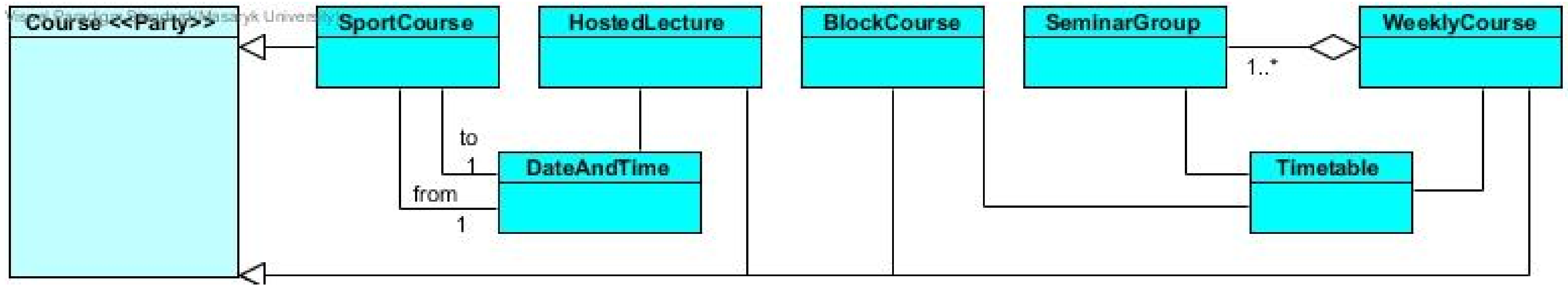
PV167 - IS MU

Authors: Václav Hála, Michaela Bocánová

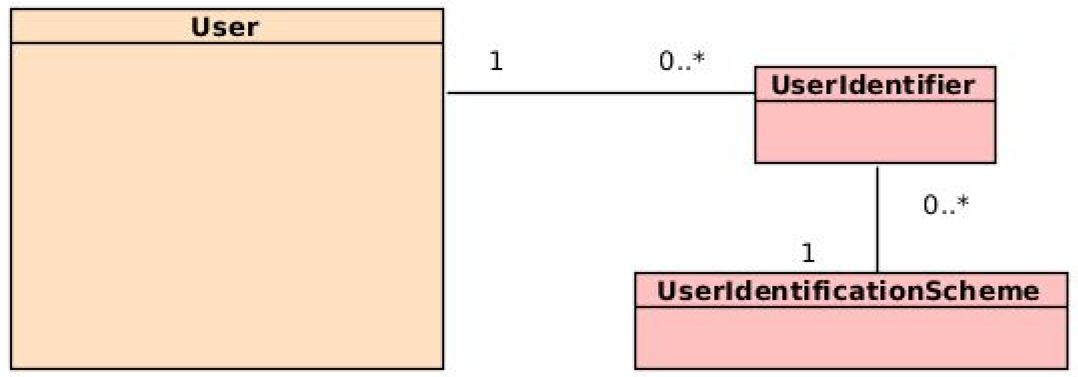
Analysis Patterns

Party - Course



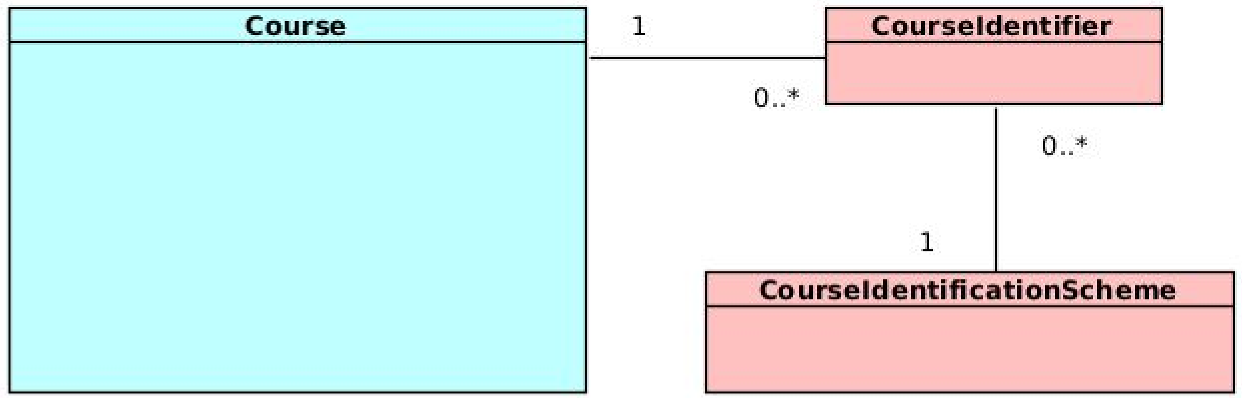
- Generalization of various types of study i.e. weekly course, sport course ...
- Avoid duplication of common relations
- Concrete type is not important in common relations (e.g. StudyPlan)

