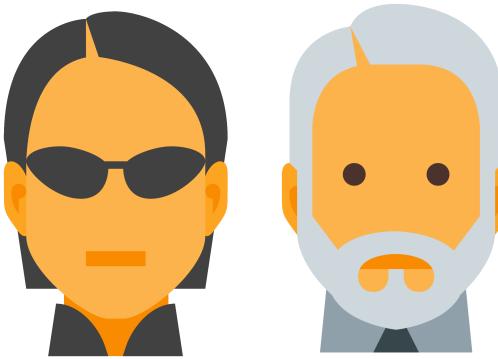


155 PATTERNS



Wilmer Krisp

278 SOFTWARE ARCHITECTURE
DESIGN MODELS

COMPLETE CATALOG OF ALL CLASSICAL
PATTERNS IN THE ARCHIMATE LANGUAGE

01

Design Patterns

Elements of Reusable Object-

GANG OF FOUR

02

Enterprise patterns

Catalog of Patterns of Enterprise

MARTIN FOWLER

03

Analysis Patterns

Reusable Object Models

MARTIN FOWLER

04

Domain Driven Design

Tackling Complexity in the Heart

ERIC EVANS

05

Clean Architecture

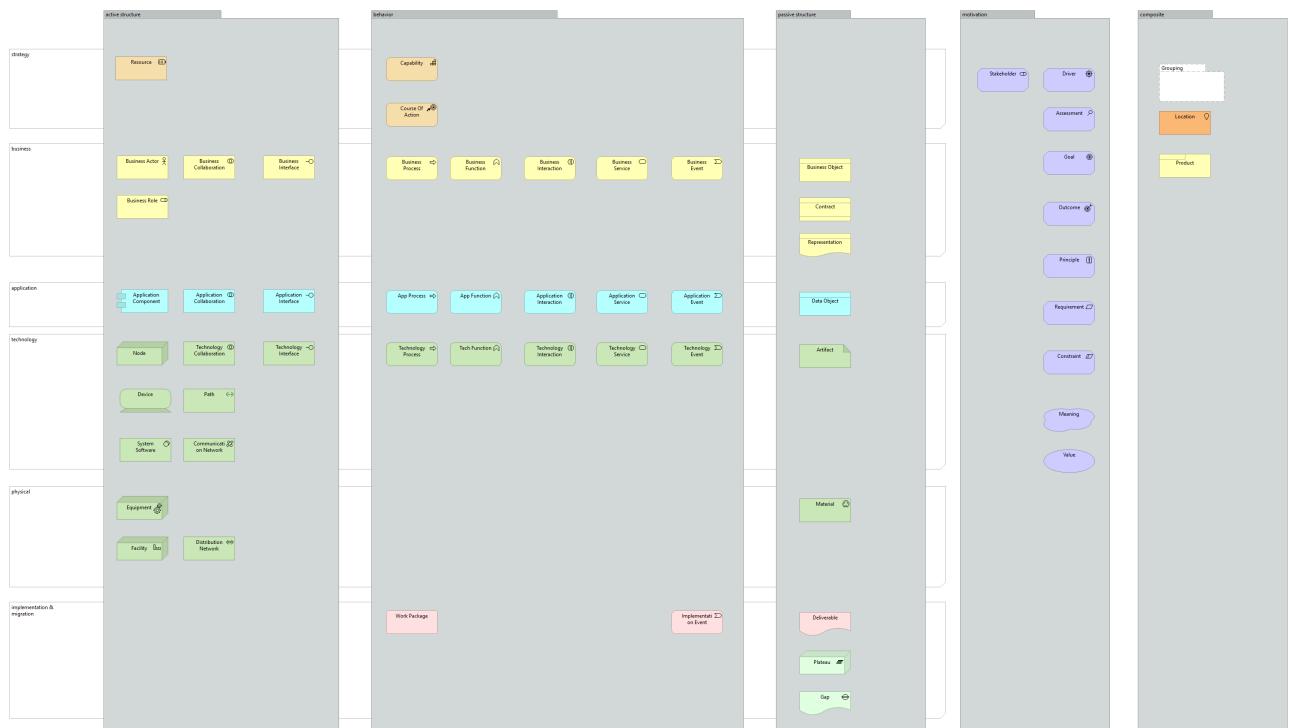
Agile Software Development

UNCLE BOB

USED NOTATION

ARCHIMATE METAMODEL

The Open Group



CREATIONAL PATTERNS	10
ABSTRACT FACTORY	11
BUILDER	15
FACTORY METHOD	17
PROTOTYPE	19
SINGLETON	21
STRUCTURAL PATTERNS	23
ADAPTER OF CLASS	24
ADAPTER OF OBJECT	26
BRIDGE	29
COMPOSITE	33
DECORATOR	35
FAÇADE	37
FLYWEIGHT	40
FLYWEIGHT + COMPOSITE	42
PROXY	43
BEHAVIORAL PATTERNS	45
CHAIN OF RESPONSIBILITY	46
COMMAND	48
INTERPRETER	50
ITERATOR	52
MEDIATOR	54
MEMENTO	56
OBSERVER	58
STATE	60
STRATEGY	62
TEMPLATE METHOD	64
VISITOR	67

BUSINESS LOGIC	73
DOMAIN MODEL	74
SERVICE LAYER	75
TRANSACTION SCRIPT	76
TABLE MODULE	77
DATA SOURCES	78
ACTIVE RECORD	79
