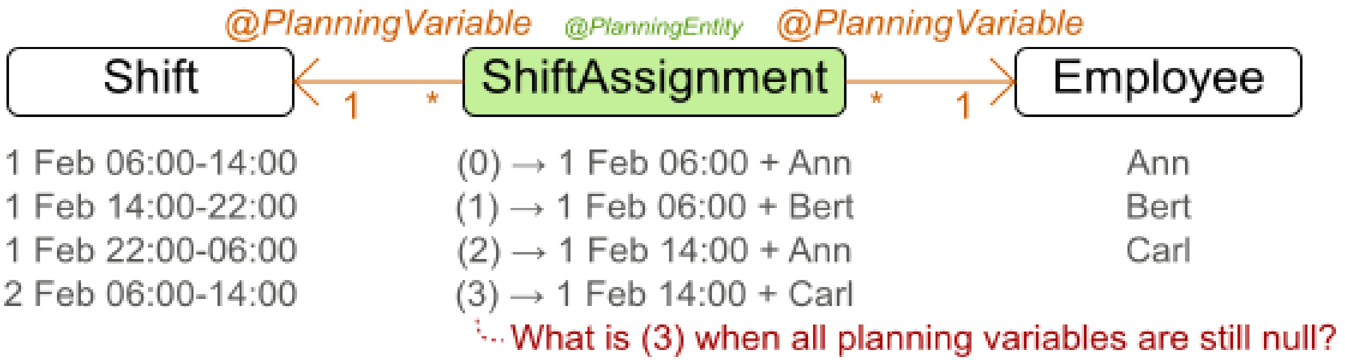
What is (3) when all planning variables are still null?

Ann
Bert
Carl

Bad model

ShiftAssignment lacks a business identification. The 2 planning variables make the search space a lot larger than necessary.

Employee shift rostering modeling guide



@PlanningVariable @PlanningEntity @PlanningVariable

Shift 1 * ShiftAssignment * 1 Employee

1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

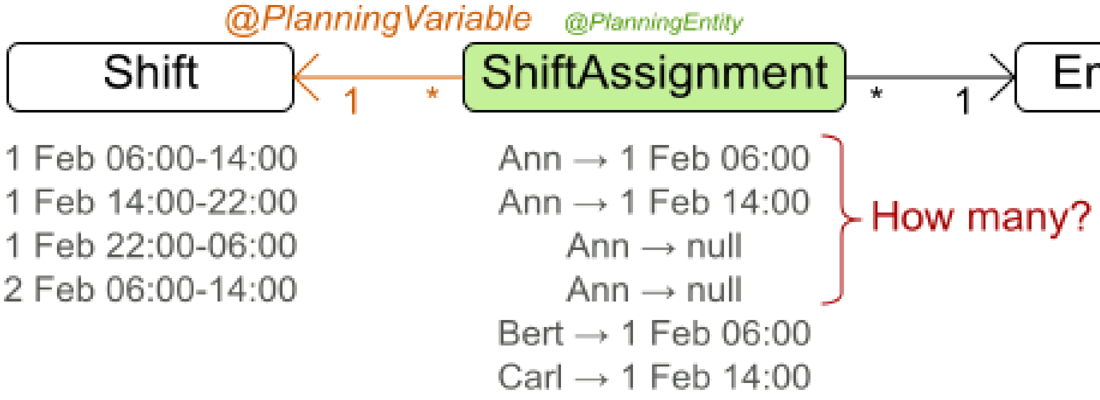
(0) → 1 Feb 06:00 + Ann
(1) → 1 Feb 06:00 + Bert
(2) → 1 Feb 14:00 + Ann
(3) → 1 Feb 14:00 + Carl

What is (3) when all planning variables are still null?

Ann
Bert
Carl

Bad model

ShiftAssignment lacks
a business identification.
The 2 planning variables
make the search space
a lot larger
than necessary.



@PlanningVariable @PlanningEntity

Shift 1 * ShiftAssignment * 1 Employee

1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

Ann → 1 Feb 06:00
Ann → 1 Feb 14:00
Ann → null
Ann → null
Bert → 1 Feb 06:00
Carl → 1 Feb 14:00

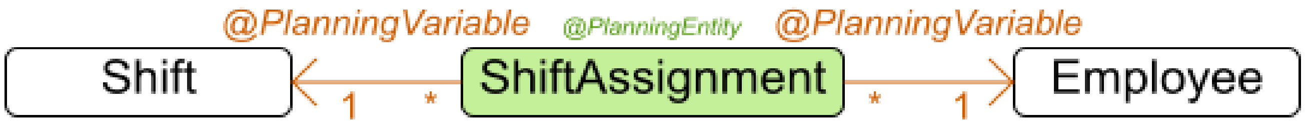
How many?

Ann
Bert
Carl

Bad model

The number of shifts
per employee
is impossible to determine
in advance:
it differs per solution.
The nulls make the search
space a bit larger
than necessary.

Employee shift rostering modeling guide



1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

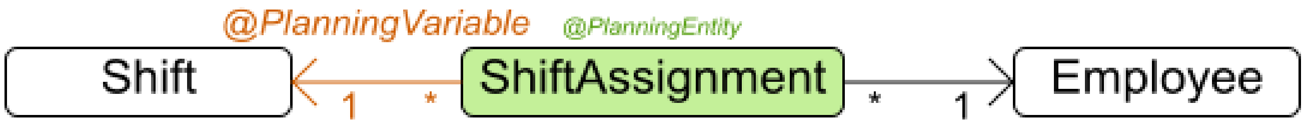
(0) → 1 Feb 06:00 + Ann
(1) → 1 Feb 06:00 + Bert
(2) → 1 Feb 14:00 + Ann
(3) → 1 Feb 14:00 + Carl

What is (3) when all planning variables are still null?

Ann
Bert
Carl

Bad model

ShiftAssignment lacks
a business identification.
The 2 planning variables
make the search space
a lot larger
than necessary.



1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

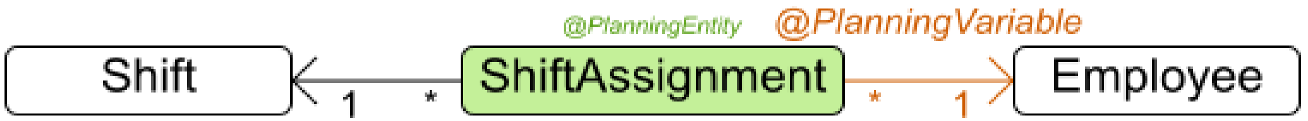
Ann → 1 Feb 06:00
Ann → 1 Feb 14:00
Ann → null
Ann → null
Bert → 1 Feb 06:00
Carl → 1 Feb 14:00

How many?

Ann
Bert
Carl

Bad model

The number of shifts
per employee
is impossible to determine
in advance:
it differs per solution.
The nulls make the search
space a bit larger
than necessary.



1 Feb 06:00-14:00
1 Feb 14:00-22:00
1 Feb 22:00-06:00
2 Feb 06:00-14:00

1 Feb 06:00 → Ann
1 Feb 06:00 → Bert
1 Feb 14:00 → Ann
1 Feb 14:00 → Carl

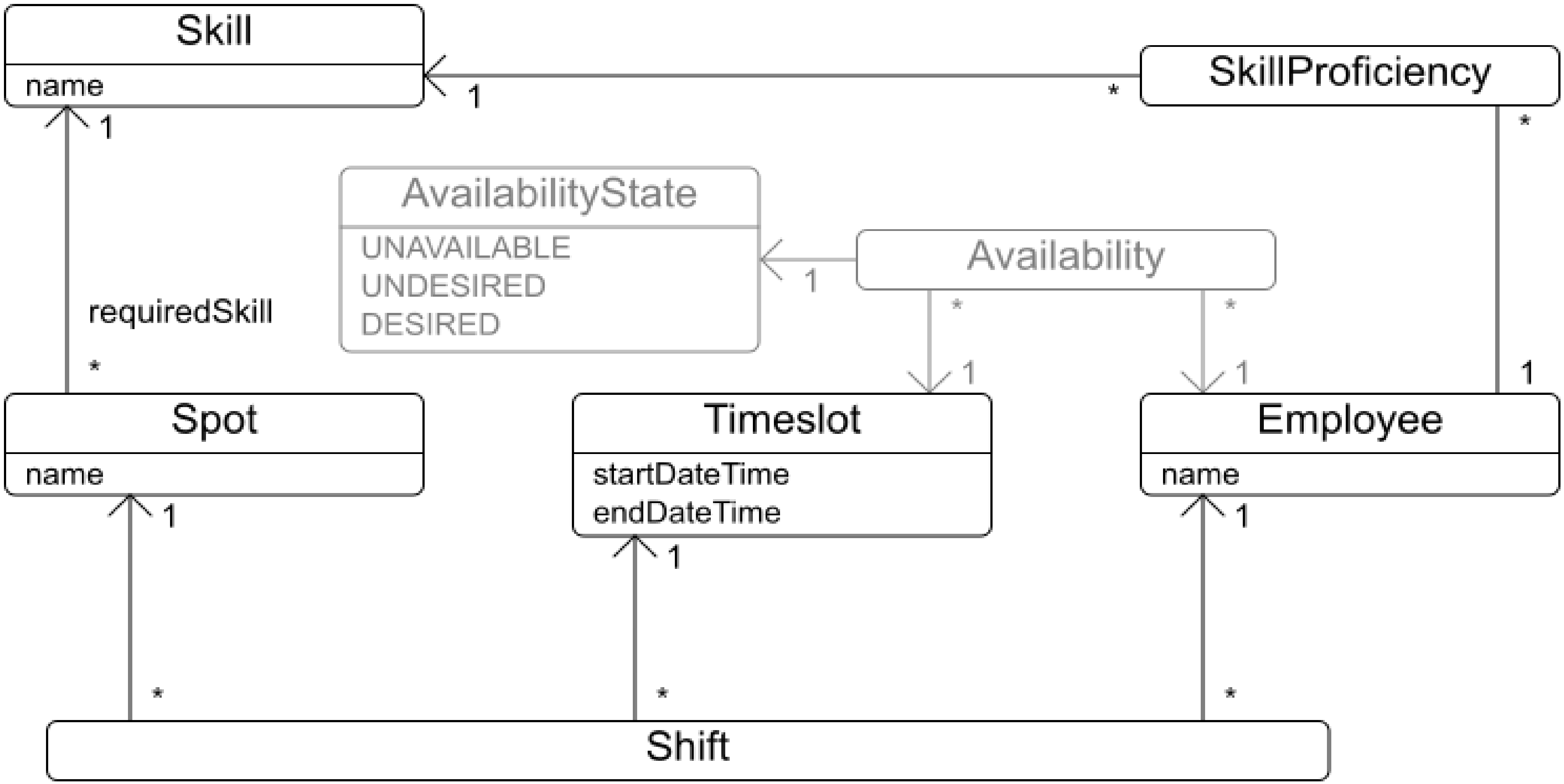
Ann
Bert
Carl

Good model

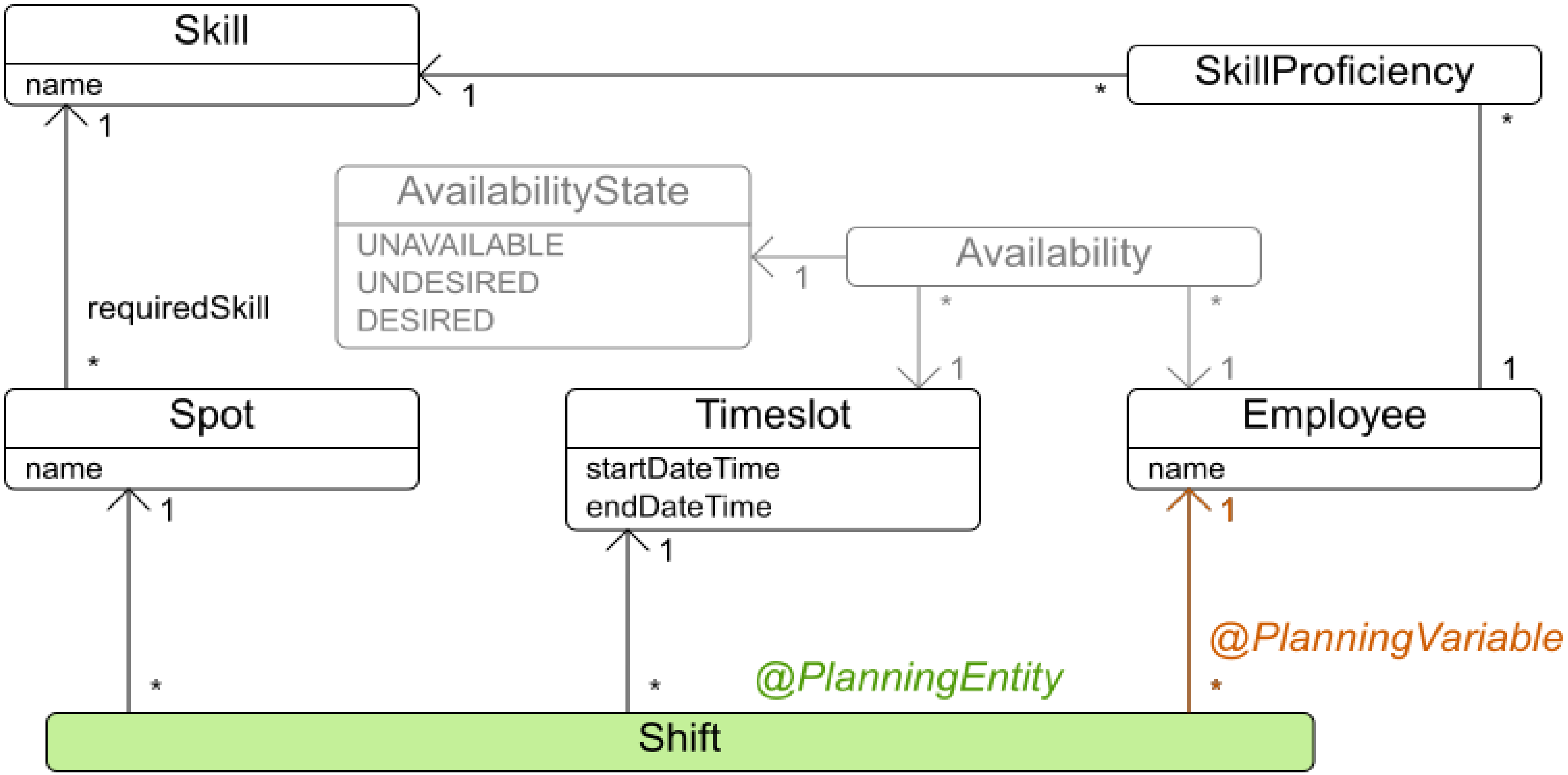
The number of employees
per shift
is known in advance:
it is part of the
requirements.

