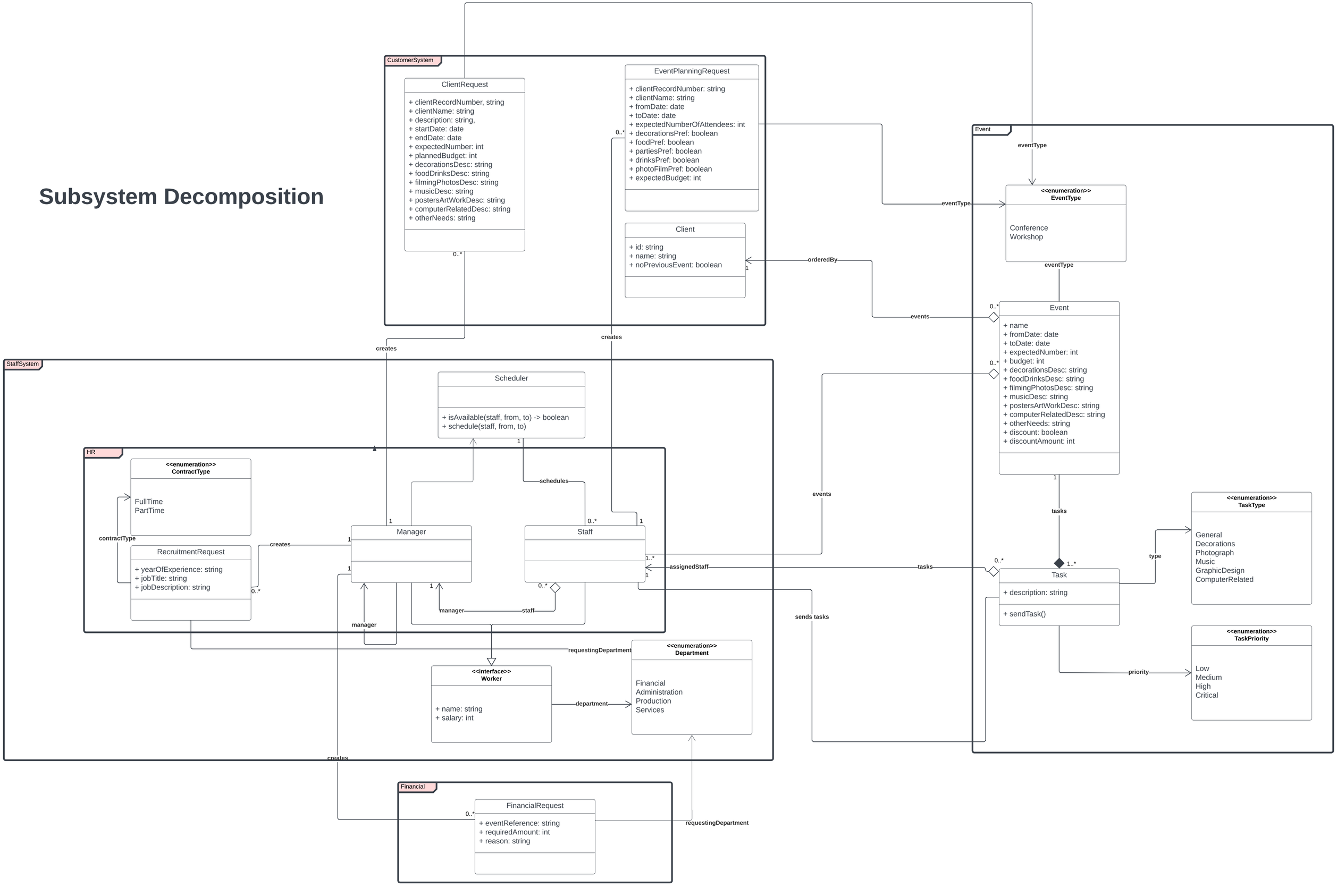