DATA MAPPER	80
ROW DATA GATEWAY	81
TABLE DATA GATEWAY	82
MODELLING BEHAVIOUR	83
IDENTITY MAP	84
LAZY LOAD	85
UNIT OF WORK.	86
MODELLING STRUCTURE HIERARCHY	87
CLASS TABLE INHERITANCE	88
CONCRETE TABLE INHERITANCE	89
INHERITANCE MAPPERS	90
SINGLE TABLE INHERITANCE	91
MODELLING STRUCTURE RELATIONS	92
ASSOCIATION TABLE MAPPING	93
DEPENDENT MAPPING	94
EMBEDDED VALUE	95
FOREIGN KEY MAPPING	96
IDENTITY FIELD	97
SERIALIZED LOB	98

METADATA	99
METADATA MAPPING	100
QUERY OBJECT	101
REPOSITORY	102
WEB REPRESENTATION CONTROLLER	103
MODEL VIEW CONTROLLER	104
APPLICATION CONTROLLER	105
FRONT CONTROLLER	106
PAGE CONTROLLER	107
WEB REPRESENTATION VIEW	108
TEMPLATE VIEW	109
TRANSFORM VIEW	110
TWO STEP VIEW	111
DATA TRANSFER OBJECT	113
REMOTE FAÇADE	114
PARALLEL PROCESSING	115
COARSE-GRAINED LOCK	116
IMPLICIT LOCK	117
OPTIMISTIC OFFLINE LOCK	118
PESSIMISTIC OFFLINE LOCK	119
SESSION STATE	120
CLIENT SESSION STATE	121
DATABASE SESSION STATE	122
SERVER SESSION STATE	123
COMMON PATTERNS	124
GATEWAY	125
LAYER SUPERTYPE	126
MAPPER	127
MONEY	128
PLUGIN	129
RECORD SET	130
REGISTRY	131
SEPARATED INTERFACE	132
SERVICE STUB	133
SPECIAL CASE	134
VALUE OBJECT	135