The PlanningSolution is easy to determine

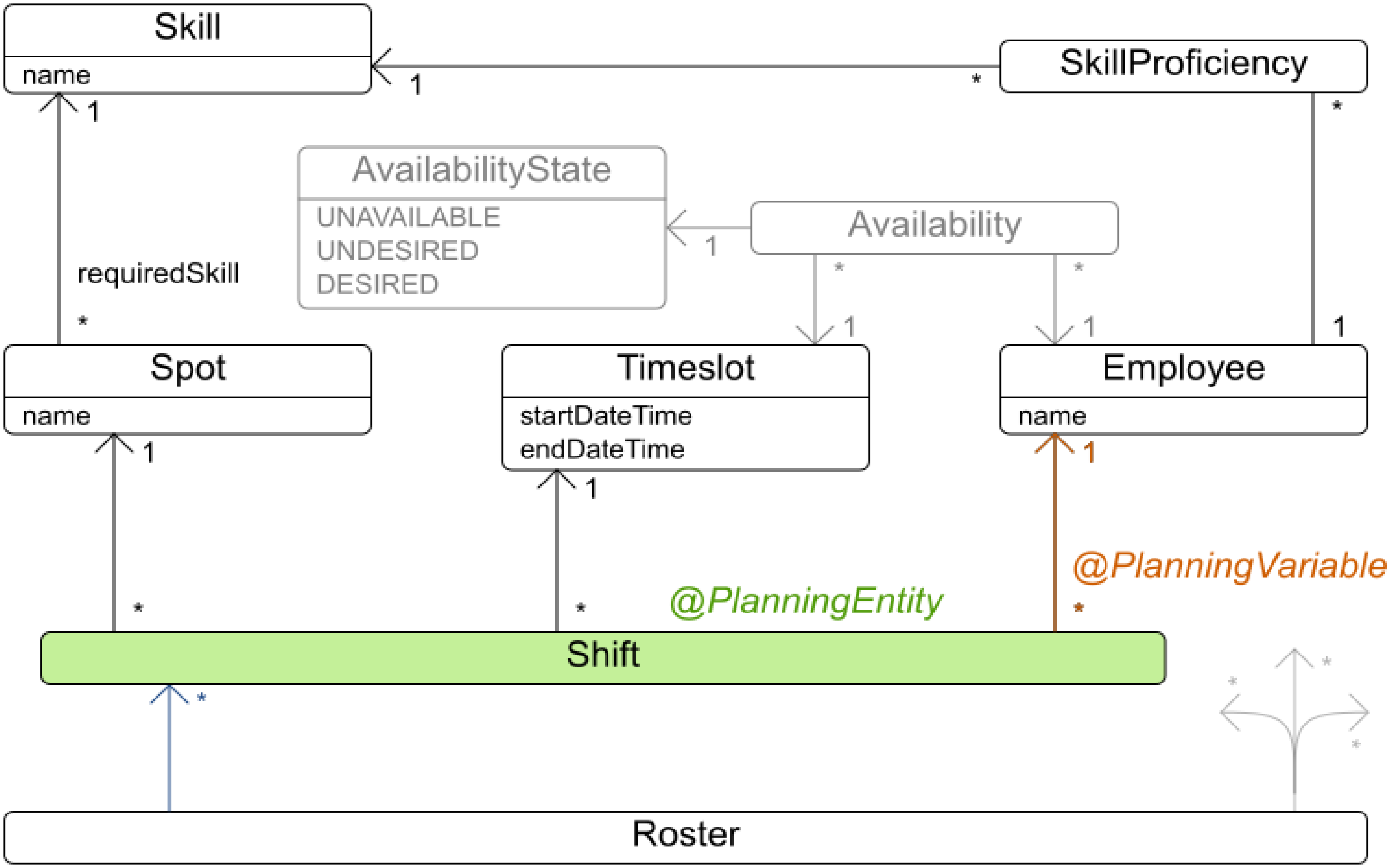
Employee rostering class diagram



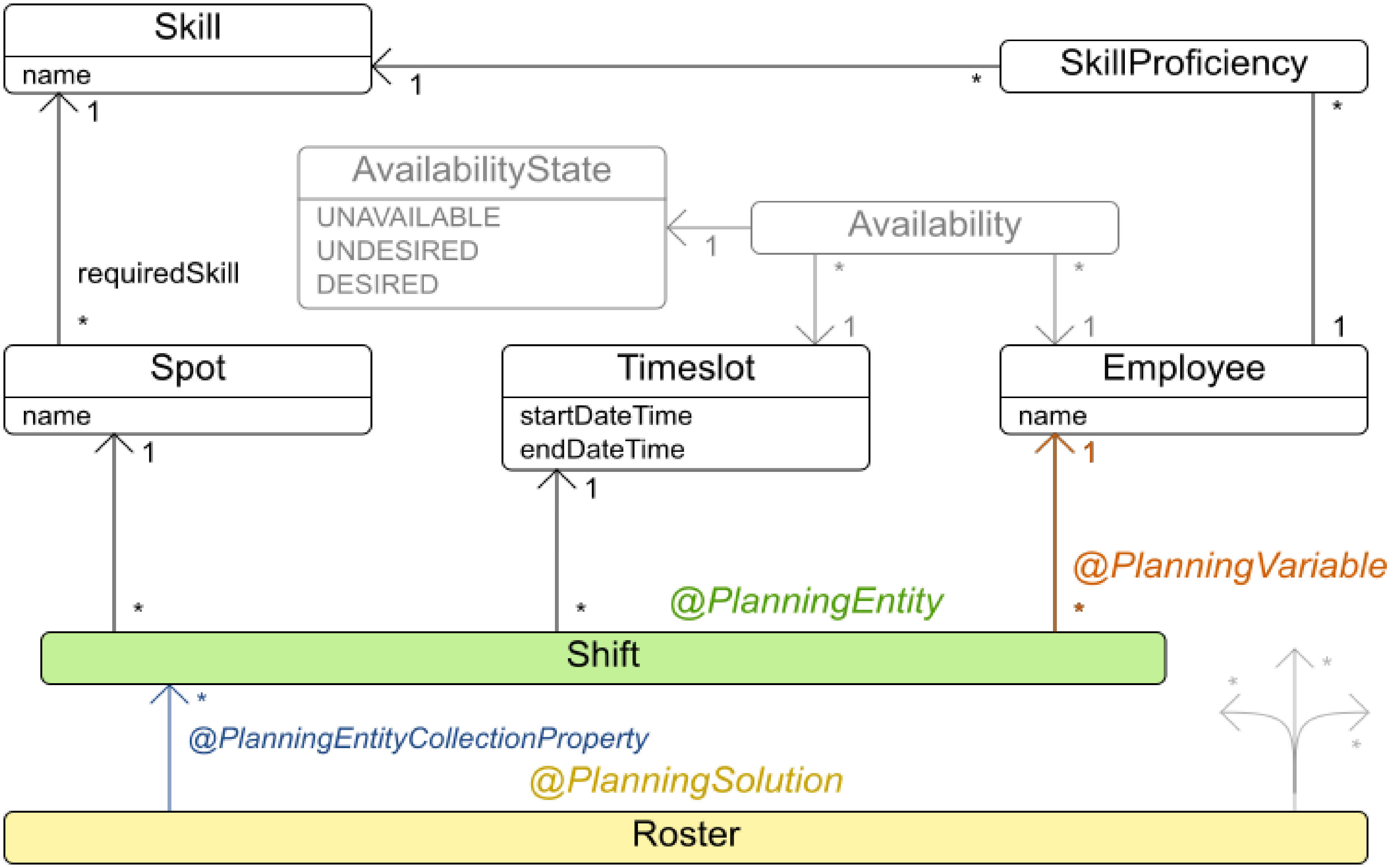
Employee rostering class diagram



Employee rostering class diagram

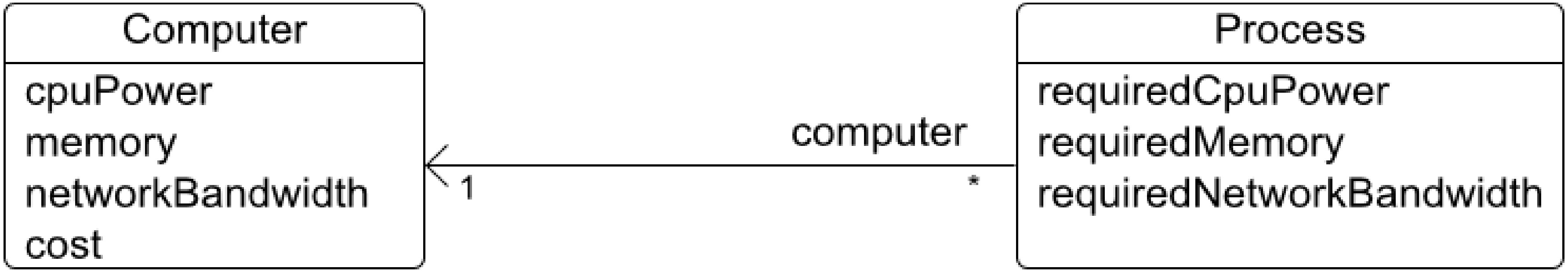


Employee rostering class diagram

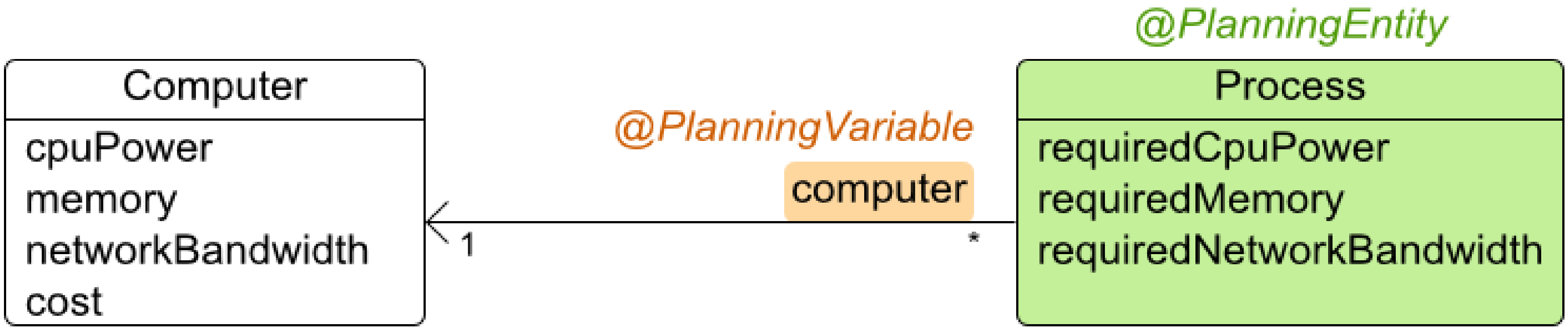


Exercises

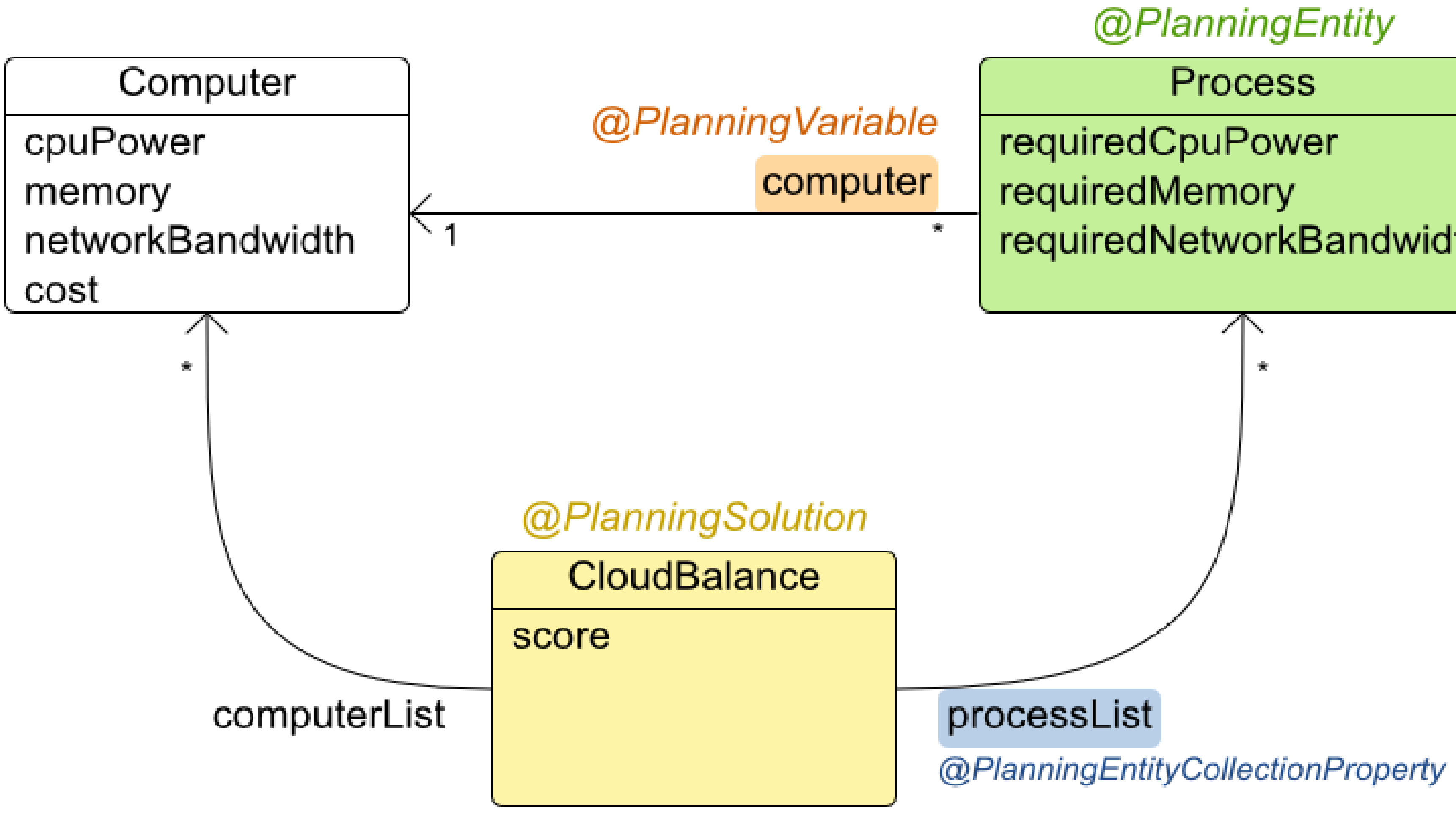
Cloud balance class diagram



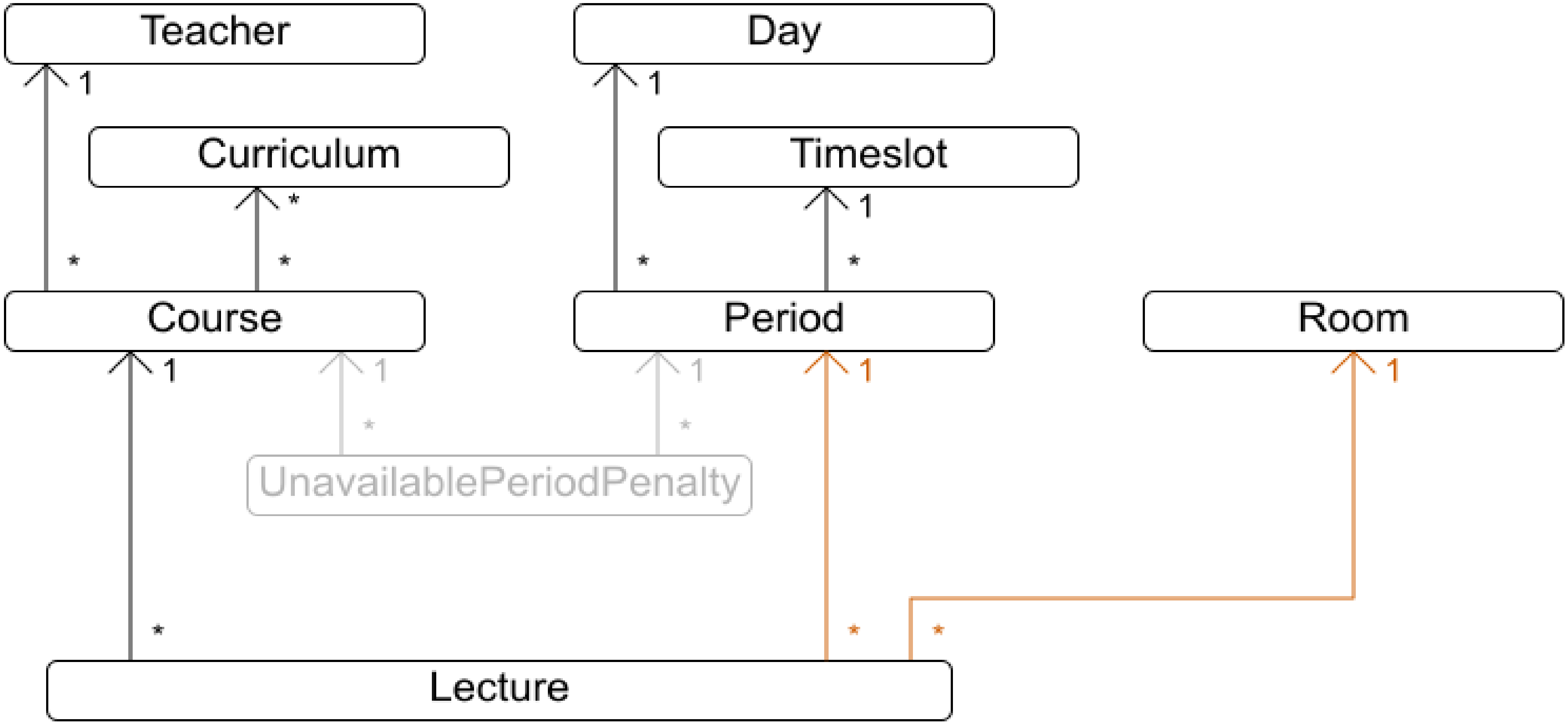
Cloud balance class diagram



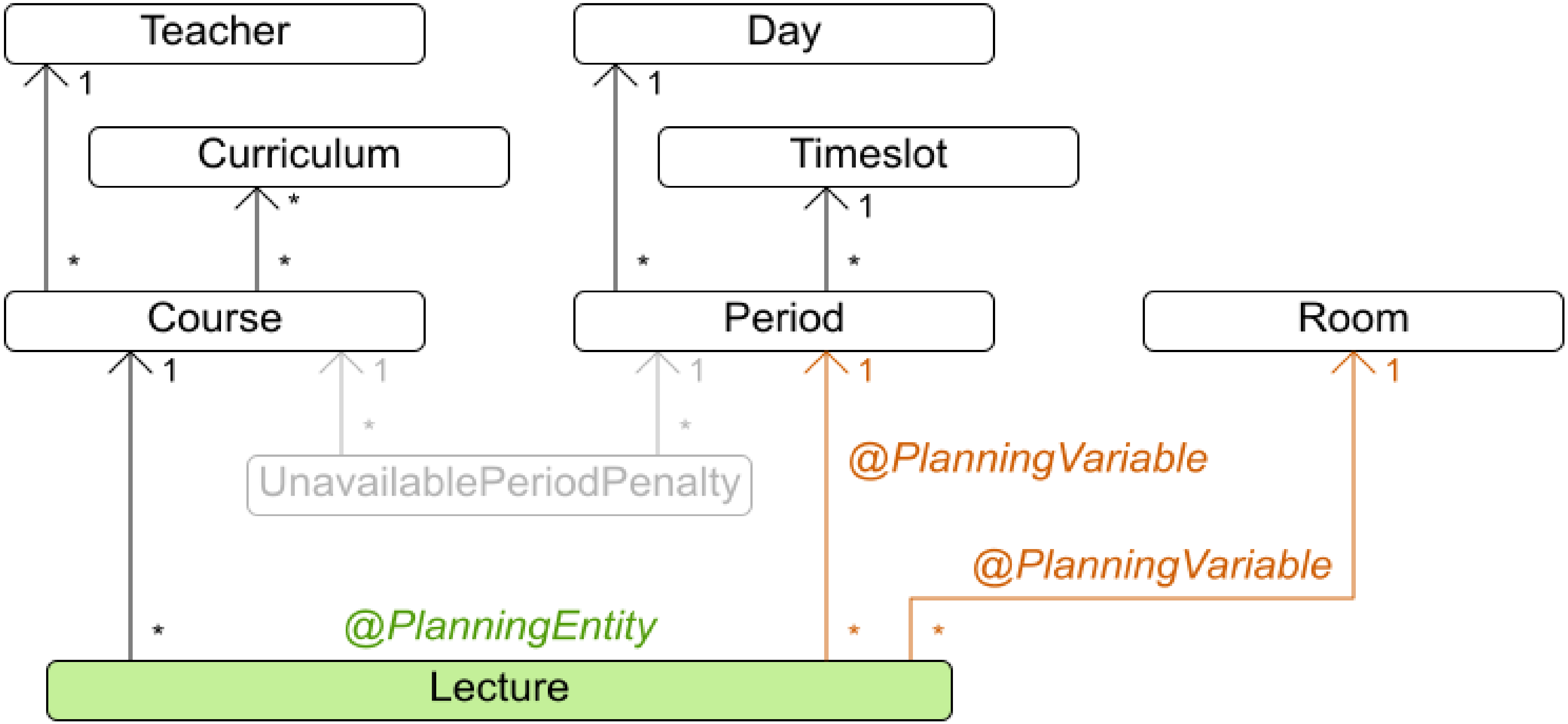