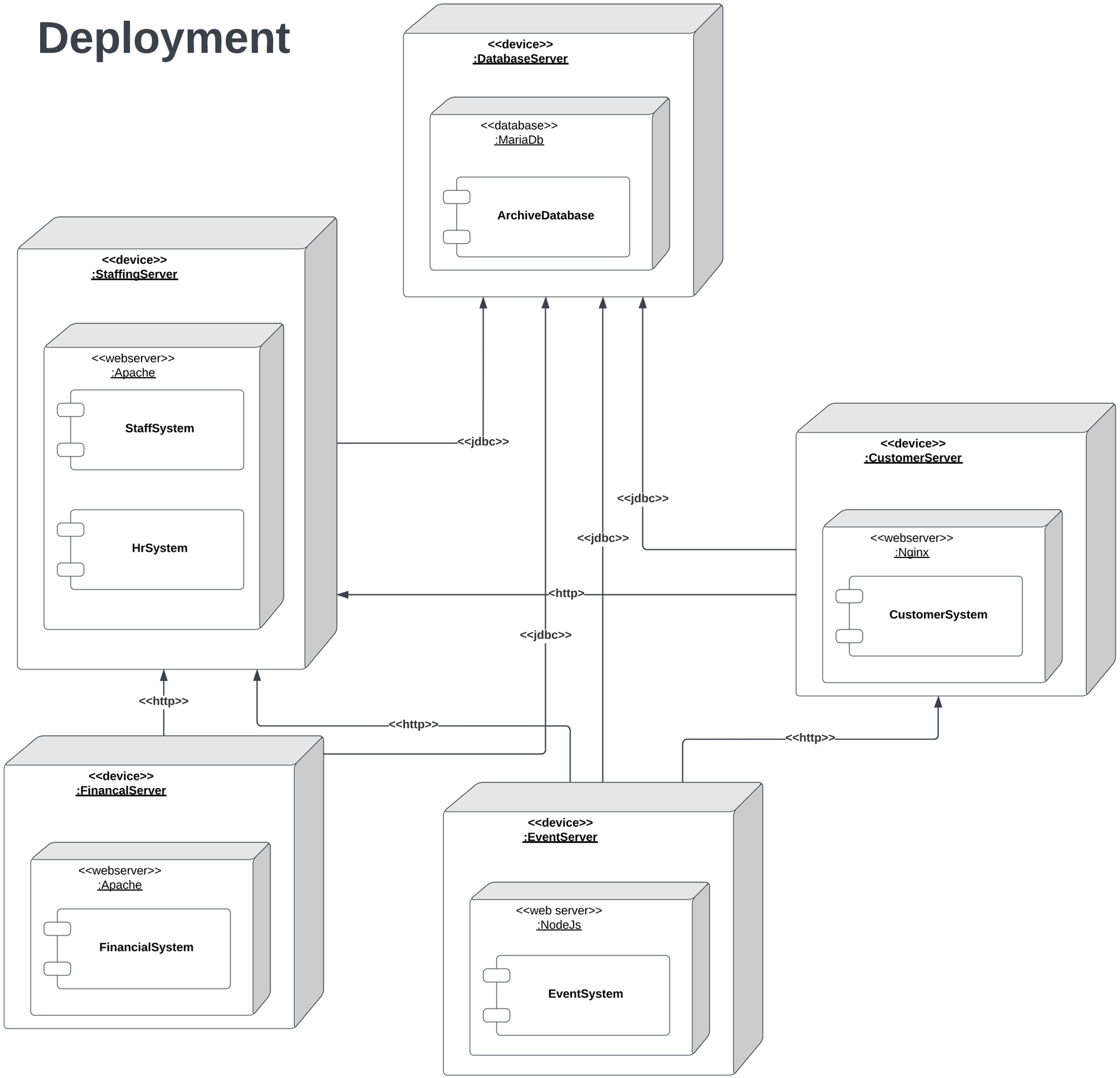
Subsystem Decomposition