3 ANALYSIS PATTERNS 136

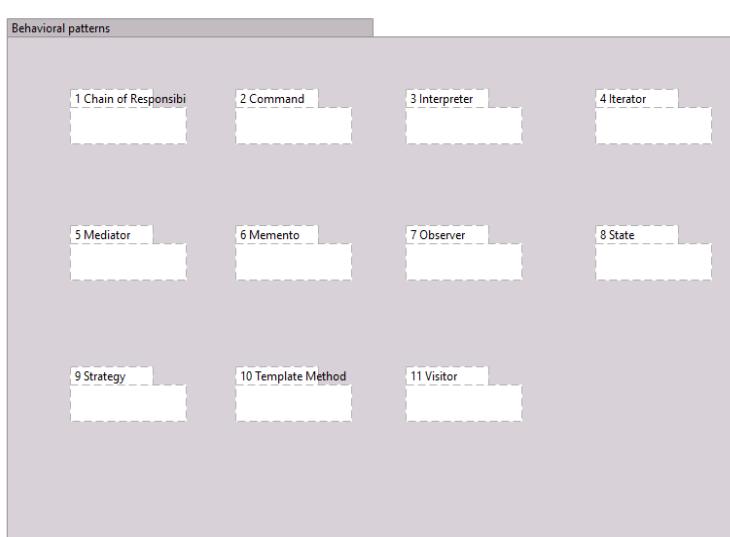
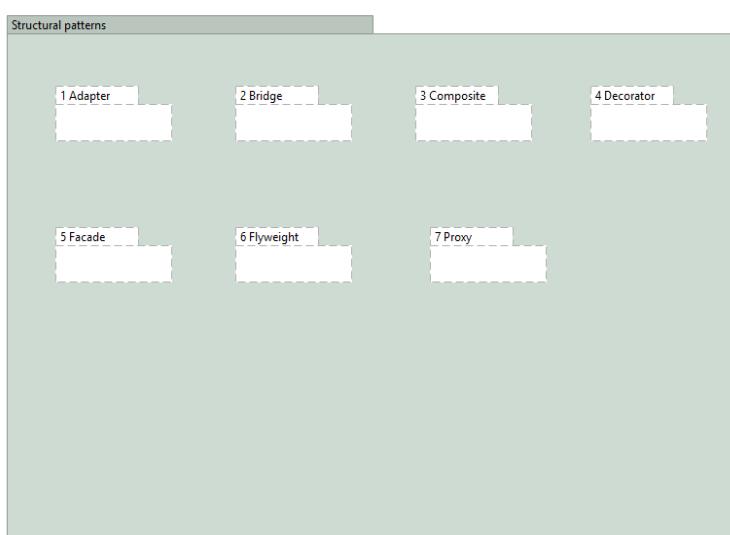
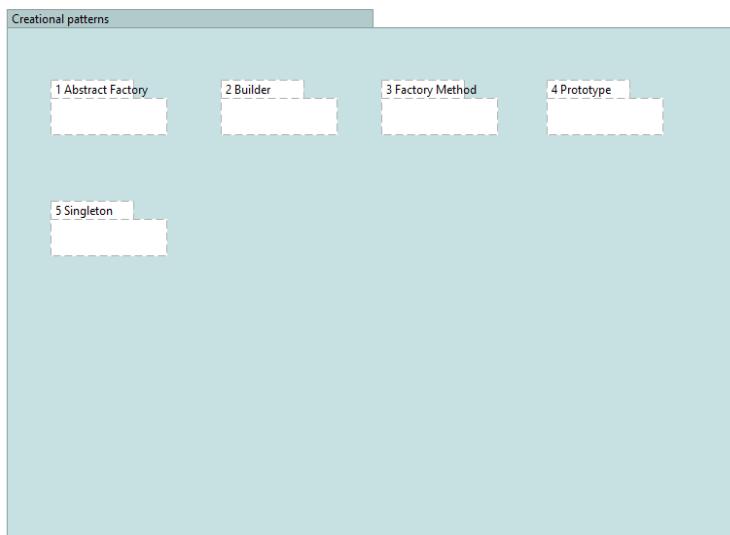
ACCOUNTABILITY	138
PARTY	139
ACCOUNTABILITY	140
ORGANIZATION HIERARCHIES	142
ORGANIZATION STRUCTURE	143
ACCOUNTABILITY KNOWLEDGE LEVEL	144
PARTY TYPE GENERALIZATIONS	145
HIERARCHIC ACCOUNTABILITY	146
OPERATING SCOPES	147
POST	148
OBSERVATIONS AND MEASUREMENTS	149
QUANTITY	150
CONVERSION RATIO	151
OBSERVATIONS AND MEASUREMENTS	152
COMPOUND UNITS	153
MEASUREMENT	154
OBSERVATION	155
SUBTYPING OBSERVATION CONCEPTS	156
PROTOCOL	157
DUAL TIME RECORD	158