Cloud balance class diagram



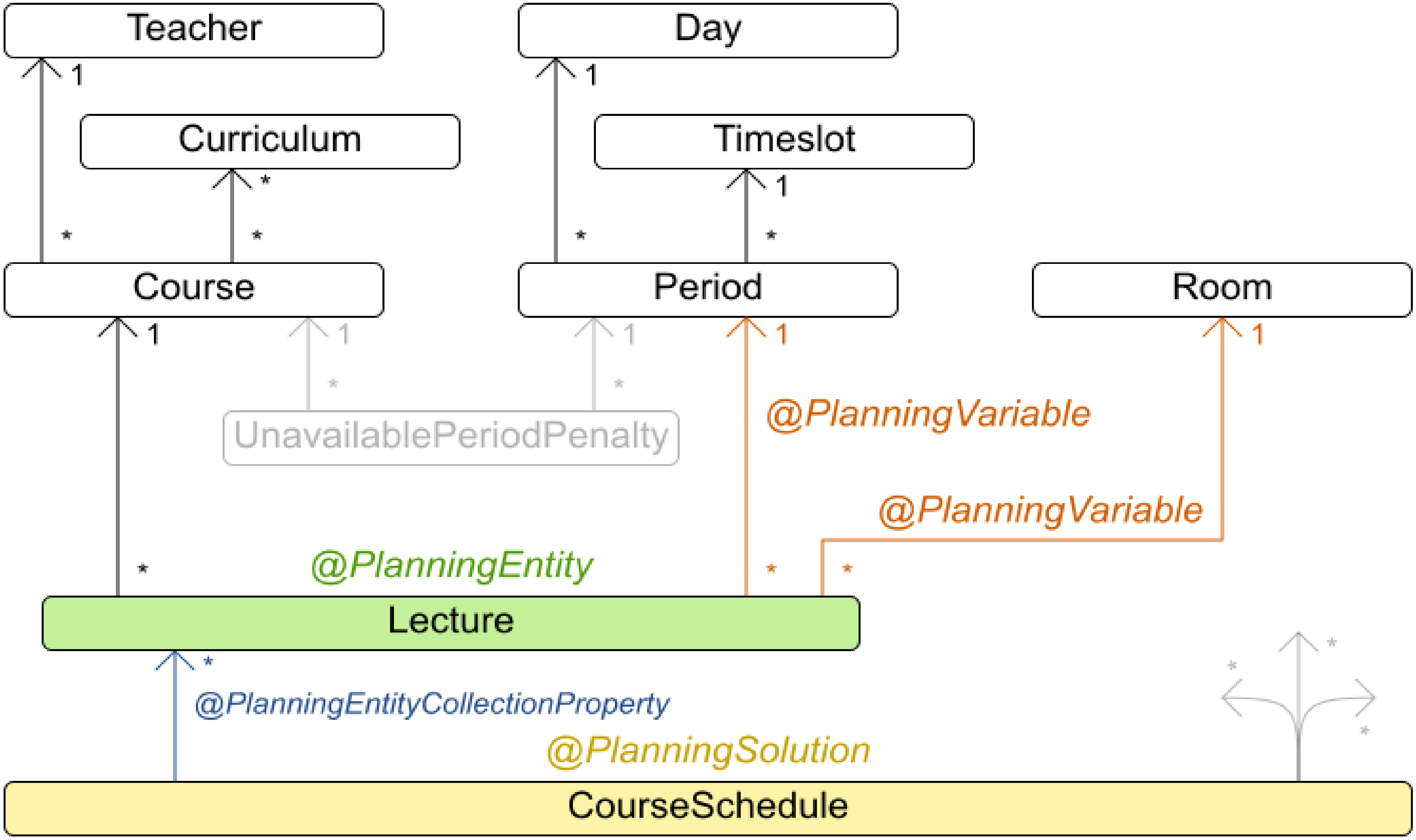
Curriculum course class diagram



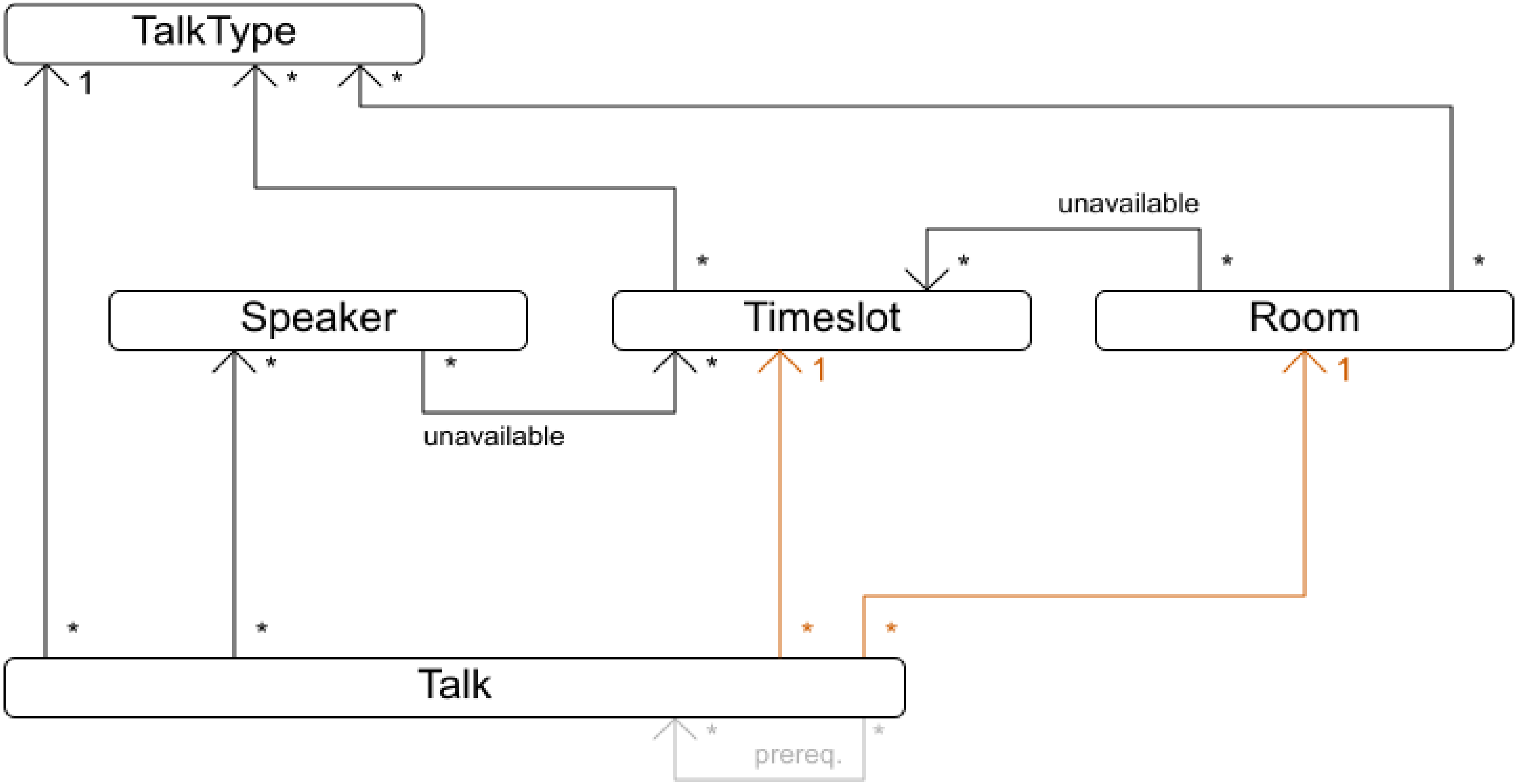
Curriculum course class diagram



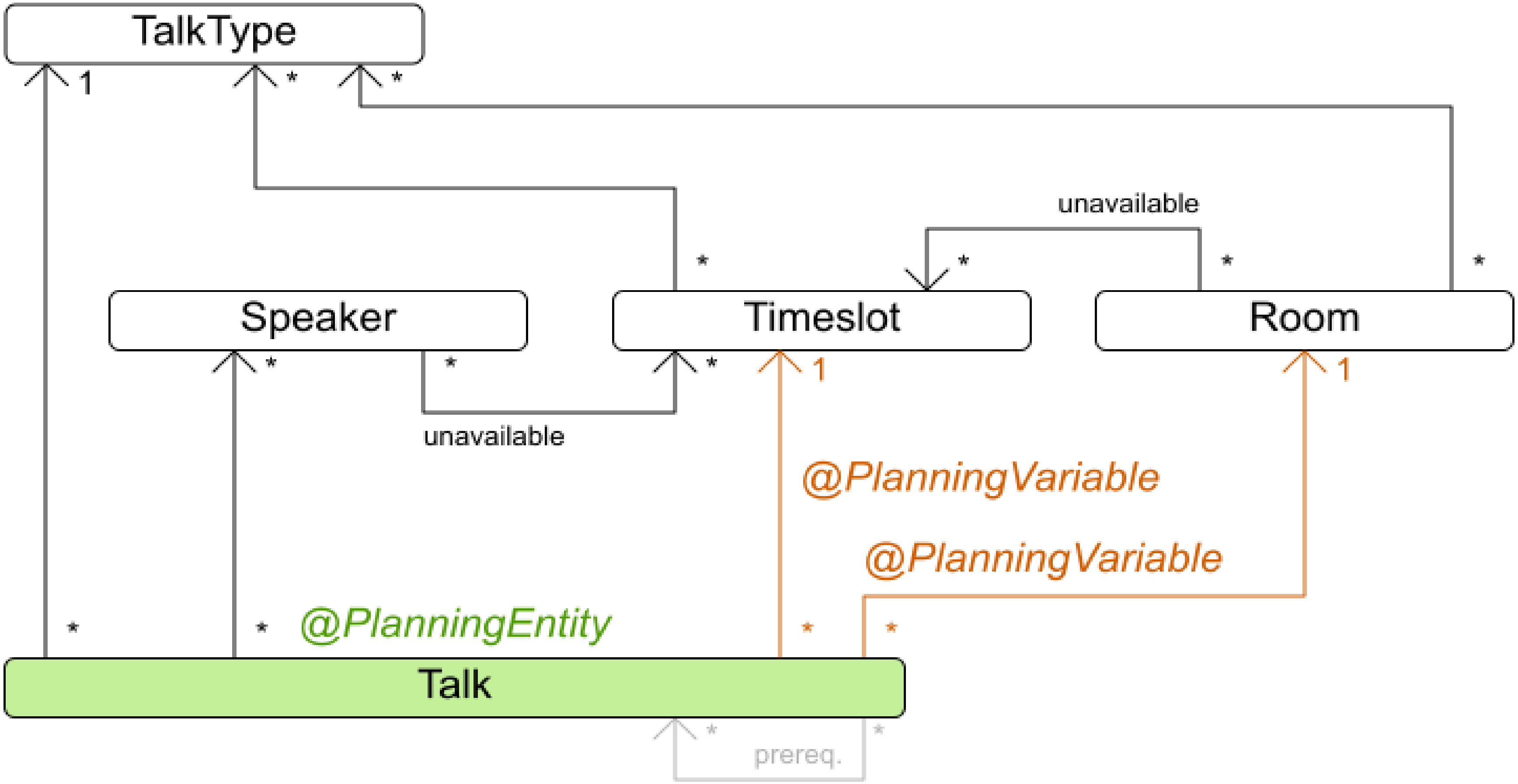
Curriculum course class diagram



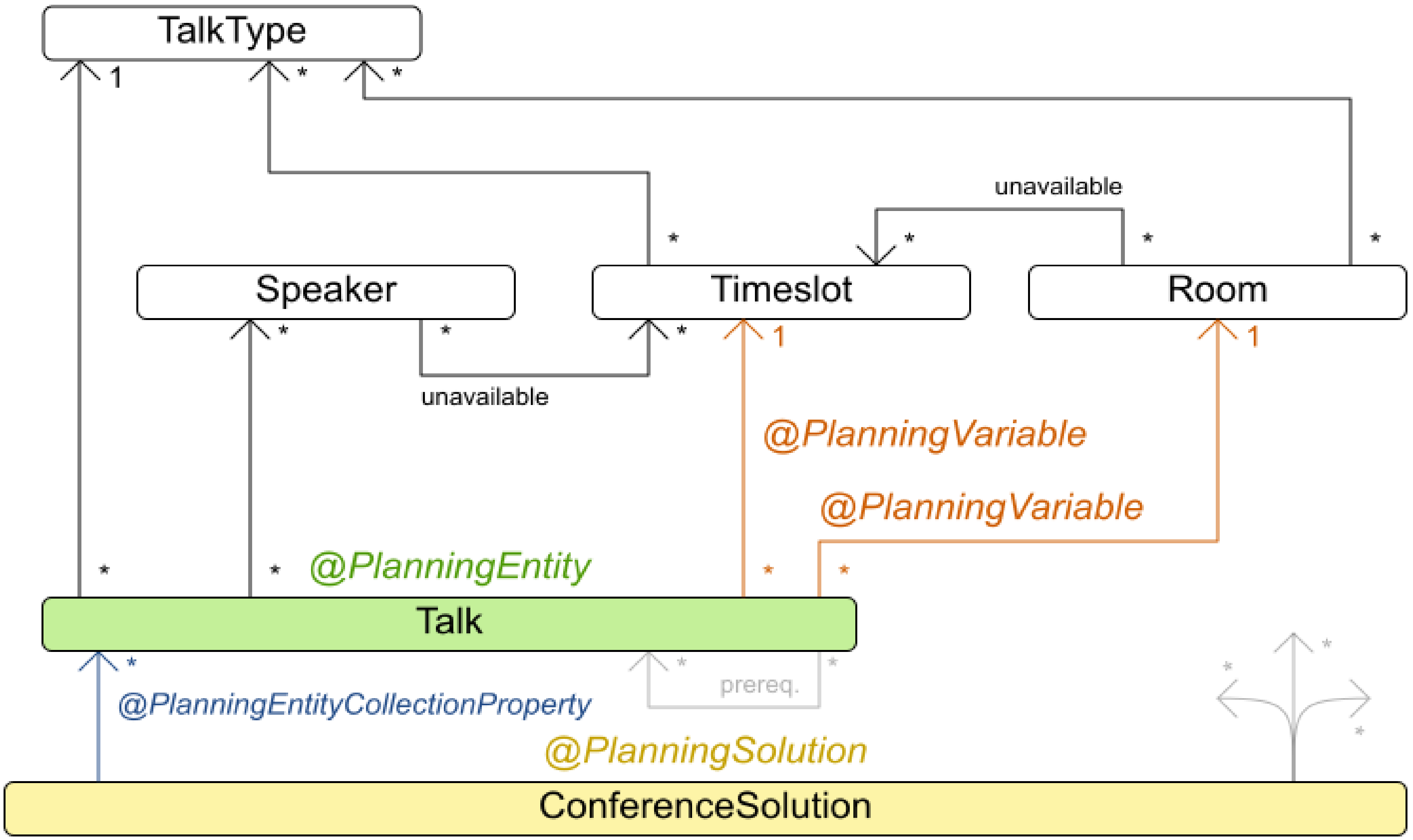
Conference scheduling class diagram



Conference scheduling class diagram



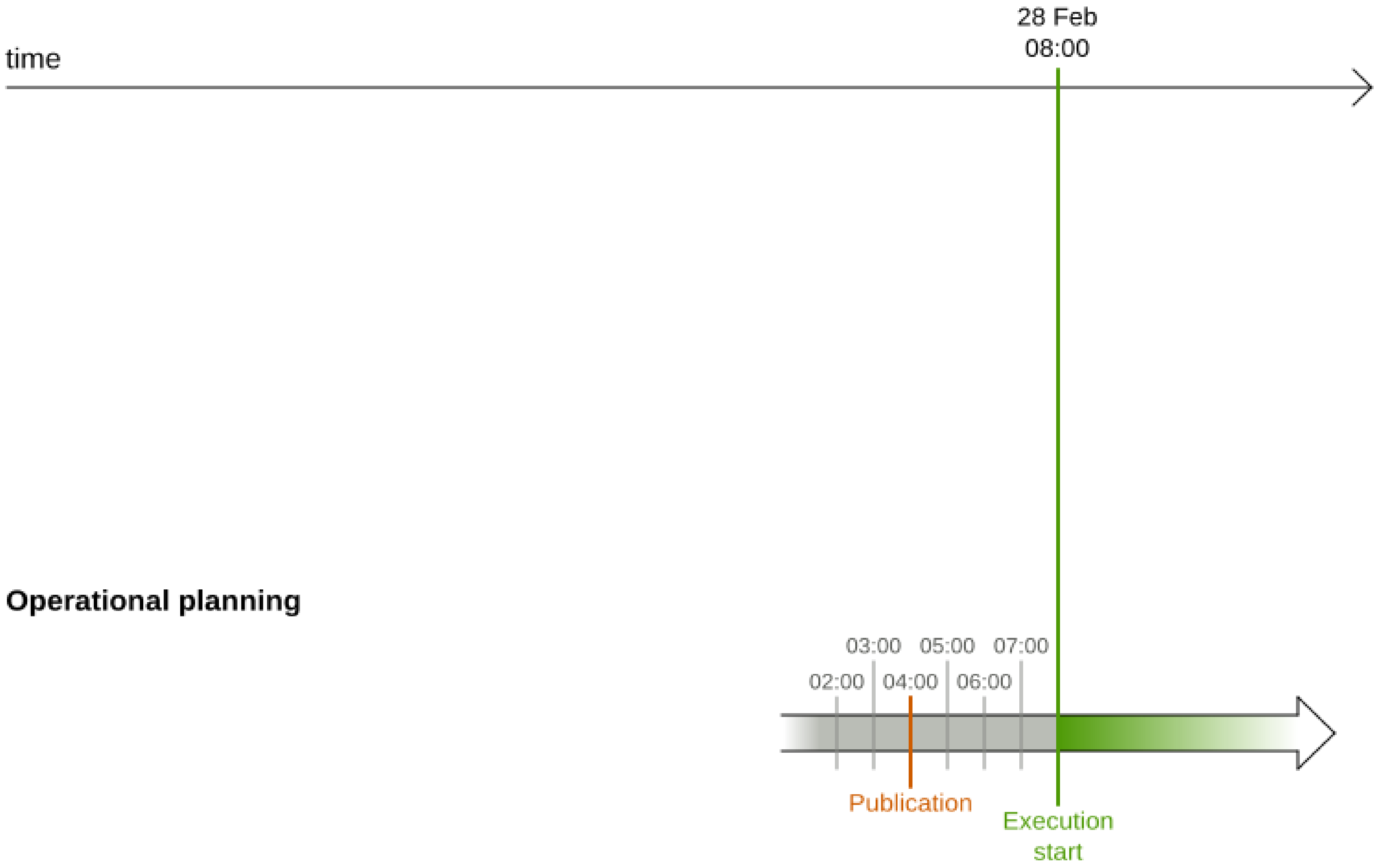
Conference scheduling class diagram



Assigning to time

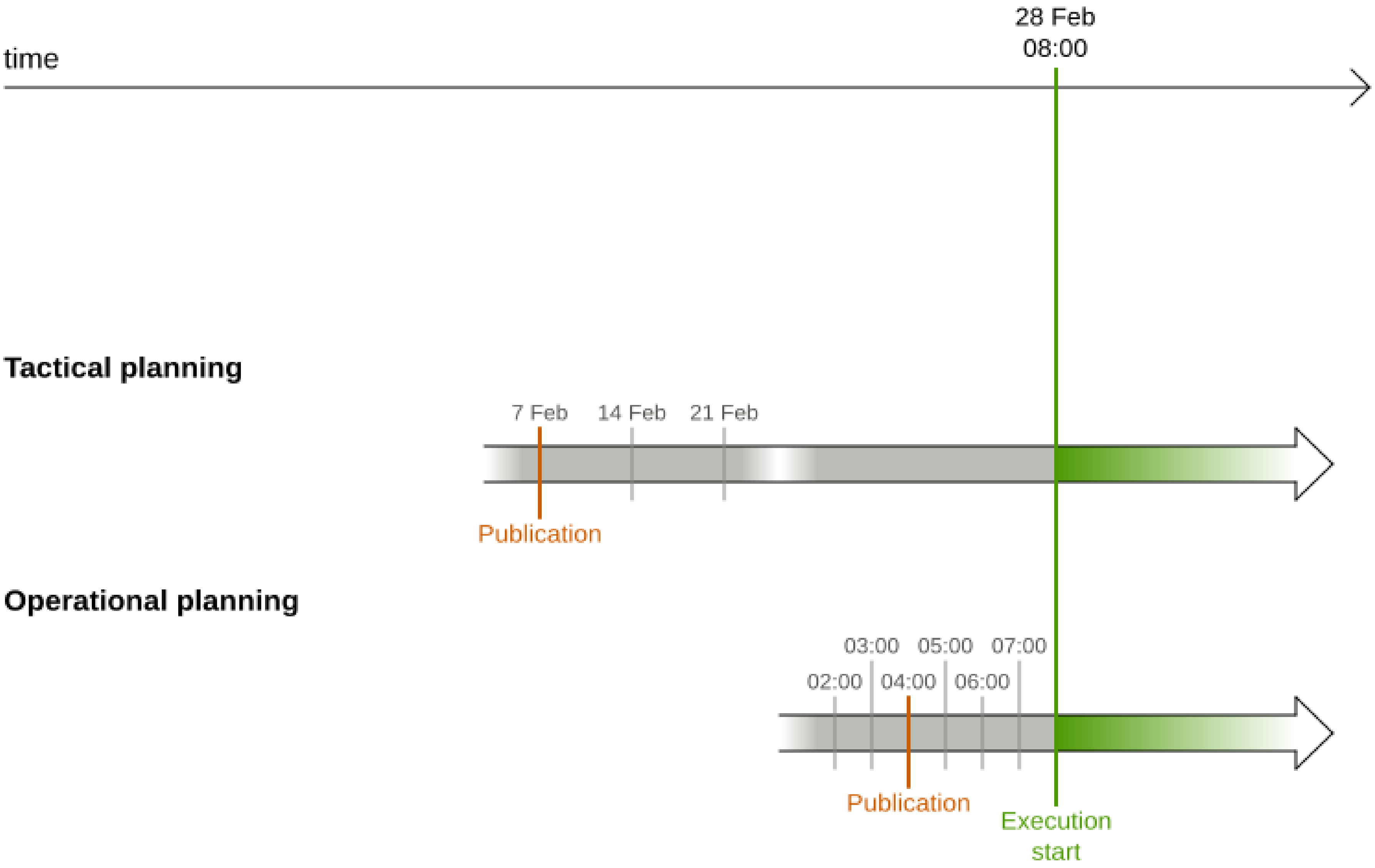