Identification Scheme - User



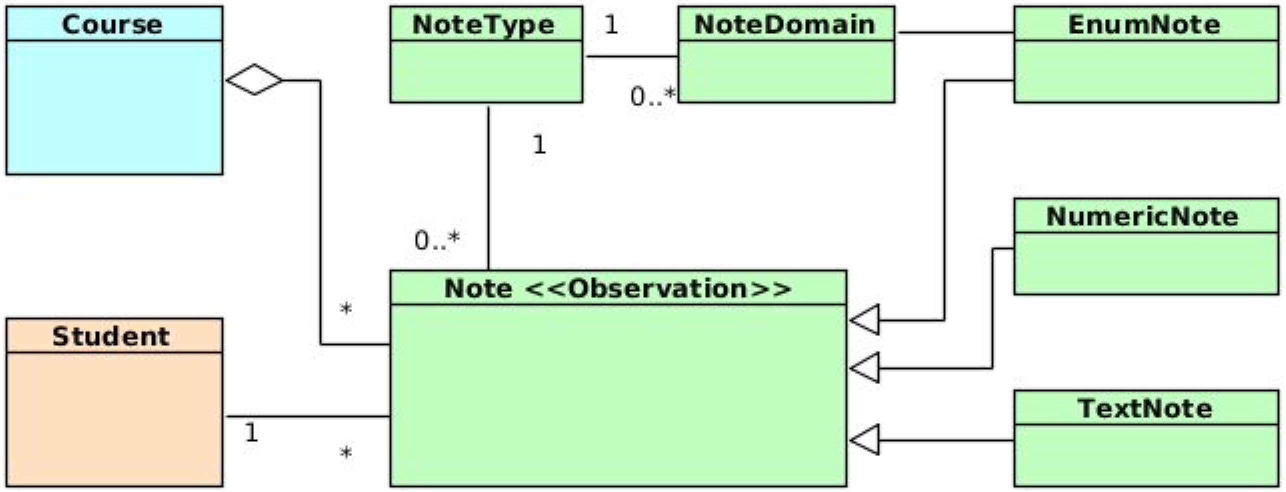
- One User has many names - UCO, alias, ISIC number, …
- Client can select different Identification Schemes for different purposes
- Adding new scheme does not affect User
- Having name in some schemes may be optional (e.g. alias)