Deployment



DatabaseServer	Stores archive for events, customers, and staff. Uses MariaDb was chosen for the large ammount of data stored.
StaffingServer	Handles request regarding staff and HR related issues. Uses Apache for reliability.
CustomerServer	Handles reqeuest regarding client and initiating events from client.
EventServer	Handles event related data and processing
FinancialServer	Responsible for handling fincial related services

Persistent Storage

Event	Stores all information about events and is the true source of event data. Stored on a centralized MariaDB database.
Staff Record	Keep track of all staff info and schedulings. Stored on a centralized MariaDB database.
Client Record	Store all information about Clients current and historical events and requests. Stored on a centralized MariaDB database.
Task	Connected to event, keeps track of scheduling and activities and personnel for an event planning task. Stored on a centralized MariaDB database.
Event Planning Request	Stored data from the 'Event planning request' form. Stored on a centralized MariaDB database.
Client Request	Contains data fro mthe client request form. Stored on a centralized MariaDB database.
Financial Request	Stores information regarding previous financial requests for the financial department to use. Stored on a centralized MariaDB database.
Recruitment Request	Stores all information about recruitment process, in order to simplify future recruitments. Stored on a centralized MariaDB database.

Access Control

Object Actors	Financial Request	Task	Event Planning Request
Financial Manager	create() writeFeedback() accept() deny()		
Senior Customer Support	create()		create() writeFeedback() decline()
Production Manager	create()	create() checkComments() writeComment()	
Customer Support			create() forwardToSenior()

Financial Manager can handle financial request in every possible way, including creating, writing feedback, accepting, and denying.

Senior Customer Support handles Event planning request, including creating writing feedback and declining a request immediately would it seem infeasible.

Production Manager can create both FInancial Requests (but is not allowed to do anything else) and Tasks for the project, as well as managing the comments for a task.

Customer Support can create Event Planning requests, however it is only able to forward it Senior for further reviewing and feedback.

Global Control Flow

The system allows evenet request to enter through dedicated staff, Customer Service. The event it following a strict path through Senior Customer Management, Financial Manager, Administration Manager, each with their respective tasks that only they are allowed to perform, such as writing feedback or declining the event request completely.

Boundary Conditions

Use case name	CreateEvent
Entry Condition	Customer Service creates a form using information supplied by Client
Flow of Events	1. Customer Service Looks over the information 2. Customer Service Forwards the form to the Senior Customer Service
Quality Conditions	If the Event appears infeasible for the Senior Customer Manager, it will be discarded immedieately
Exit Condition	- Senior Customer Manager approves the event and forward it to Financial Department for feedback or - Senior Customer Manager declines the event

Use case name	RequestAsset
Entry Condition	Production Manager fills in a form using information supplied by the Client business meeting result
Flow of Events	1. Upon filling the form, it is sent to the centralized system. 2. Subteam checks the system for their tasks
Quality Conditions	If a task is finished, it should be archived and not delted
Exit Condition	Task is delayed due to lack of staff or Subteam can see the task in their lists of tasks

Design Patterns

Design Pattern	
Repository	This pattern will be implemented to access the data in the archive stored in the database, such as Client Record, Staff Record etc. This will be used since a single monolithic database approach is used.
Facade	This pattern will be used for the scheduler, as this could involve multiple steps, such as calendar inspection, comparision. It might also need to take into account semseter and other form of vacacies. A facade will simplify the scheduler providing a simple interface such as "schedule(emp1, emp2)".
Observer	This pattern will be used together with tasks. In the system modelled tasks stores comments and whether they are completed or not, which is useful for Production Staff and Production Manager. Obvserver will be used to notify the other part when a task is updated (such as adding a comment or marking the task complete). Implementing the Observer pattern could simplify future development when notification are needed.

OCL

endDate has to be after startDate

```
Context ClientRequest::constructor(..., startDate: date, endDate: date, ...)
   Pre: endDate > startDate
```

Event has to be at least a month away for time to plan

```
Context ClientRequest::constructor(..., startDate: date, endDate: date, ...)
   Pre: startDate-30 >= date.today
```

Has to select at least one preference

```
Context EventPlanningRequest::constructor(...)
   Pre: *pref->exists(x | x = true)
```

Discount for clients with more than 5 events

```
Context Event Inv:
   discount = count(client.events) > 10
```

Client can't be created with ID same as other client

```
Context Client::constructor(...):
   !( clients->exists(client | client.id = id) )
```