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



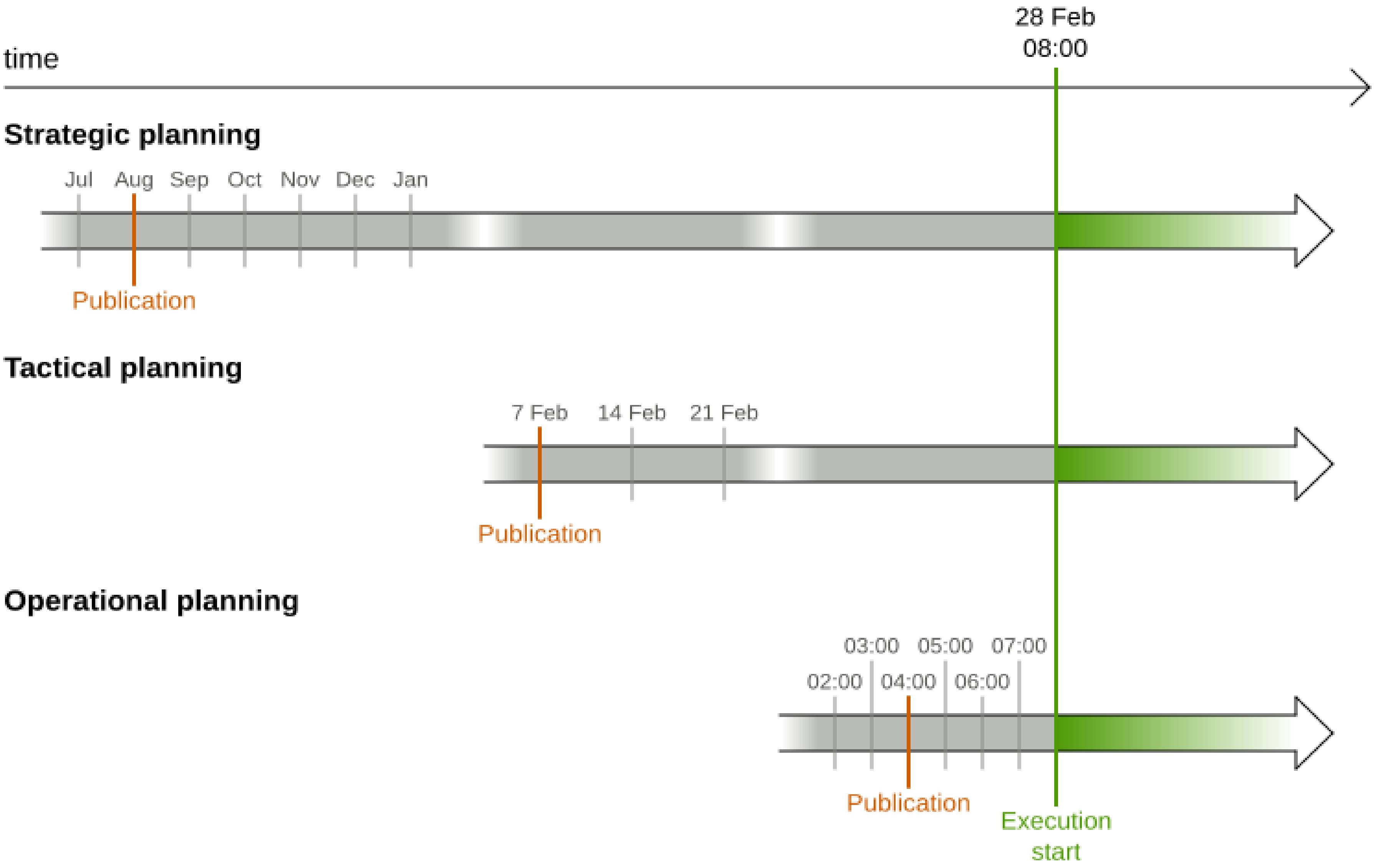
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



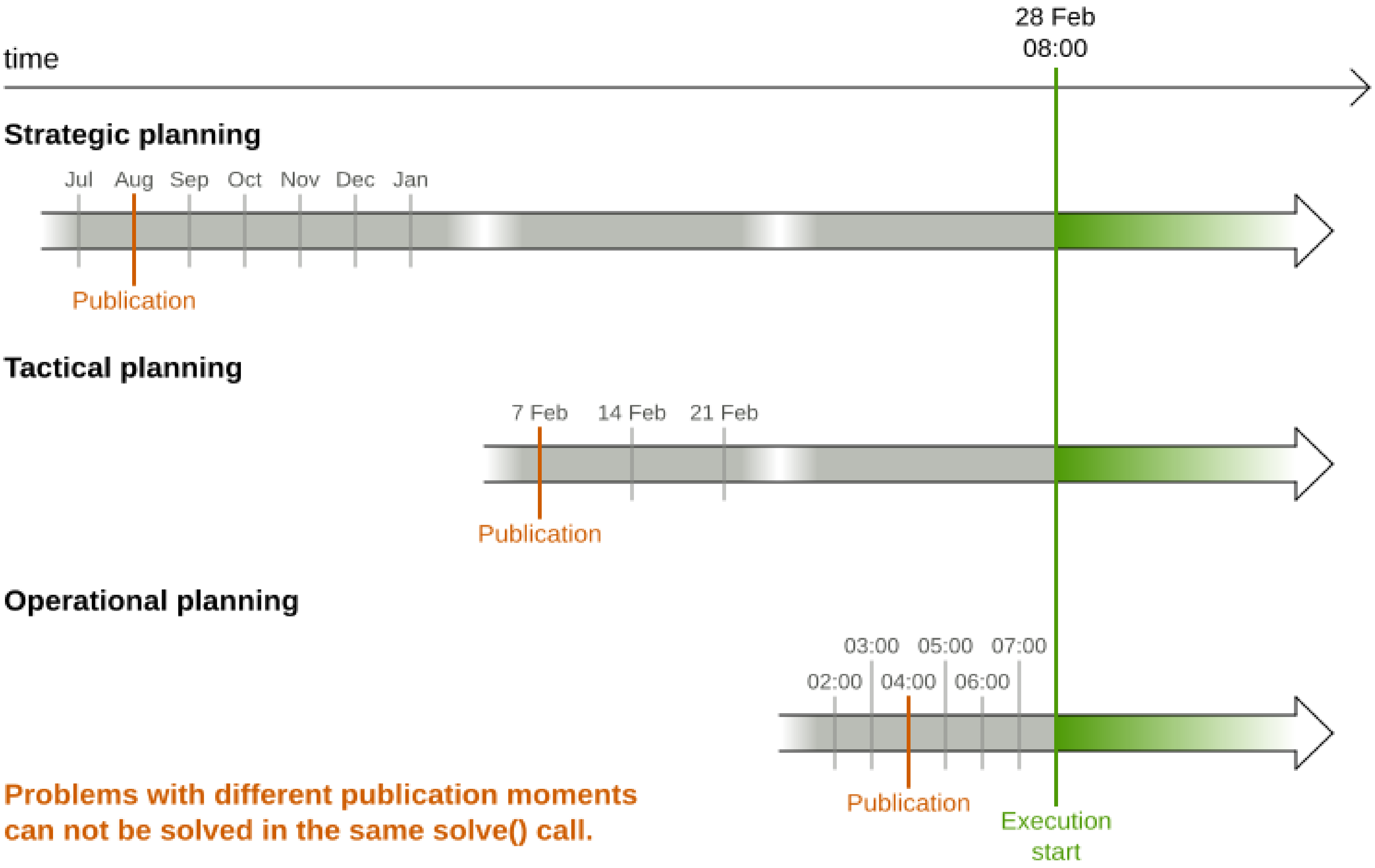
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



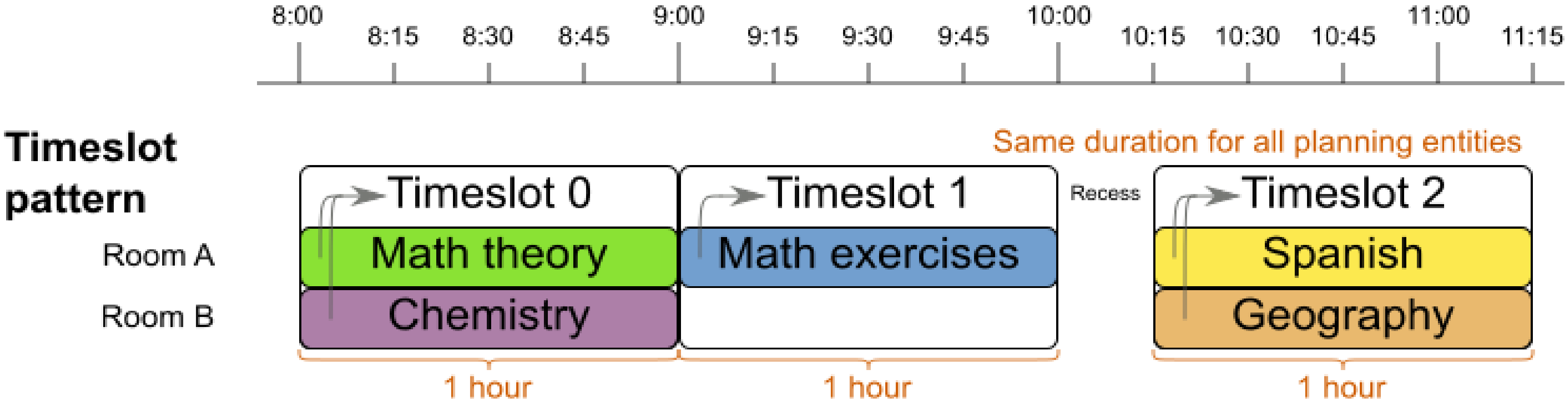
Problems with different publication moments can not be solved in the same solve() call.

Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.

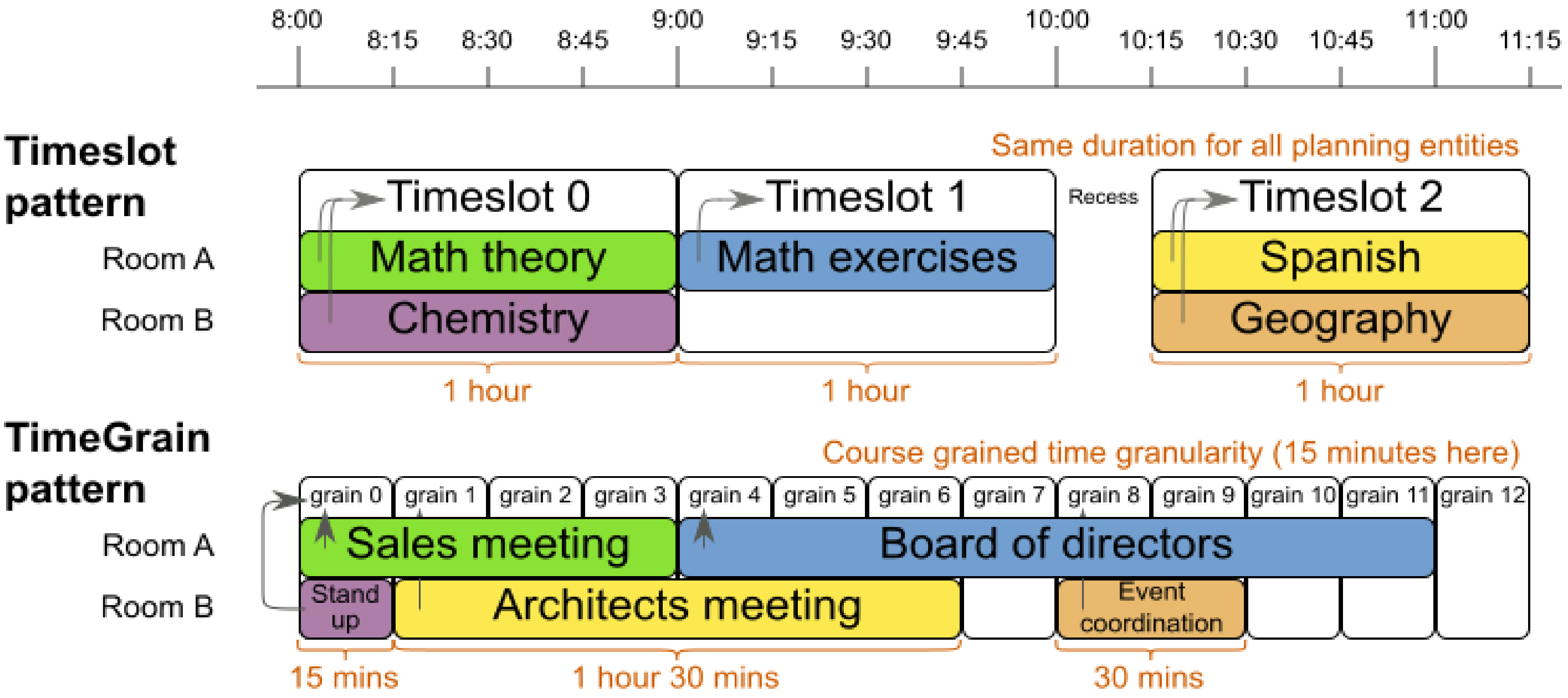
Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.