REJECTED OBSERVATION	159
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION	160
ASSOCIATED OBSERVATION	161
PROCESS OF OBSERVATION	162
OBSERVATIONS FOR CORPORATE FINANCE	163
ENTERPRISE SEGMENT	164
MEASUREMENT PROTOCOL	165
RANGE	166
OBSERVATIONS FOR CORPORATE FINANCE	167
PHENOMENON WITH RANGE	170
REFERRING TO OBJECTS	171
NAME	172
IDENTIFICATION SCHEME	173
OBJECT MERGE	174
OBJECT EQUIVALENCE	175
REFERRING TO OBJECTS	176
INVENTORY AND ACCOUNTING	177
ACCOUNT	178
TRANSACTIONS	179
SUMMARY ACCOUNT	180
MEMO ACCOUNT	181
POSTING RULES	182
INVENTORY AND ACCOUNTING	183
INDIVIDUAL INSTANCE METHOD	184
POSTING RULE EXECUTION	185
POSTING RULES FOR MANY ACCOUNTS	186
CHOOSING ENTRIES	187
ACCOUNTING PRACTICE	188
SOURCES OF AN ENTRY	189
BALANCE SHEET AND INCOME STATEMENT	190
CORRESPONDING ACCOUNT	191
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)	192
SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)	193
BOOKING ENTRIES TO MULTIPLE ACCOUNTS	194
PLANNING	195
PROPOSED AND IMPLEMENTED ACTION	196
COMPLETED AND ABANDONED ACTIONS	197
SUSPENSION	198
PLAN	199
PROTOCOL	200
RESOURCE ALLOCATION	201
PLANNING	202
PLANNING (NO OUTCOME)	203
OUTCOME AND START FUNCTIONS	204
TRADING	205
CONTRACT	206
PORTFOLIO	207
QUOTE	208
SCENARIO	209
TRADING	210
DERIVATIVE CONTRACTS	211
FORWARD CONTRACTS	212
OPTIONS	213
PRODUCT	214
SUBTYPE STATE MACHINES	215
PARALLEL APPLICATION AND DOMAIN HIERARCHIES	216
DERIVATIVE CONTRACTS	217
TRADING PACKAGES	218
MULTIPLE ACCESS LEVELS TO A PACKAGE	219
MUTUAL VISIBILITY	220

TRADING PACKAGES	221
LAYERED ARCHITECTURE	222
TWO-TIER ARCHITECTURE	223
THREE-TIER ARCHITECTURE	224
PRESENTATION AND APPLICATION LOGIC	225
DATABASE INTERACTION	226
TYPE MODEL DESIGN	227
IMPLEMENTING ASSOCIATIONS	228
IMPLEMENTING GENERALIZATION	229
OBJECT CREATION	230
OBJECT DESTRUCTION	231
ENTRY POINT.	232
IMPLEMENTING CONSTRAINTS	233
4 DOMAIN DRIVEN DESIGN	234
MODEL AND STRUCTURAL ELEMENTS	234
MODEL-DRIVEN DESIGN	235
LAYERED ARCHITECTURE (ASYMMETRIC)	238
HEXAGONAL ARCHITECTURE (SYMMETRIC)	240
COMPOSITE UI	244
ENTITIES	245
VALUE-OBJECTS	247
DOMAIN SERVICES	249
MODULES	254
AGGREGATES	255
AGGREGATE ROOT	257
BEHAVIOR-FOCUSED AGGREGATE ROOT	258
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION	259
PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES	260
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY	261
FACTORIES	262
REPOSITORIES	264
SUPPLE DESIGN	268
UBIQUITOUS LANGUAGE	269
INTENTION-REVEALING INTERFACES	270
SIDE-EFFECT FREE FUNCTIONS	271
ASSERTIONS	272
CONCEPTUAL CONTOURS	273
STANDALONE CLASSES	274
CLOSURE OF OPERATIONS	275
MODEL INTEGRITY AND CONTEXT	276
BOUNDED CONTEXT	277
CONTINUOUS INTEGRATION	278
STRATEGIC CONTEXT MAP	279
CONTEXTUAL MAP	280
SHARED KERNEL	281
CUSTOMER-SUPPLIER TEAMS	282
CONFORMIST	283
ANTICORRUPTION LAYER	284
SEPARATE WAYS	285
OPEN HOST SERVICE	286
PUBLISHED LANGUAGE	287
DISTILLATION	288
CORE DOMAIN	289
GENERIC SUBDOMAINS	290
DOMAIN VISION STATEMENT	291
HIGHLIGHTED CORE	292
COHESIVE MECHANISMS	293