Identification Scheme - Course



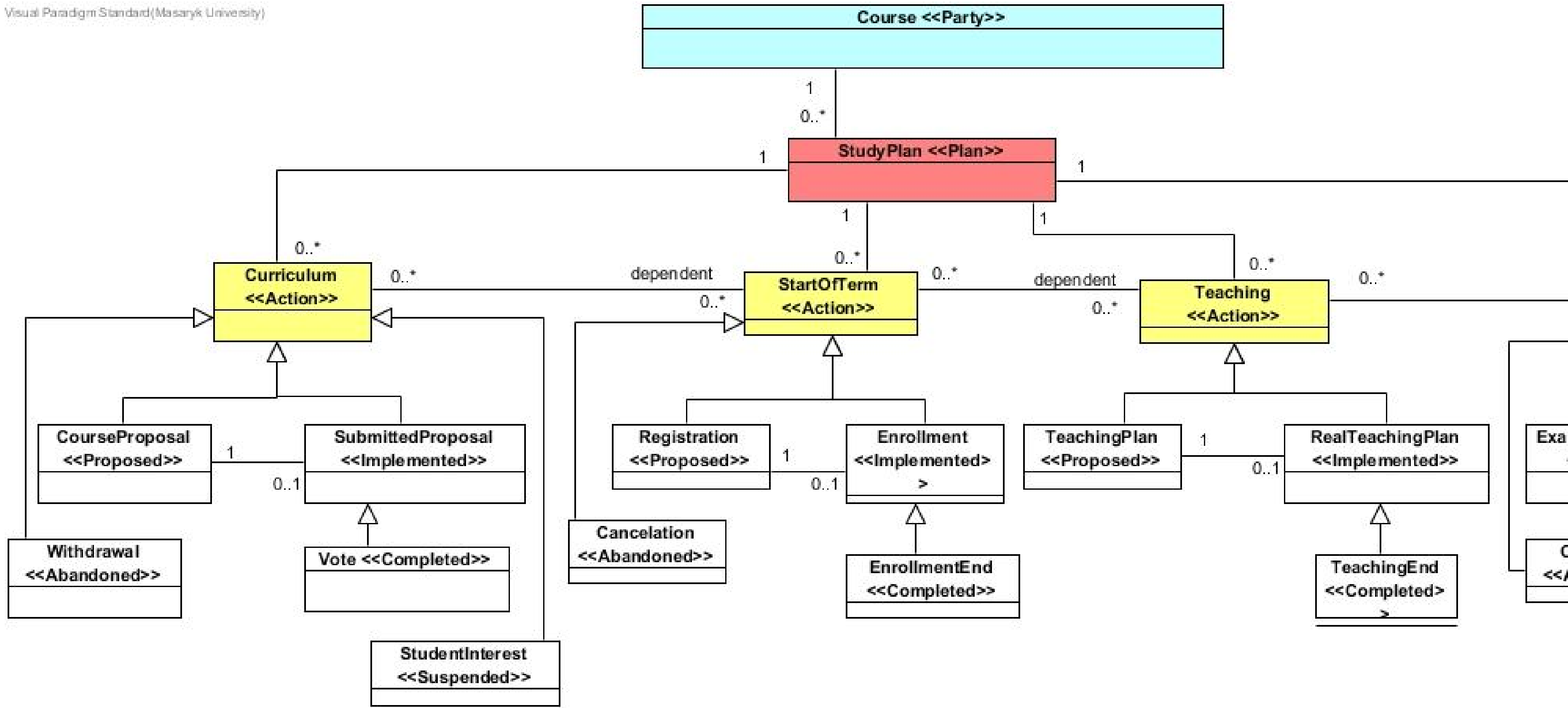
- One Course has many names - code, english name, czech name …
- Client can select different Identification Schemes for different purposes
- When searching for course only some schemes may be used

Observation - Note



- Teacher can take arbitrary notes associated with students
- Note can be of predefined type or user defined
- Note can be numerical / textual / enumerated