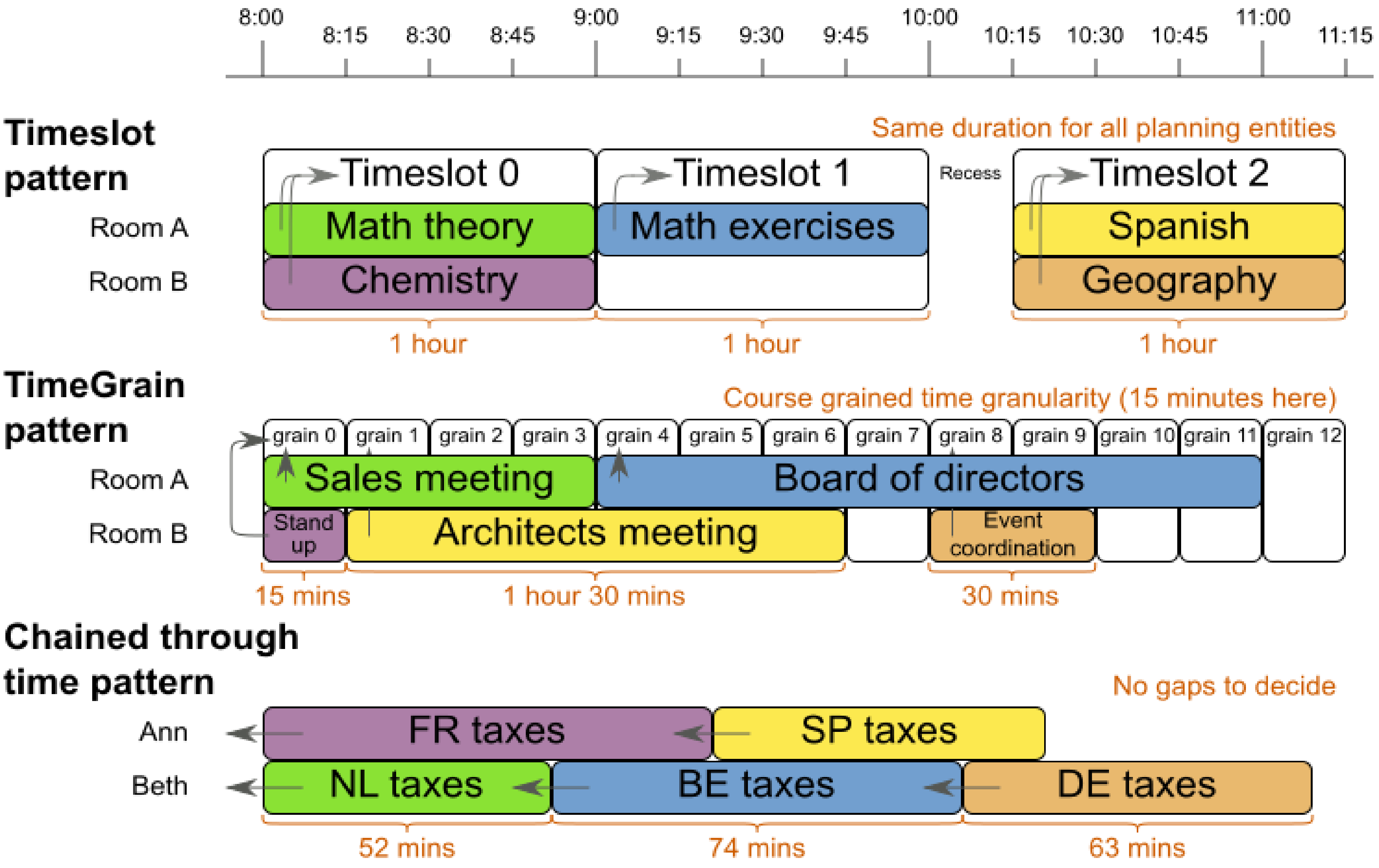
Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.



Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.



8:00 8:15 8:30 8:45 9:00 9:15 9:30 9:45 10:00 10:15 10:30 10:45 11:00 11:15

Timeslot pattern

Same duration for all planning entities

Timeslot 0
Timeslot 1
Recess
Timeslot 2

Room A
Math theory
Math exercises
Spanish

Room B
Chemistry
Geography

1 hour
1 hour
1 hour

TimeGrain pattern

Course grained time granularity (15 minutes here)

grain 0 grain 1 grain 2 grain 3 grain 4 grain 5 grain 6 grain 7 grain 8 grain 9 grain 10 grain 11 grain 12

Room A
Sales meeting
Board of directors

Room B
Stand up
Architects meeting
Event coordination

15 mins
1 hour 30 mins
30 mins

Chained through time pattern

No gaps to decide

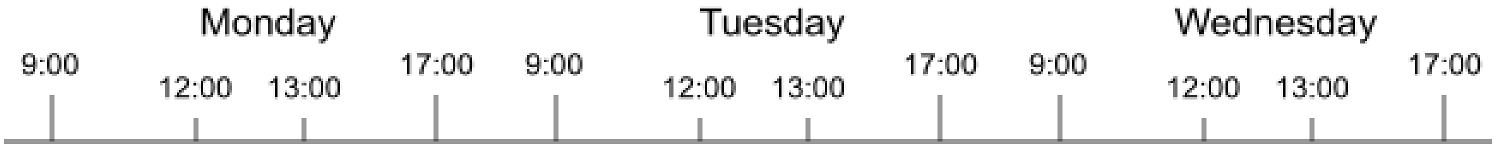
Ann
FR taxes
SP taxes

Beth
NL taxes
BE taxes
DE taxes

52 mins
74 mins
63 mins

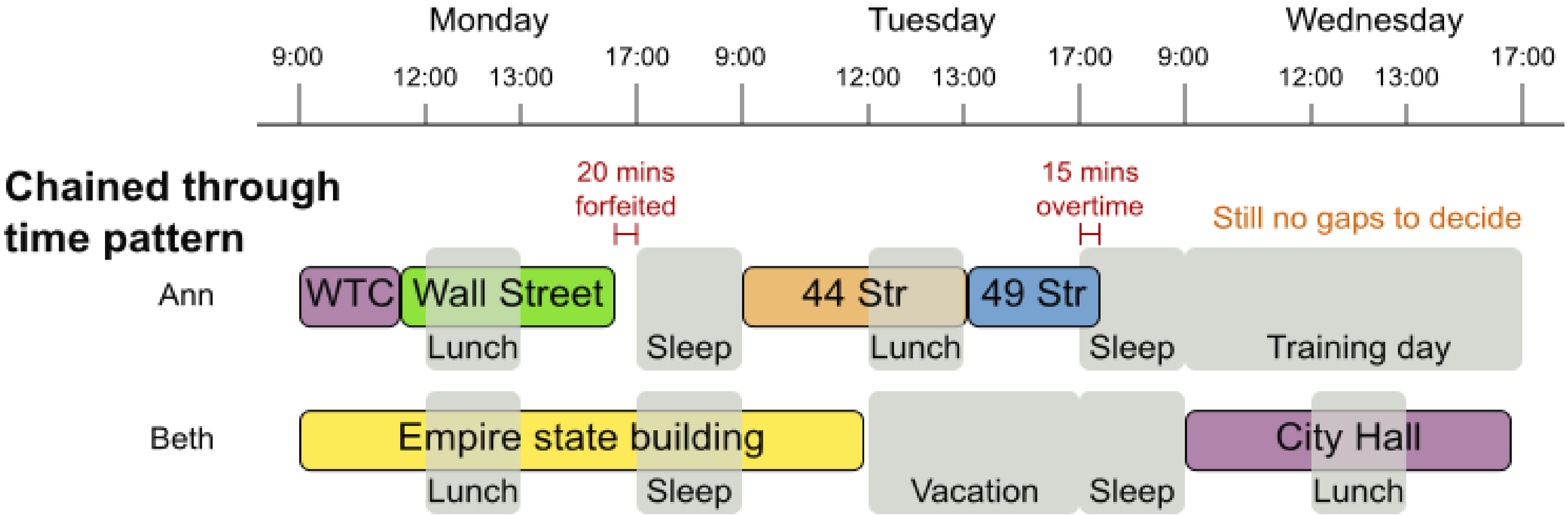
Assigning time to planning entities 2/2

There are several design patterns to deal with time, depending on your use case.



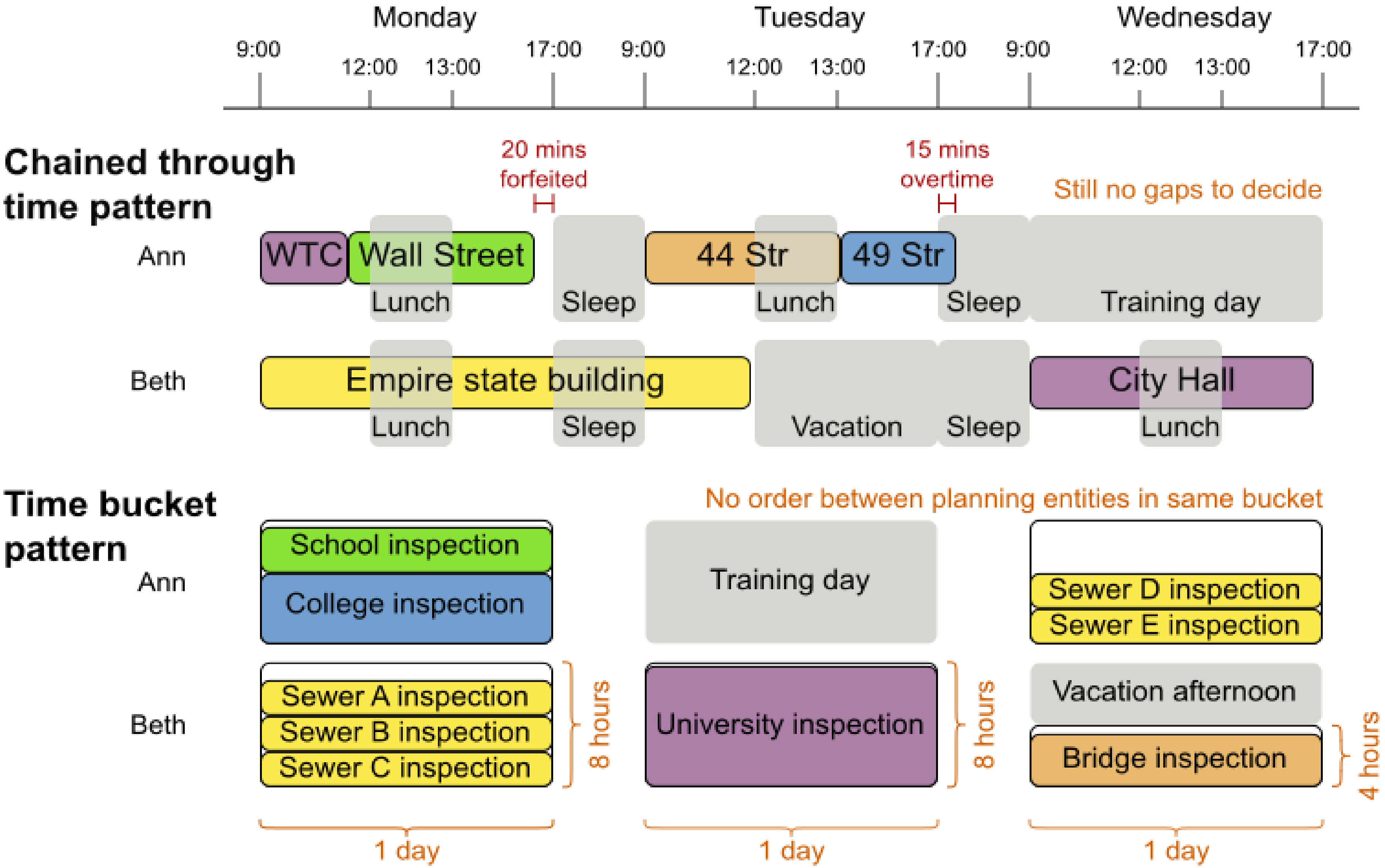
Assigning time to planning entities 2/2

There are several design patterns to deal with time, depending on your use case.



Assigning time to planning entities 2/2

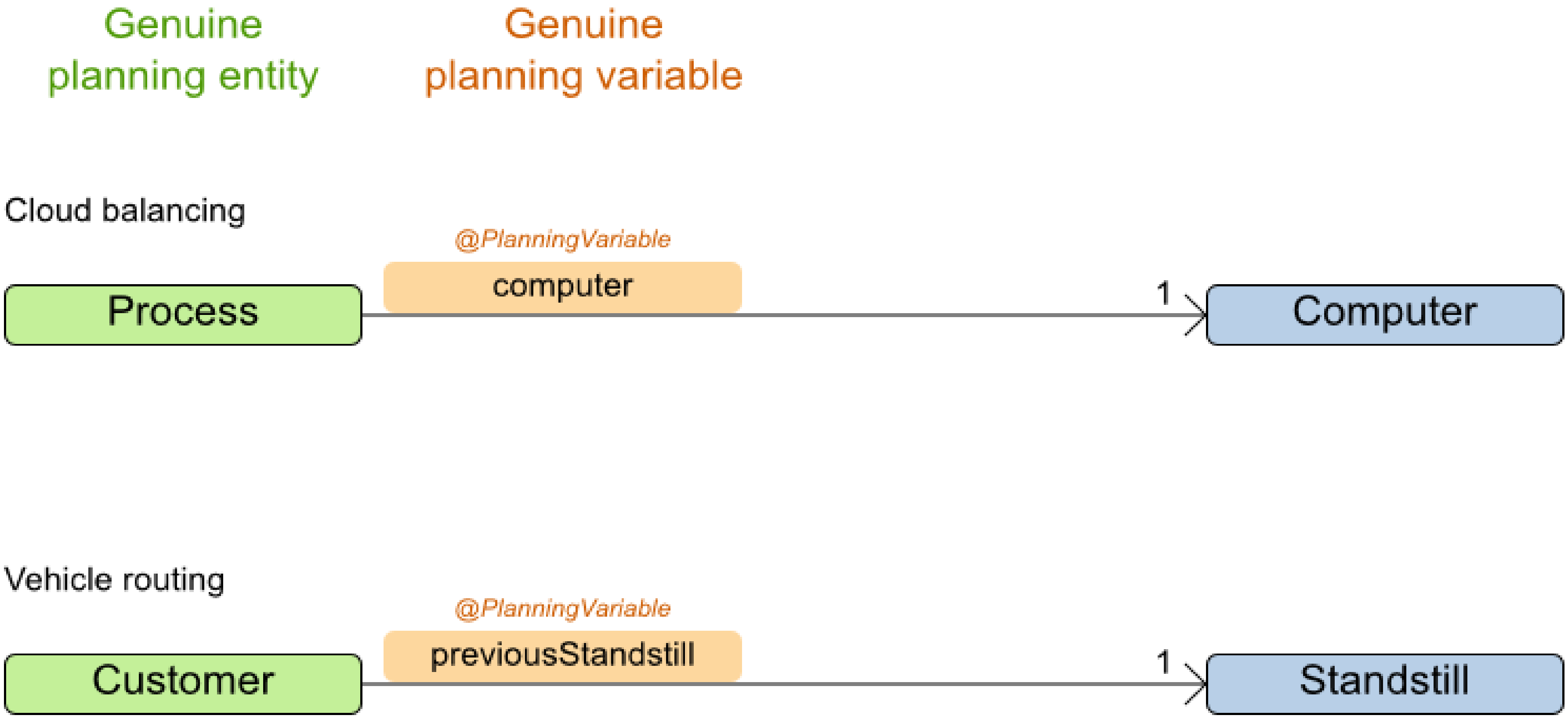
There are several design patterns to deal with time, depending on your use case.



Shadow variables

Bi-directional variable

One side of a bi-directional relationship is a genuine planning variable, the other side is a shadow variable.