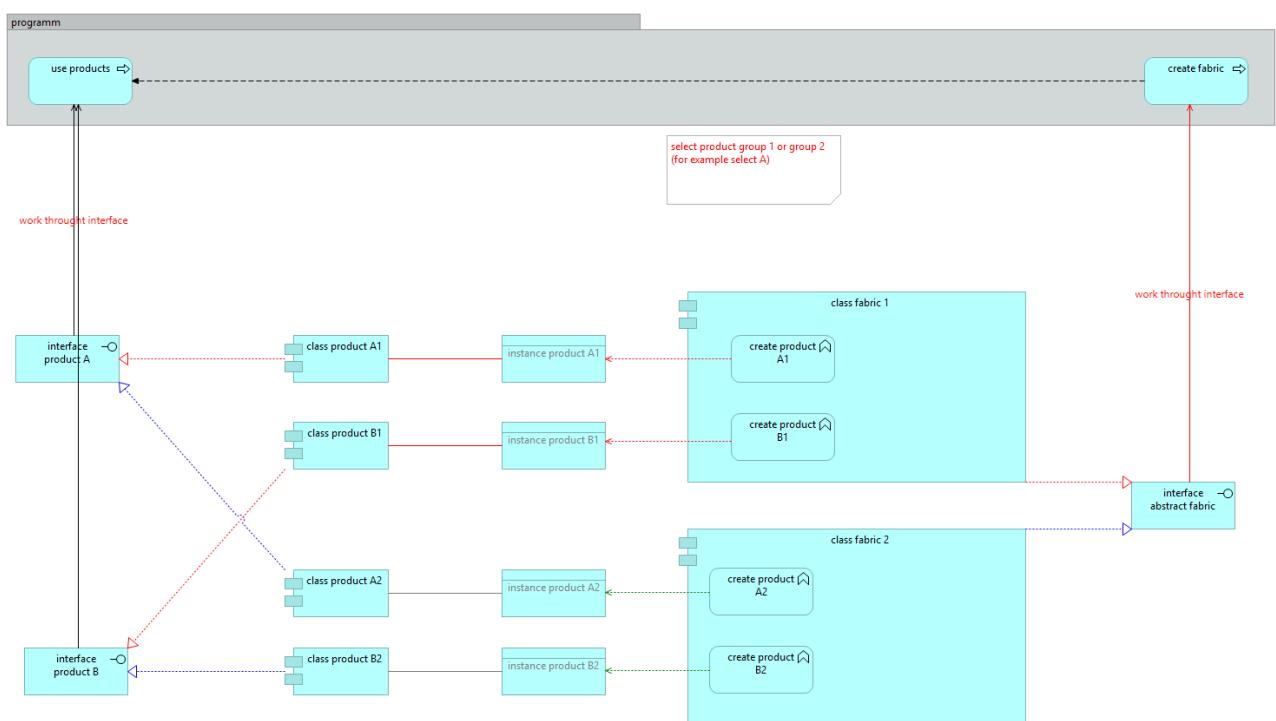
SEGREGATED CORE	294
ABSTRACT CORE	295
LARGE-SCALE STRUCTURE	296
EVOLVING ORDER	297
SYSTEM METAPHOR	298
RESPONSIBILITY LAYERS	299
KNOWLEDGE LEVEL	303
PLUGGABLE COMPONENT FRAMEWORK	304
ADDITIONAL PATTERNS	305
TYPES OF CONSISTENCY	306
EVENT SOURCING	307
EVENT PROCESSOR	308
EVENT DISPATCHER	309
INTERNAL DOMAIN EVENTS	310
EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS	311
STATIC DOMAIN EVENTS CLASS	312
ONE SUBDOMAIN PER BOUNDED CONTEXT	313
THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS	314
THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS	315
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE	316
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES	317
INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS	318
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE	319
DEPENDENCY INJECTION	320
DEPENDENCY INVERSION	322
INVERSION OF CONTROL	323
SERVICE LOCATOR	324
CQRS	325
CQS	326
WRAP LOW-LEVEL EXCEPTIONS	327
EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR	328
INTERFACE SEGREGATION	329
5 CLEAN ARCHITECTURE	330
PAYROLL DEMO APPLICATION STRUCTURE	331
STRUCTURE101 DIAGRAMS	337
ARCHITECTURAL STYLES	341