Plan - Study Plan

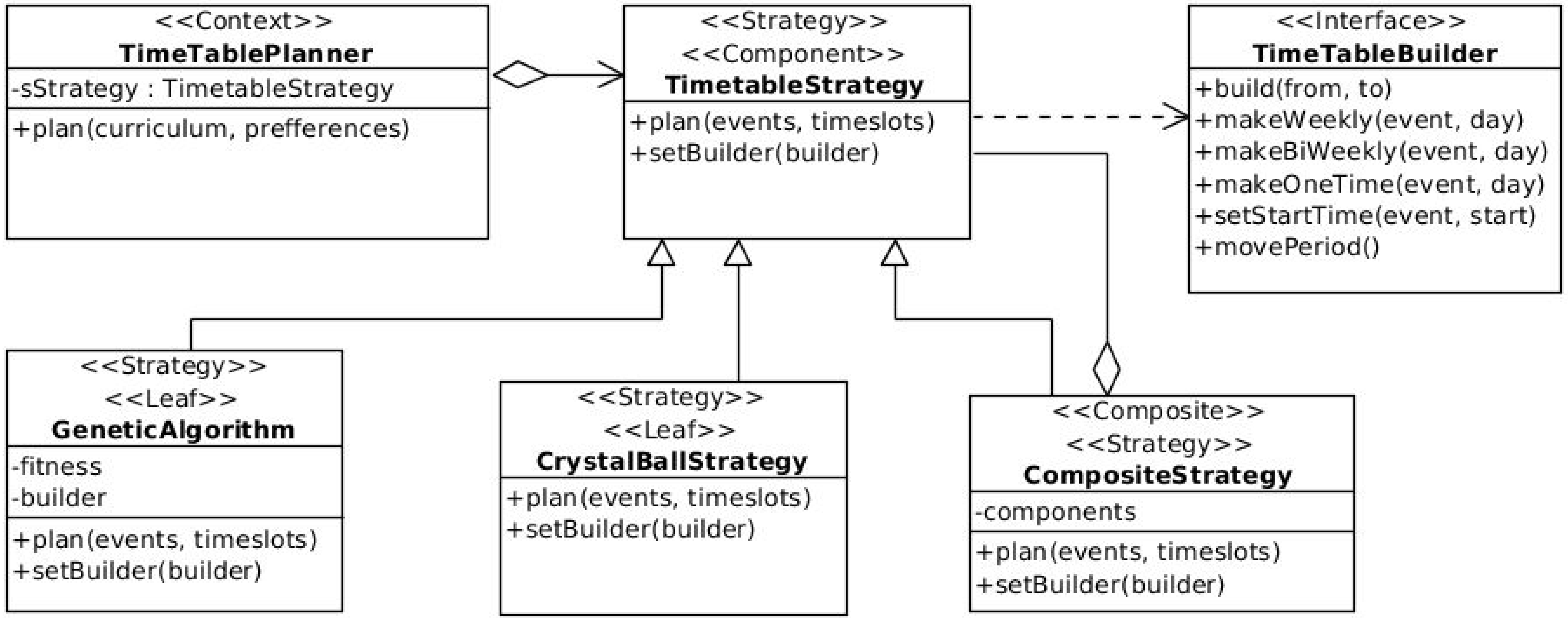


Visual Paradigm Standard(Masaryk University)
Course <<Party>>
1
0..*
StudyPlan <<Plan>>
1
1
1
1
0..*
0..*
0..*
Curriculum <<Action>>
0..*
dependent
0..*
StartOfTerm <<Action>>
0..*
dependent
0..*
Teaching <<Action>>
0..*
CourseProposal <<Proposed>>
1
0..1
SubmittedProposal <<Implemented>>
Withdrawal <<Abandoned>>
Vote <<Completed>>
StudentInterest <<Suspended>>
Registration <<Proposed>>
1
0..1
Enrollment <<Implemented>>
Cancelation <<Abandoned>>
EnrollmentEnd <<Completed>>
TeachingPlan <<Proposed>>
1
0..1
RealTeachingPlan <<Implemented>>
TeachingEnd <<Completed>>
Exa

- Models lifetime of Course from inception to end of term and finals
- Phases have strictly defined ordering, no phase can be skipped
- Every phase starts as Proposed Action and turns to Implemented Action

Design Patterns

Strategy - TimetableStrategy



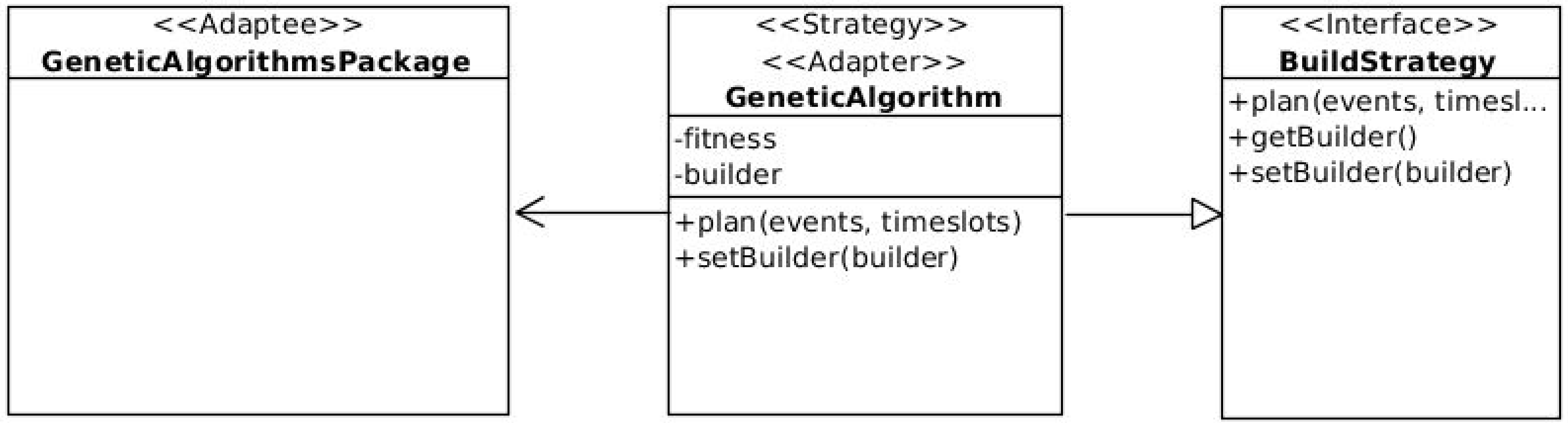
Benefits:

- Wide range of applicable algorithms
- Algorithm can be selected at runtime
- Algorithms can be further composed

Drawbacks:

- Performance penalty for indirection
- Fixed interface all algorithms must adopt

Adapter - GeneticAlgorithmAdapter



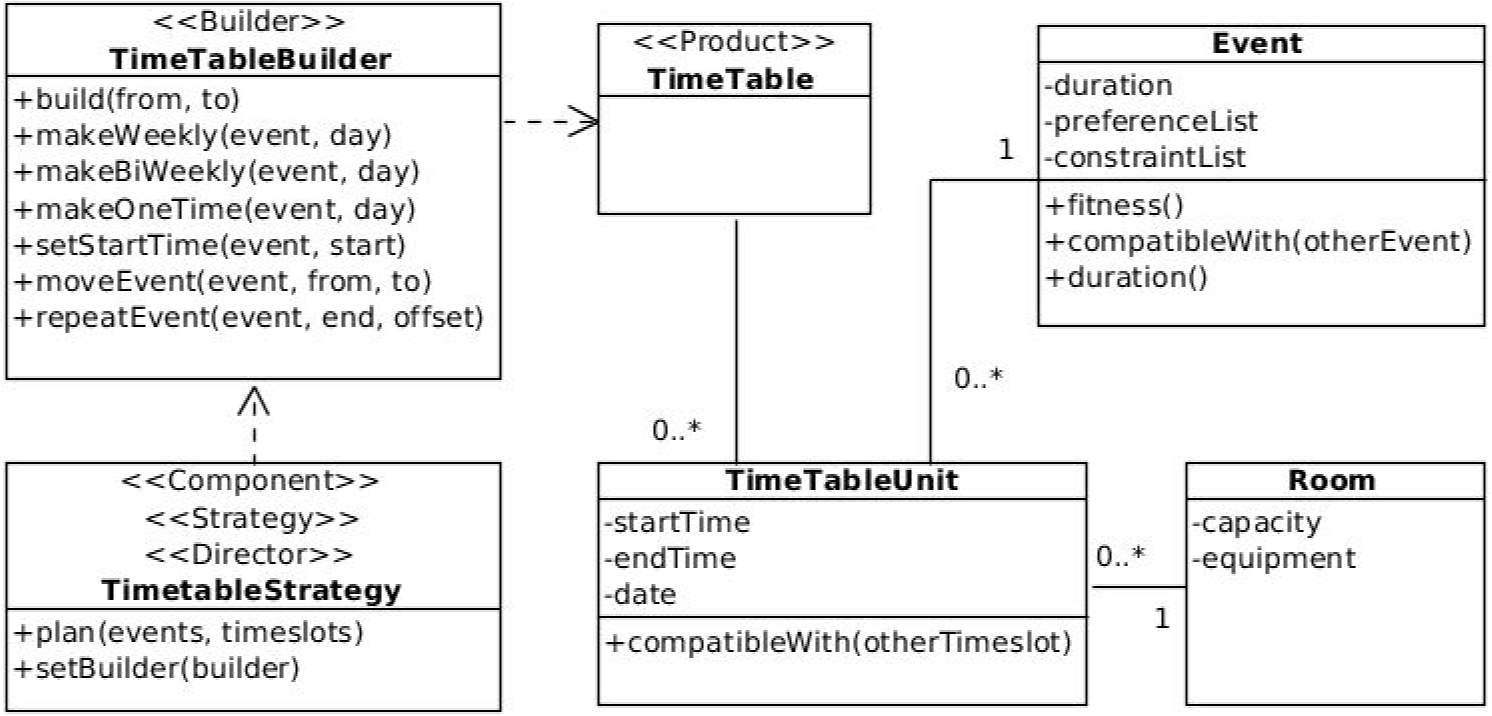
Benefits:

- Enables use of existing library

Drawbacks:

- Functions of library not adaptable to our interface can not be used

Builder - TimetableBuilder



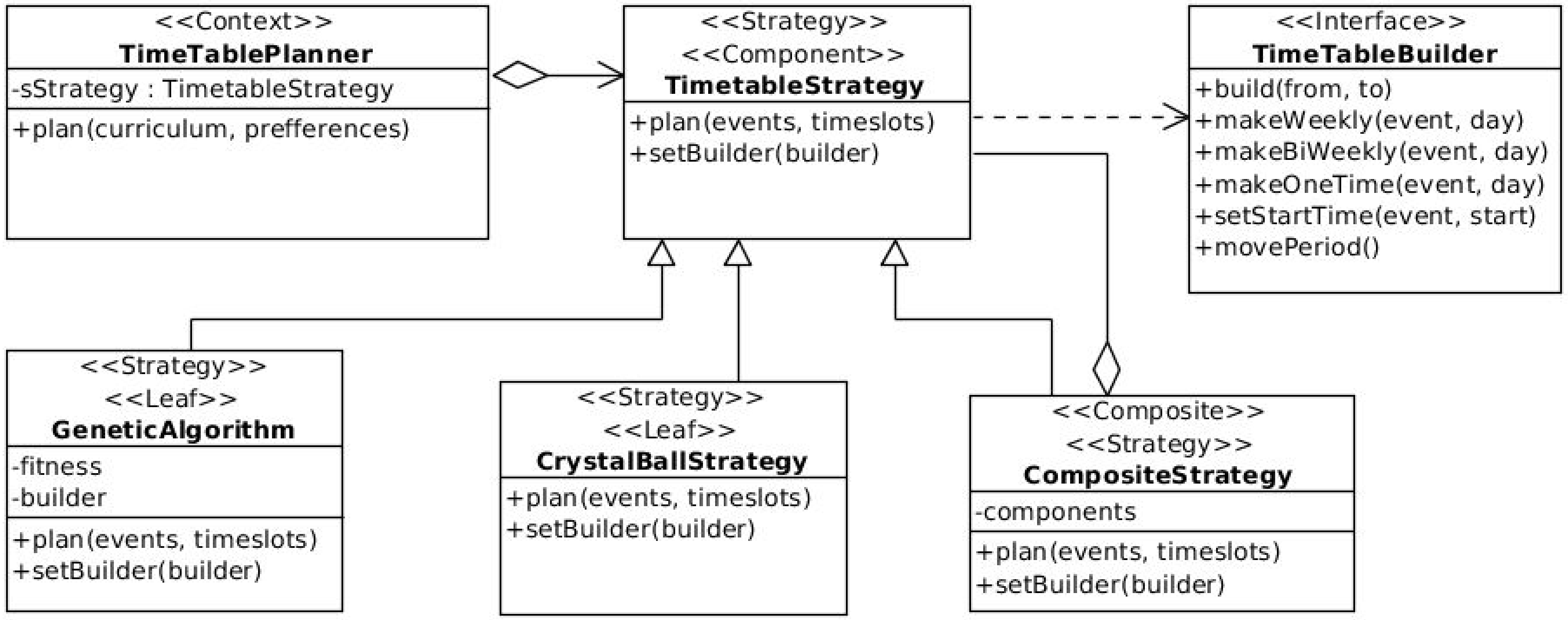
Benefits:

- High-level interface to timetable creation available to planning strategies
- Encapsulated compatibility verification
- Can manipulate group of Events

Drawbacks:

- Low-level Timetable access not available

Composite - CompositeStrategy

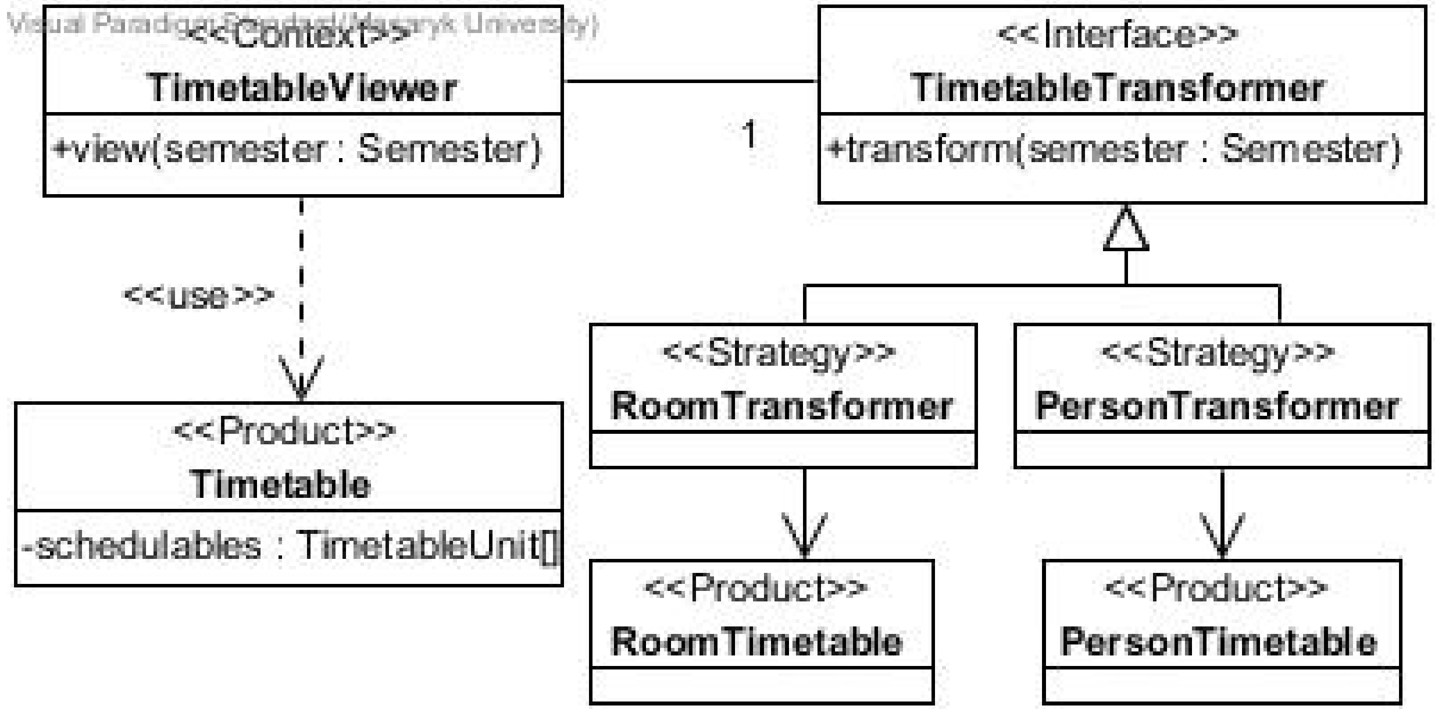


Benefits:

- Different algorithms can be grouped and evaluated recursively with varying inputs
- Client is oblivious of whether single strategy is used or best result from multiple is selected

Drawbacks:

Strategy - TimetableViewer

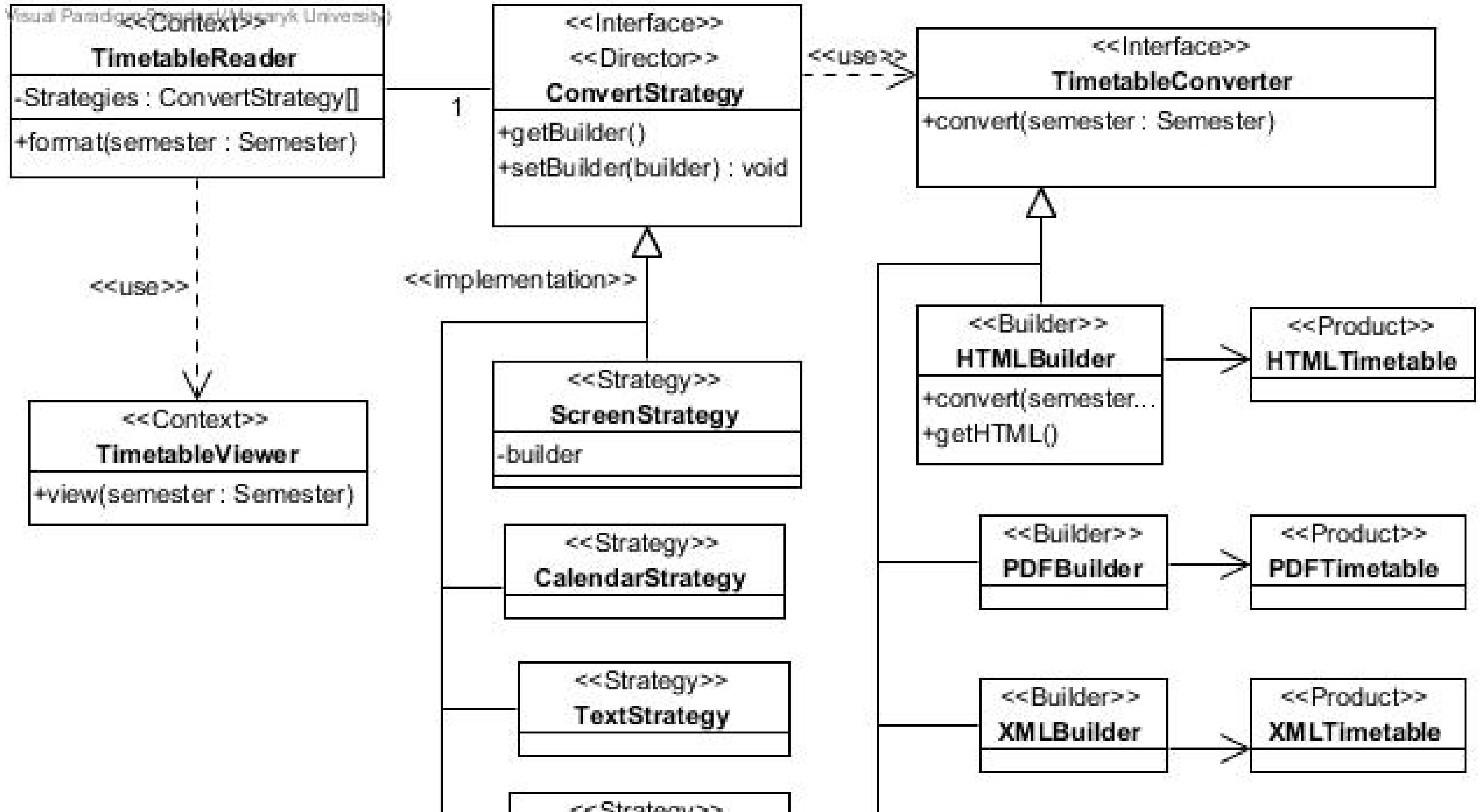


Benefits:

- Different views of one Timetable

Drawbacks:

Builder - TimetableConverter



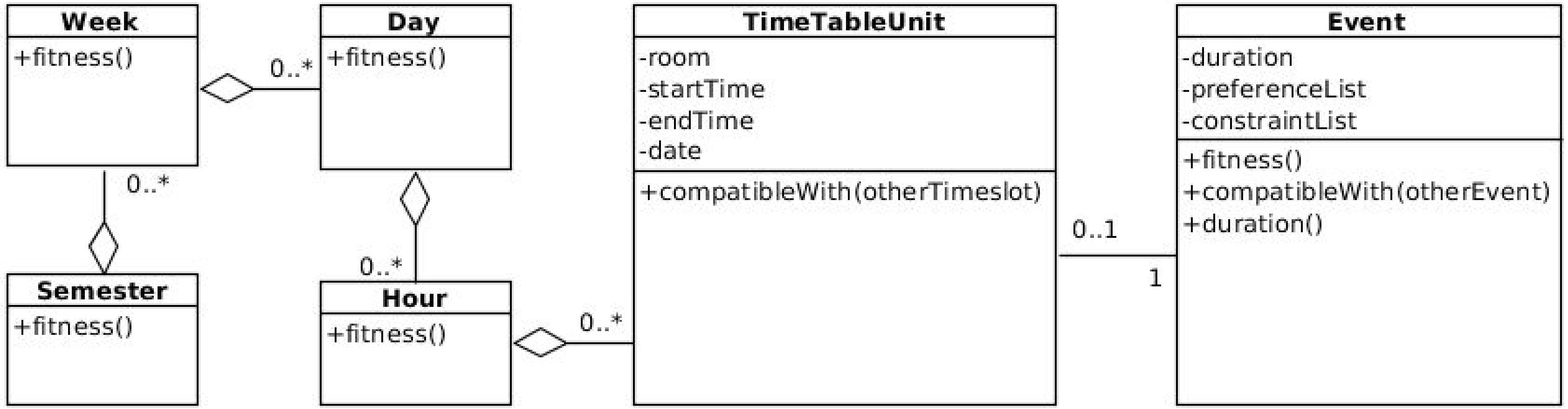
Benefits:

- Decoupling of timetable representation and presentation
- Adding new algorithm has no impact on existing presentations and vice versa

Drawbacks:

- Only API provided by the Builder can be used by the Renderer
- API of the Builder must support all existing ConvertStrategies, not every method of the Builder is used by every Strategy

Composite - Semester Hierarchy



Benefits (of not using Composite):

- Compile-time checks for concrete types
- Hierarchy never changes
- Client can require concrete type (e.g. Week)
- Repeated code for delegation to children can be separated to shared helper

Drawbacks (of using Composite):

- Composite would require runtime checks
- No restriction on possible types

Thank you for your attention