Genuine planning entity

Genuine planning variable

Cloud balancing

@PlanningVariable

Process

computer

1

Computer

Vehicle routing

@PlanningVariable

Customer

previousStandstill

1

Standstill

Bi-directional variable

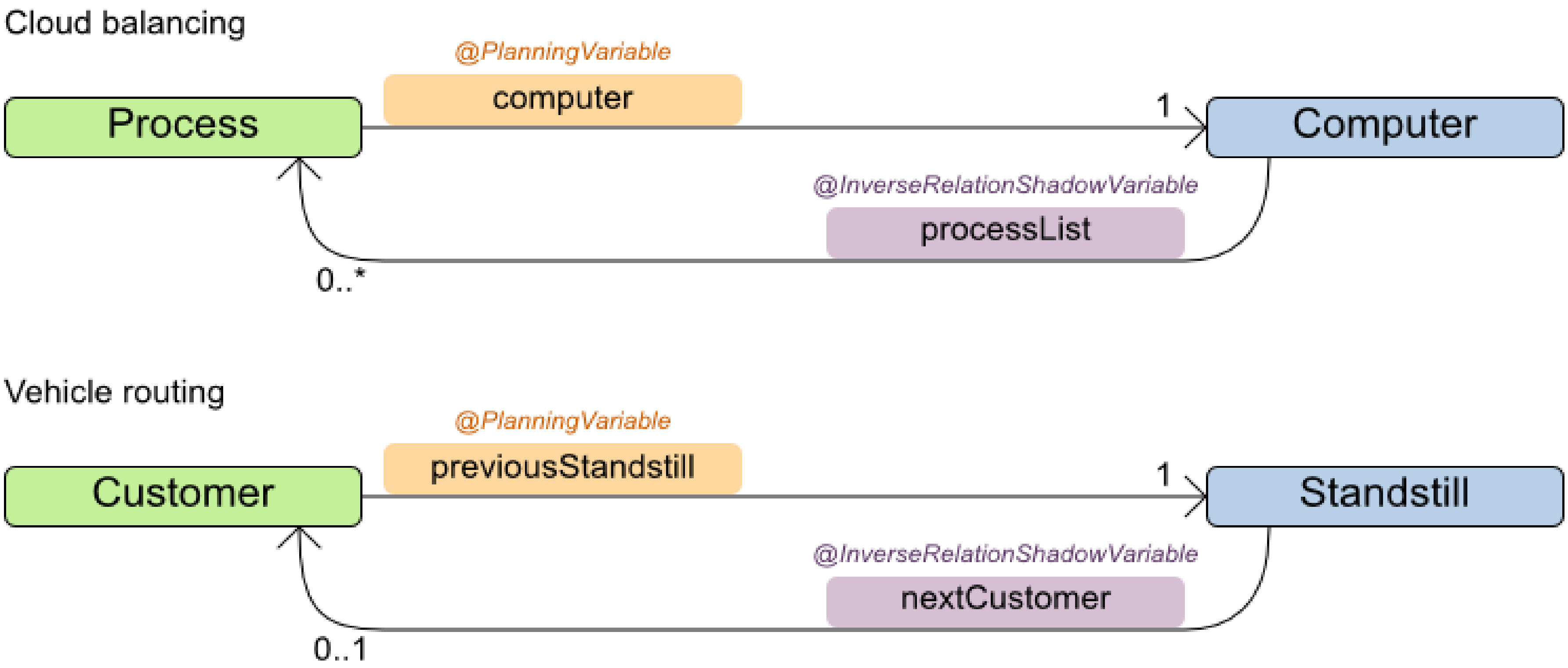
One side of a bi-directional relationship is a genuine planning variable, the other side is a shadow variable.

Genuine planning entity

Genuine planning variable

Shadow planning variable (inverse relation)

Shadow planning entity



Cloud balancing

@PlanningVariable

computer

Process

1

Computer

@InverseRelationShadowVariable

processList

0..*

Vehicle routing

@PlanningVariable

previousStandstill

Customer

1

Standstill

@InverseRelationShadowVariable

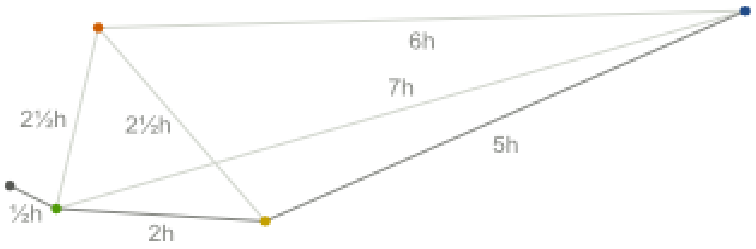
nextCustomer

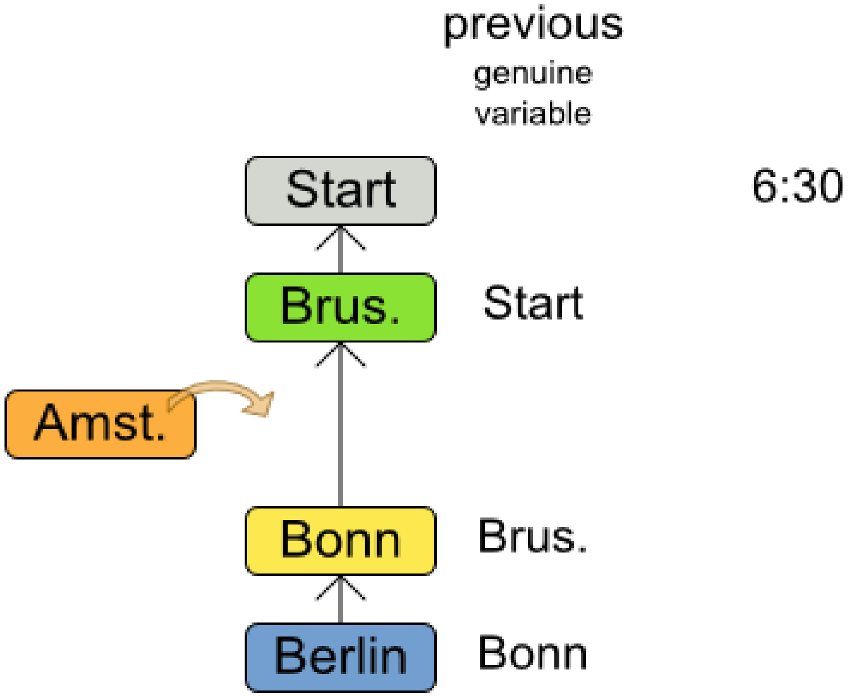
0..1

When the genuine planning variable changes,
then the inverse relationship variable changes accordingly.

Planning Variable Listener

When a Customer's assignment changes,
the arrival time of that customer (and of its trailing customers) change too.





previous
genuine
variable

Start 6:30

Brus. Start

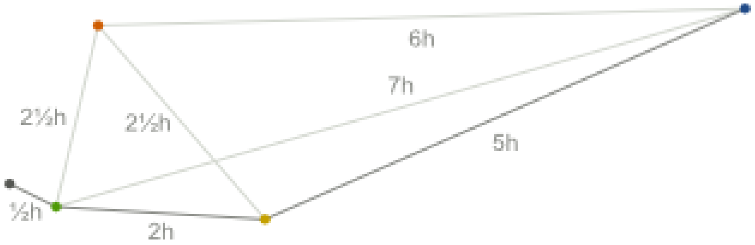
Amst.

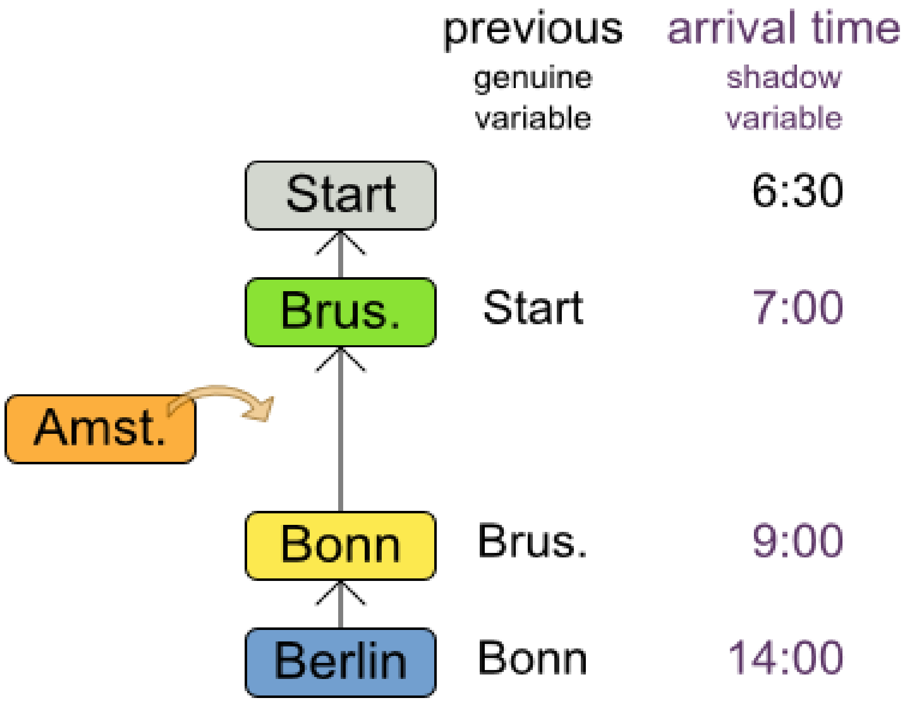
Bonn Brus.

Berlin Bonn

Planning Variable Listener

When a Customer's assignment changes,
the arrival time of that customer (and of its trailing customers) change too.

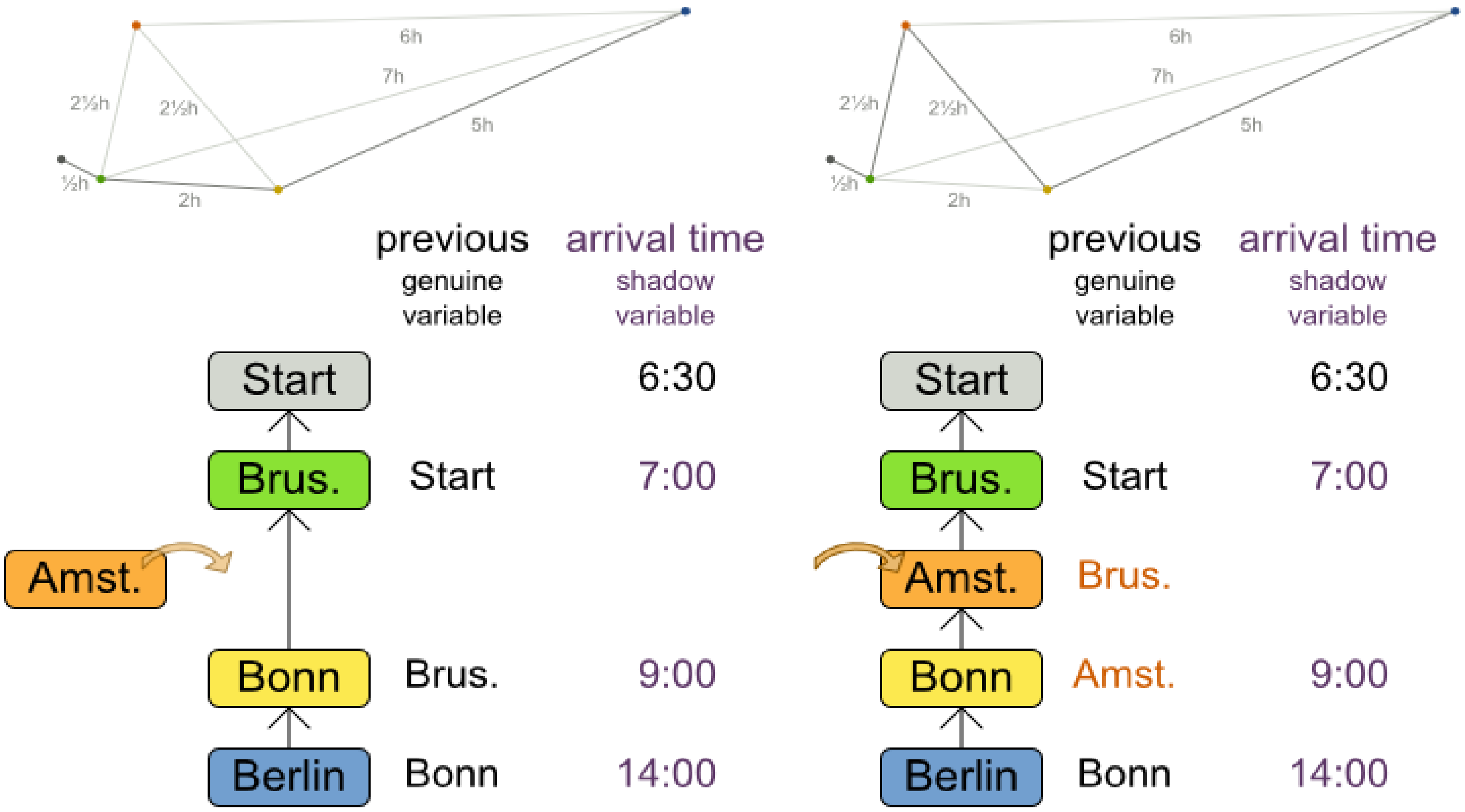




	previous genuine variable	arrival time shadow variable
Start		6:30
Brus.	Start	7:00
Amst.		
Bonn	Brus.	9:00
Berlin	Bonn	14:00

Planning Variable Listener

When a Customer's assignment changes,
the arrival time of that customer (and of its trailing customers) change too.

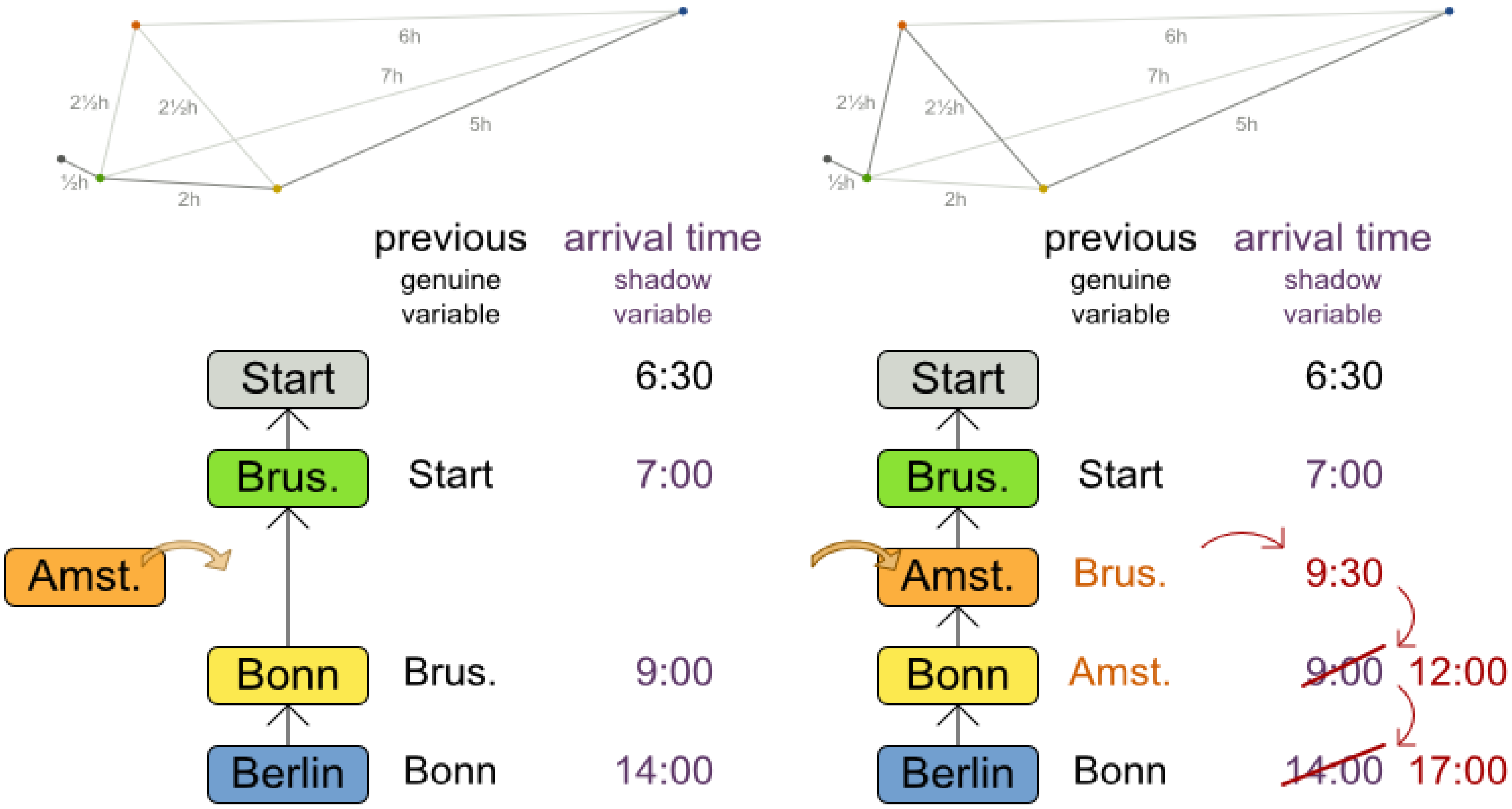


	previous genuine variable	arrival time shadow variable
Start		6:30
Brus.	Start	7:00
Amst.		
Bonn	Brus.	9:00
Berlin	Bonn	14:00

	previous genuine variable	arrival time shadow variable
Start		6:30
Brus.	Start	7:00
Amst.	Brus.	
Bonn	Amst.	9:00
Berlin	Bonn	14:00

Planning Variable Listener

When a Customer's assignment changes,
the arrival time of that customer (and of its trailing customers) change too.



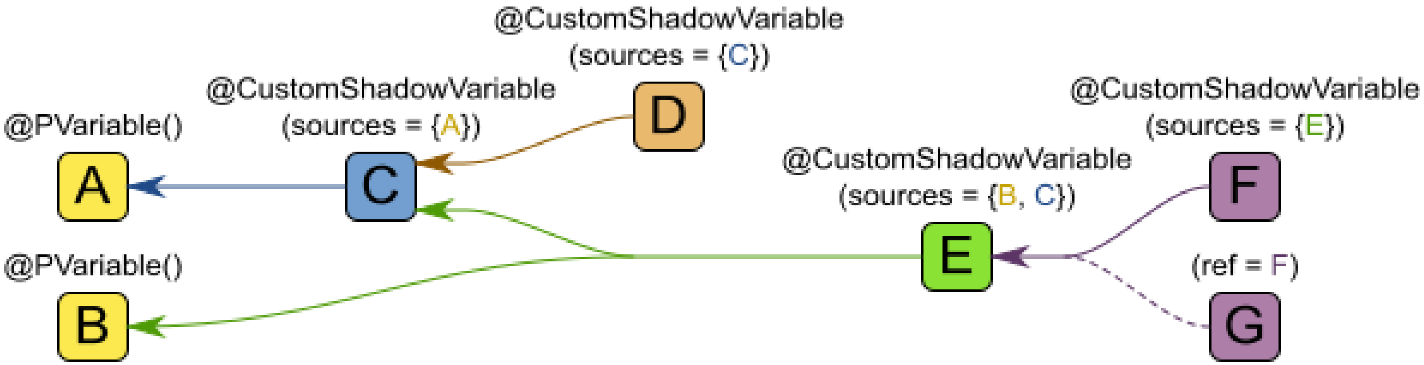
	previous genuine variable	arrival time shadow variable
Start		6:30
Brus.	Start	7:00
Bonn	Brus.	9:00
Berlin	Bonn	14:00

	previous genuine variable	arrival time shadow variable
Start		6:30
Brus.	Start	7:00
Amst.	Brus.	9:30
Bonn	Amst.	~~9:00~~ 12:00
Berlin	Bonn	~~14:00~~ 17:00

When a genuine planning variable changes,
then the Listener(s) change the shadow variable(s) accordingly.

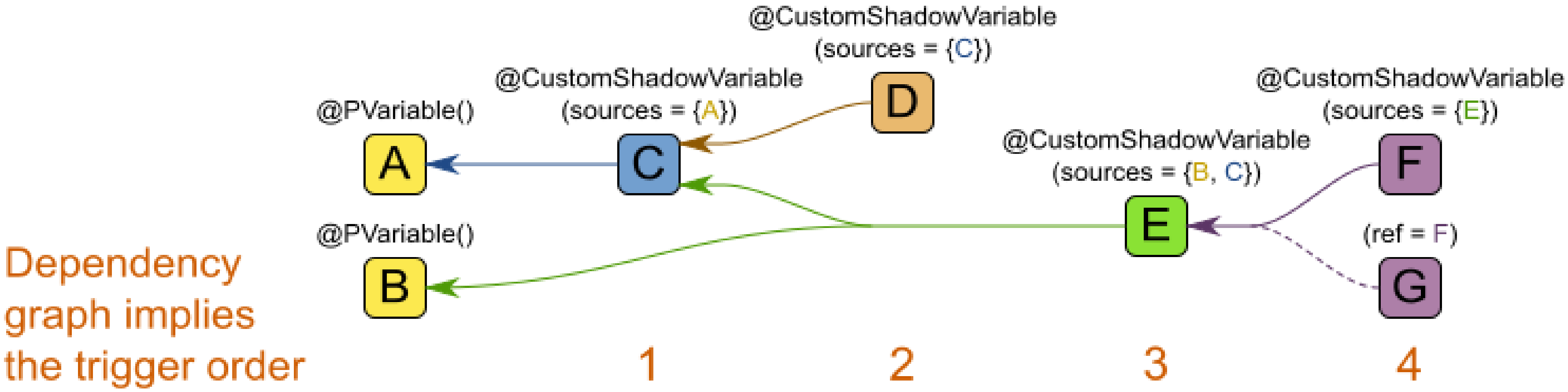
Shadow variable order

The shadow variable dependencies determine the order in which their after*() methods are called.



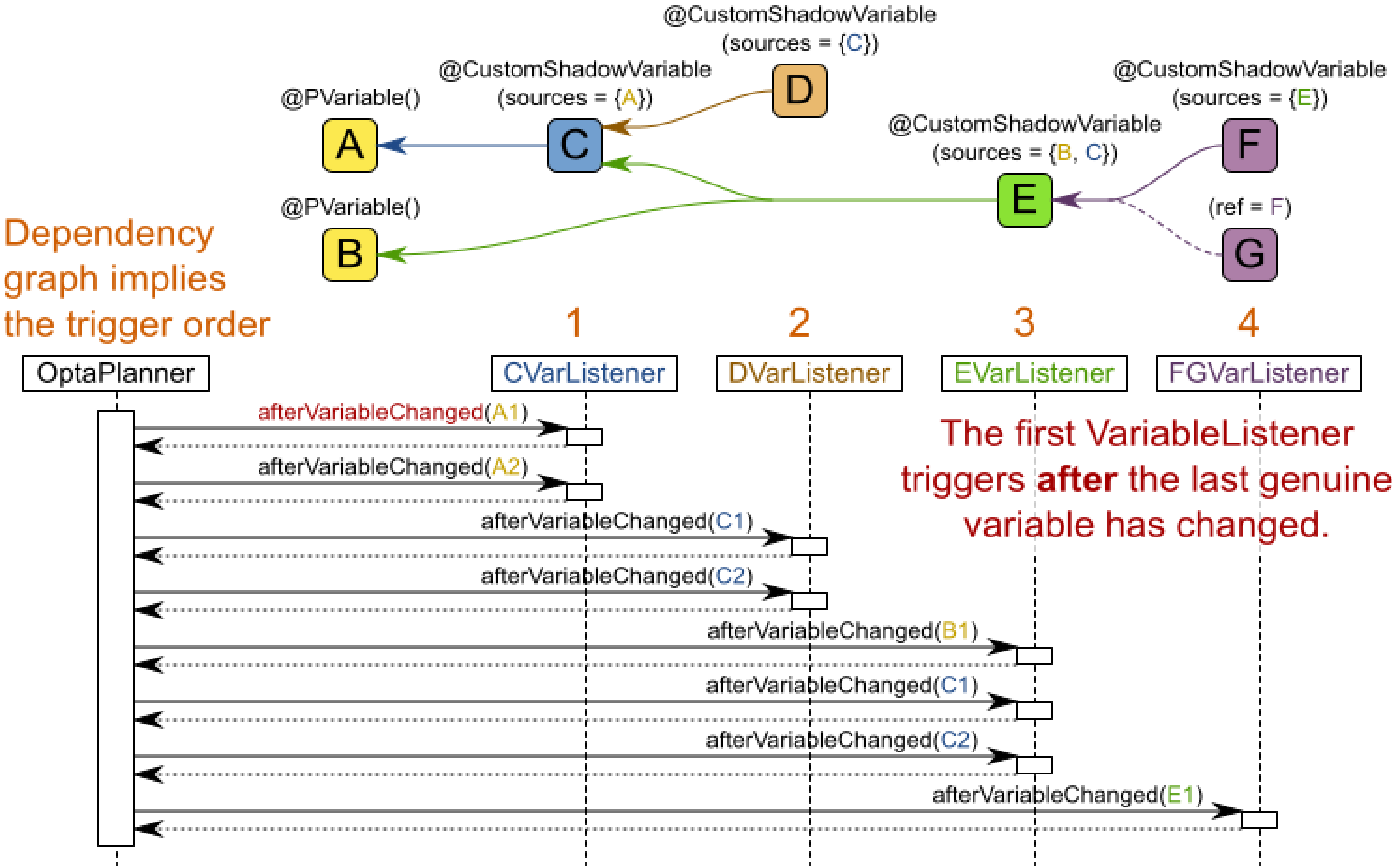
Shadow variable order

The shadow variable dependencies determine the order in which their after*() methods are called.



Shadow variable order

The shadow variable dependencies determine the order in which their after*() methods are called.



@CustomShadowVariable
(sources = {C})

D

@CustomShadowVariable
(sources = {A})

@PVariable()

A

C

@CustomShadowVariable
(sources = {E})

@CustomShadowVariable
(sources = {B, C})

F

E

(ref = F)

G

@PVariable()

B

Dependency graph implies the trigger order

1 2 3 4

OptaPlanner

CVarListener

DVarListener

EVarListener

FGVarListener

afterVariableChanged(A1)

afterVariableChanged(A2)

afterVariableChanged(C1)

afterVariableChanged(C2)

afterVariableChanged(B1)

afterVariableChanged(C1)

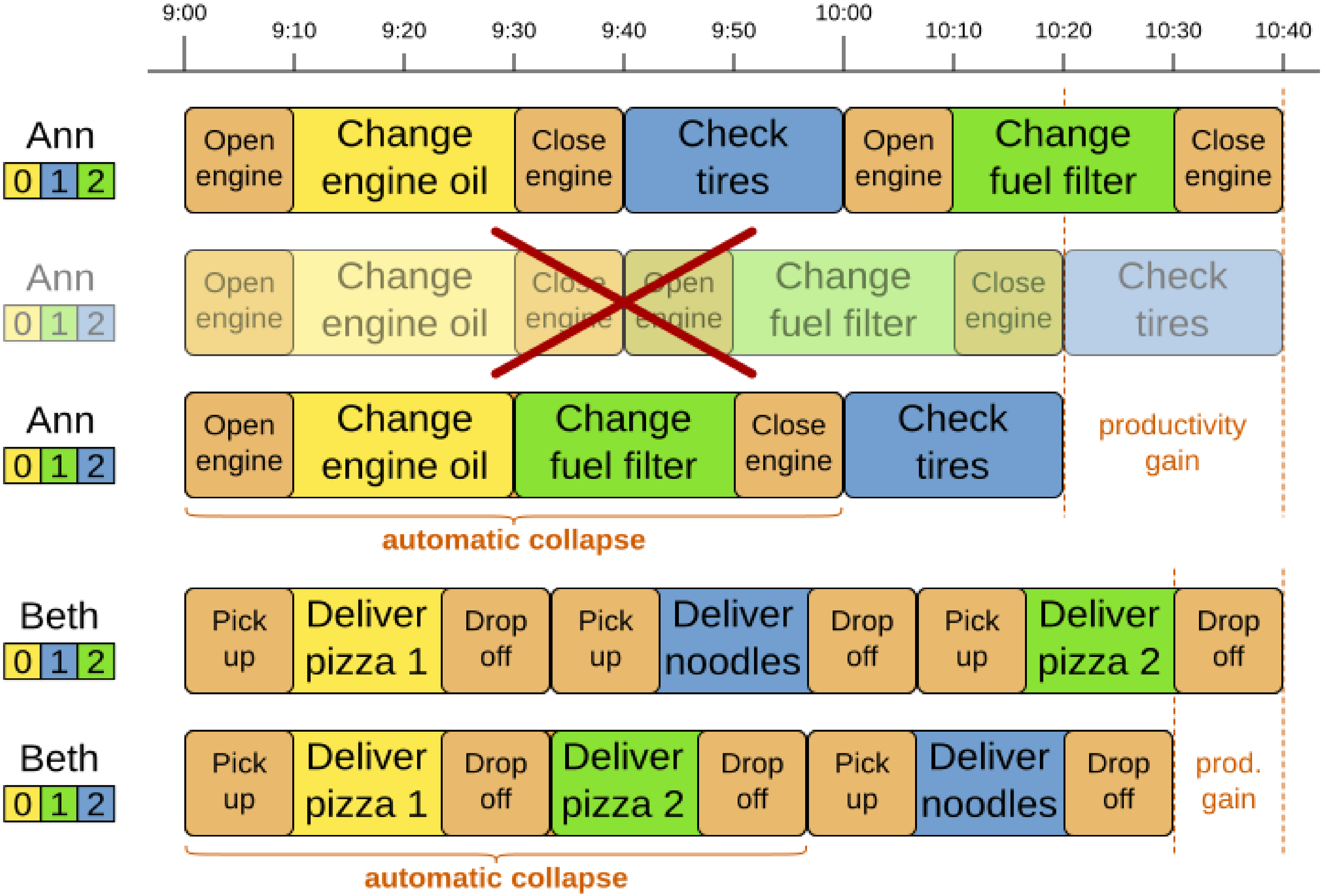
afterVariableChanged(C2)

afterVariableChanged(E1)

The first VariableListener triggers **after** the last genuine variable has changed.

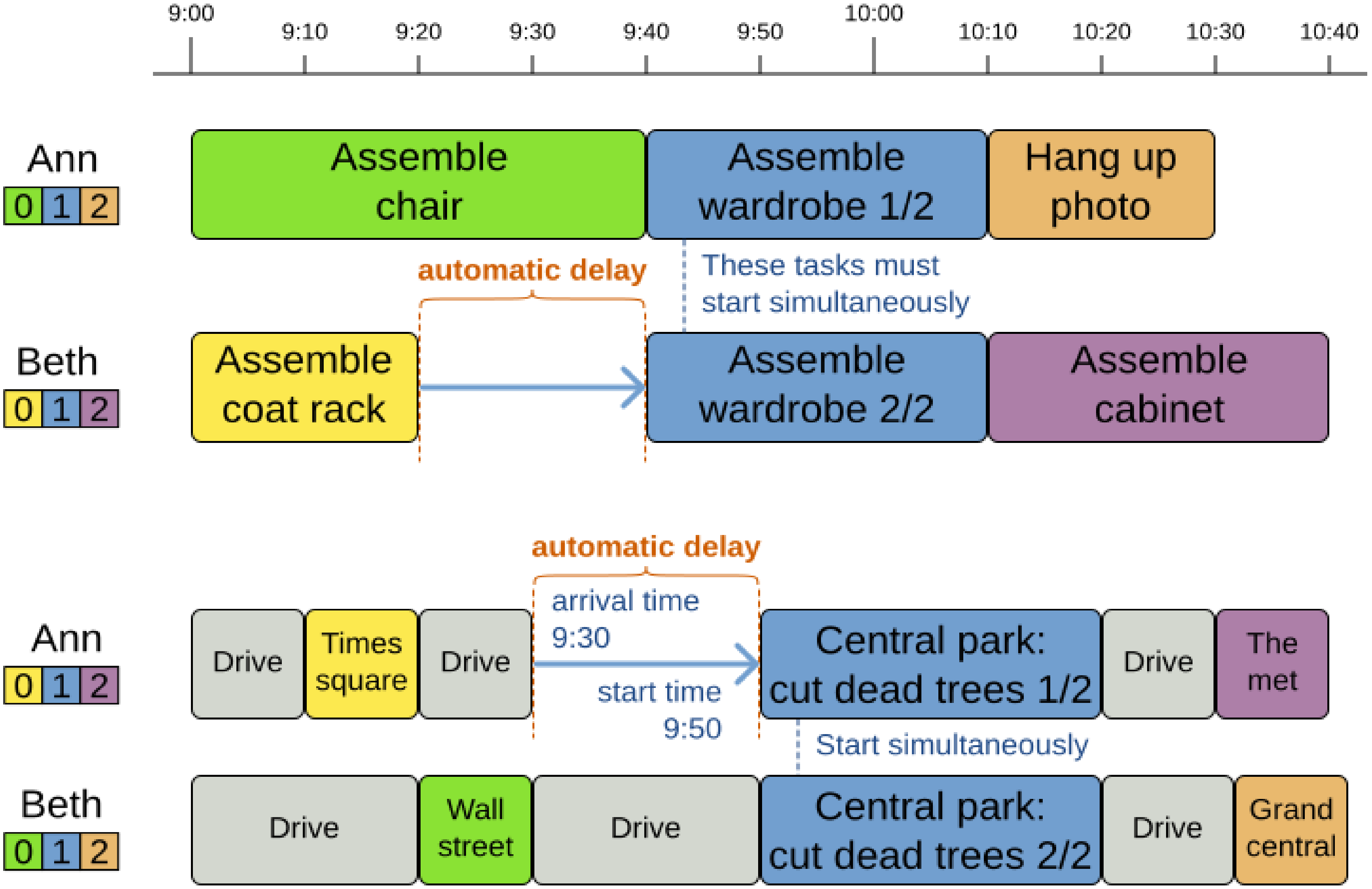
Chained through time: automatic collapse

What if two tasks take less time if (and only if) they are scheduled consecutively?



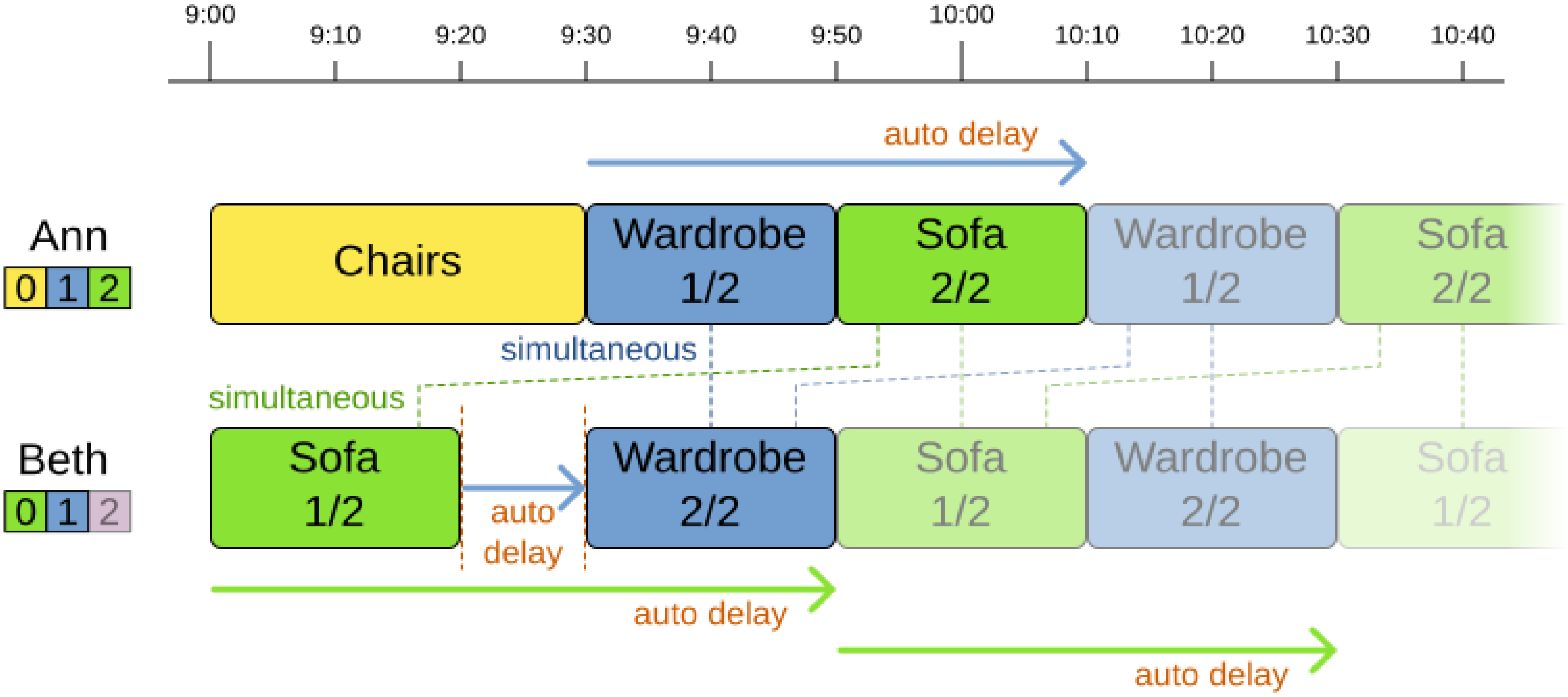
Chained through time: auto delay until last

What if two tasks need to happen at the same time?



Auto delay until last: loop detection

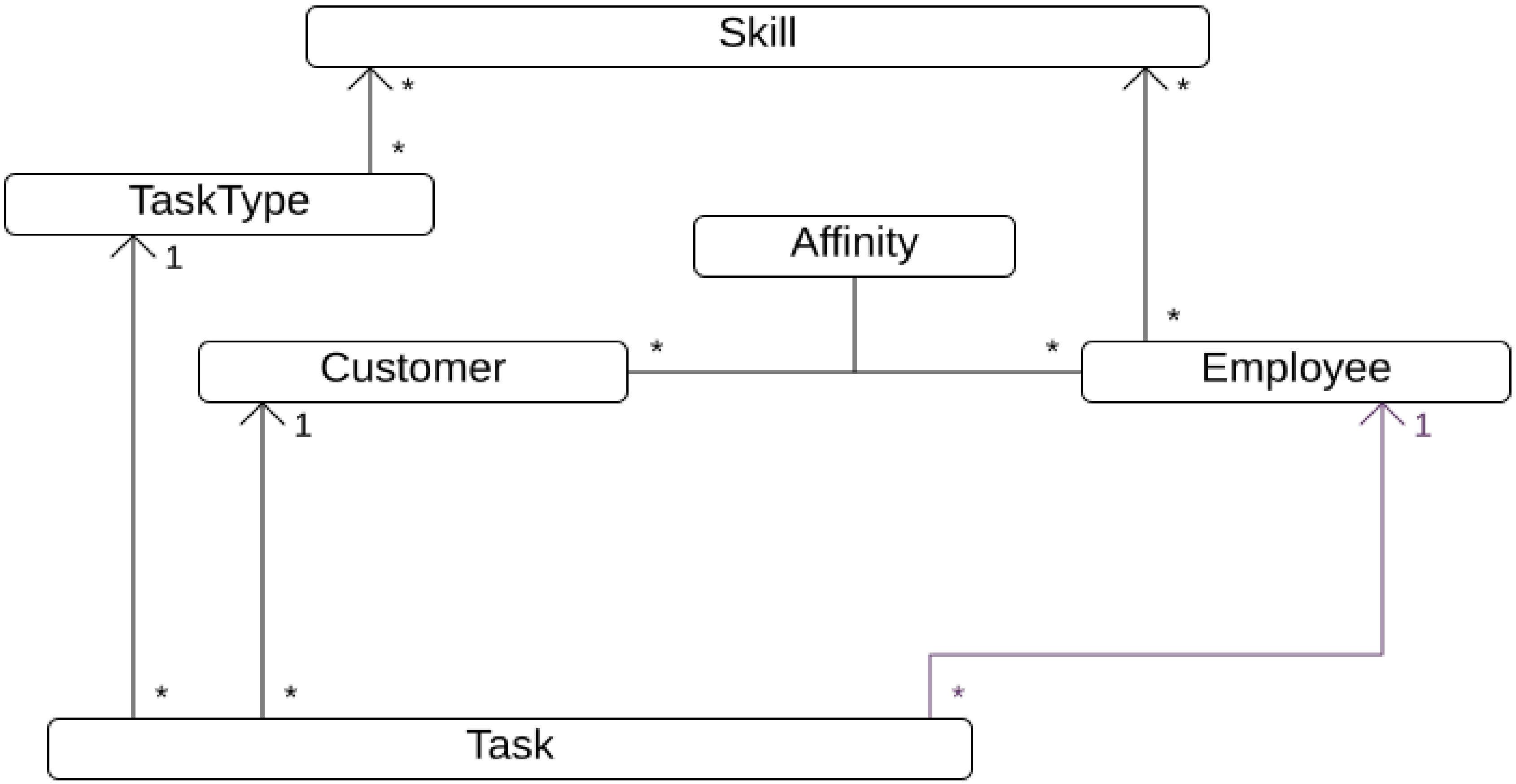
Loop detection is critical with more than once group of simultaneous tasks. It must be deterministic



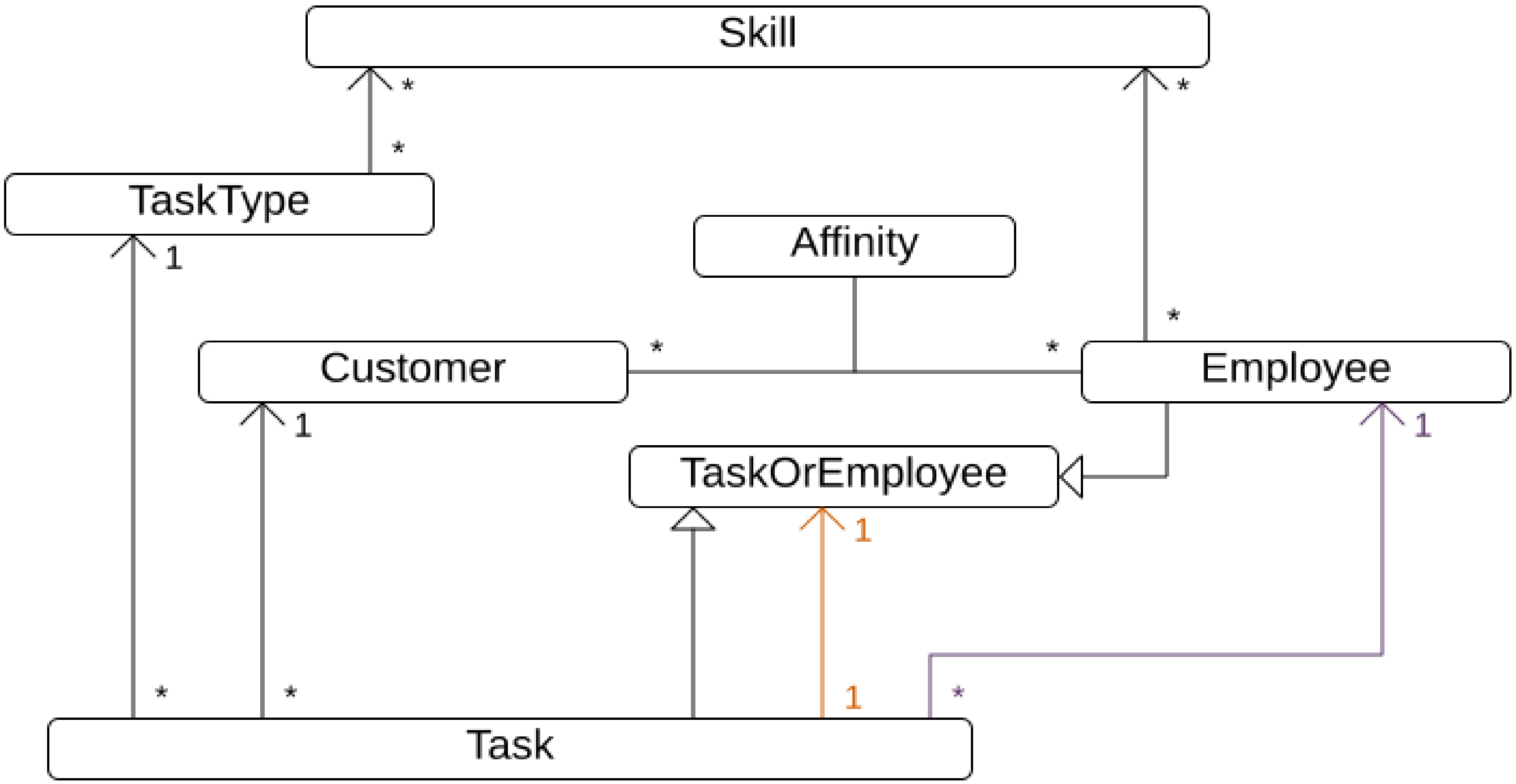
If a loop is detected, it must always be resolved in the same way, regardless of the source of the VariableListener.
Therefore, put all shadow variables involved to arrivalTime null (never).

Exercises

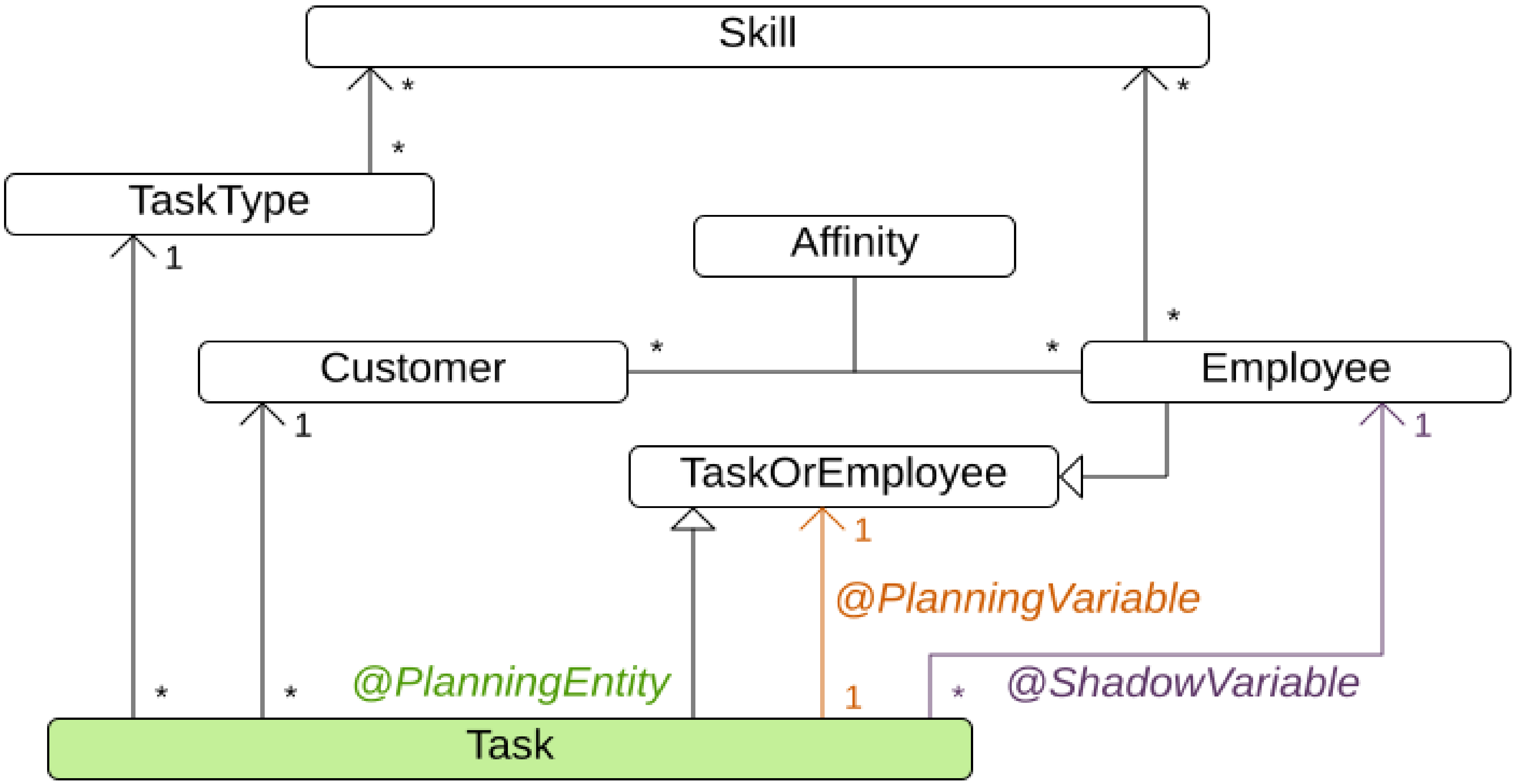
Task assigning class diagram



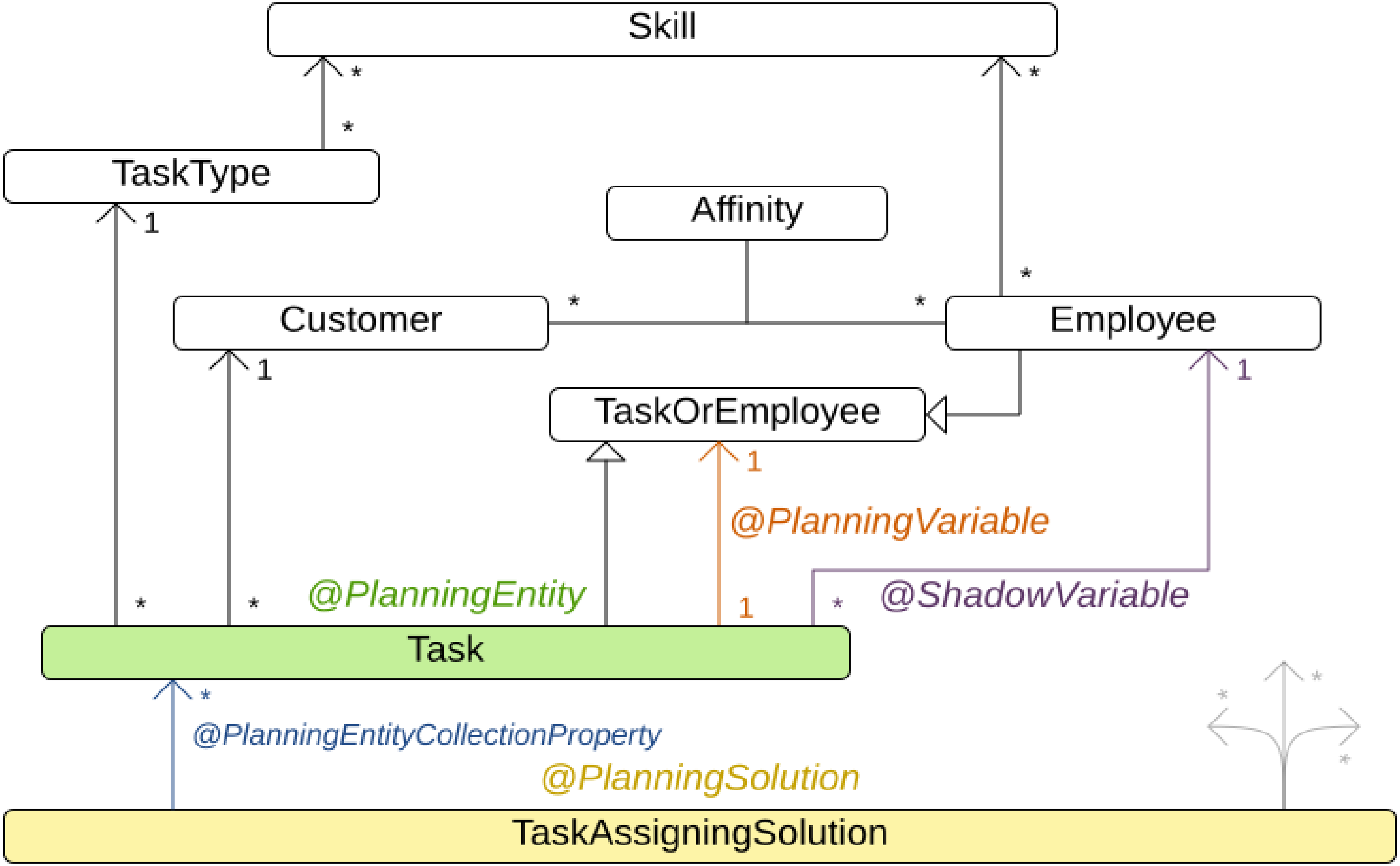
Task assigning class diagram



Task assigning class diagram



Task assigning class diagram



Getting started

Quick starts

- github.com/kiegroup/optaplanner-quickstarts (https://github.com/kiegroup/optaplanner-quickstarts)

```
$ git clone git@github.com:kiegroup/optaplanner-quickstarts.git
...
$ cd optaplanner-quickstarts
$ cd quarkus-school-timetabling
$ mvn quarkus:dev
...
```

Q & A

Homepage	www.optaplanner.org (https://www.optap
Slides	www.optaplanner.org/learn/slides.html (https://www.optaplanner.org/learn/slides
User guide	www.optaplanner.org/learn/documentatio (https://www.optaplanner.org/learn/docur
Feedback	@GeoffreyDeSmet (https://twitter.com/GeoffreyDeSmet)