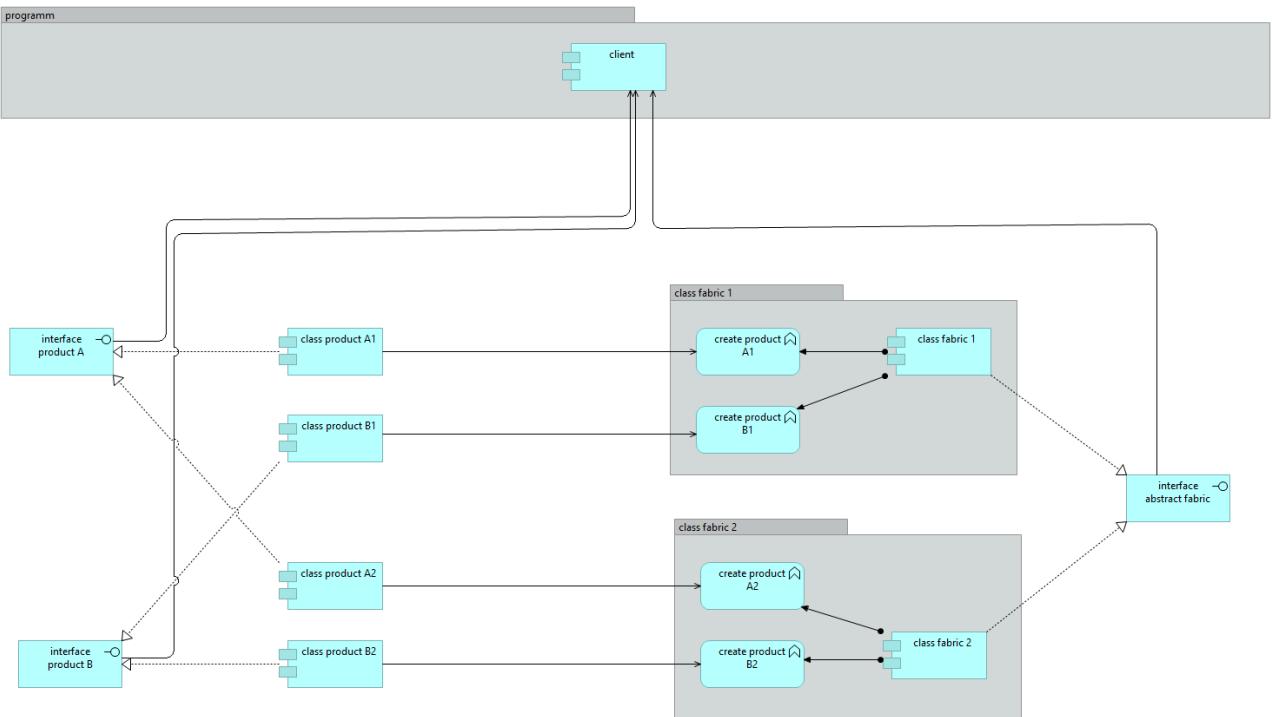
1 Design Patterns



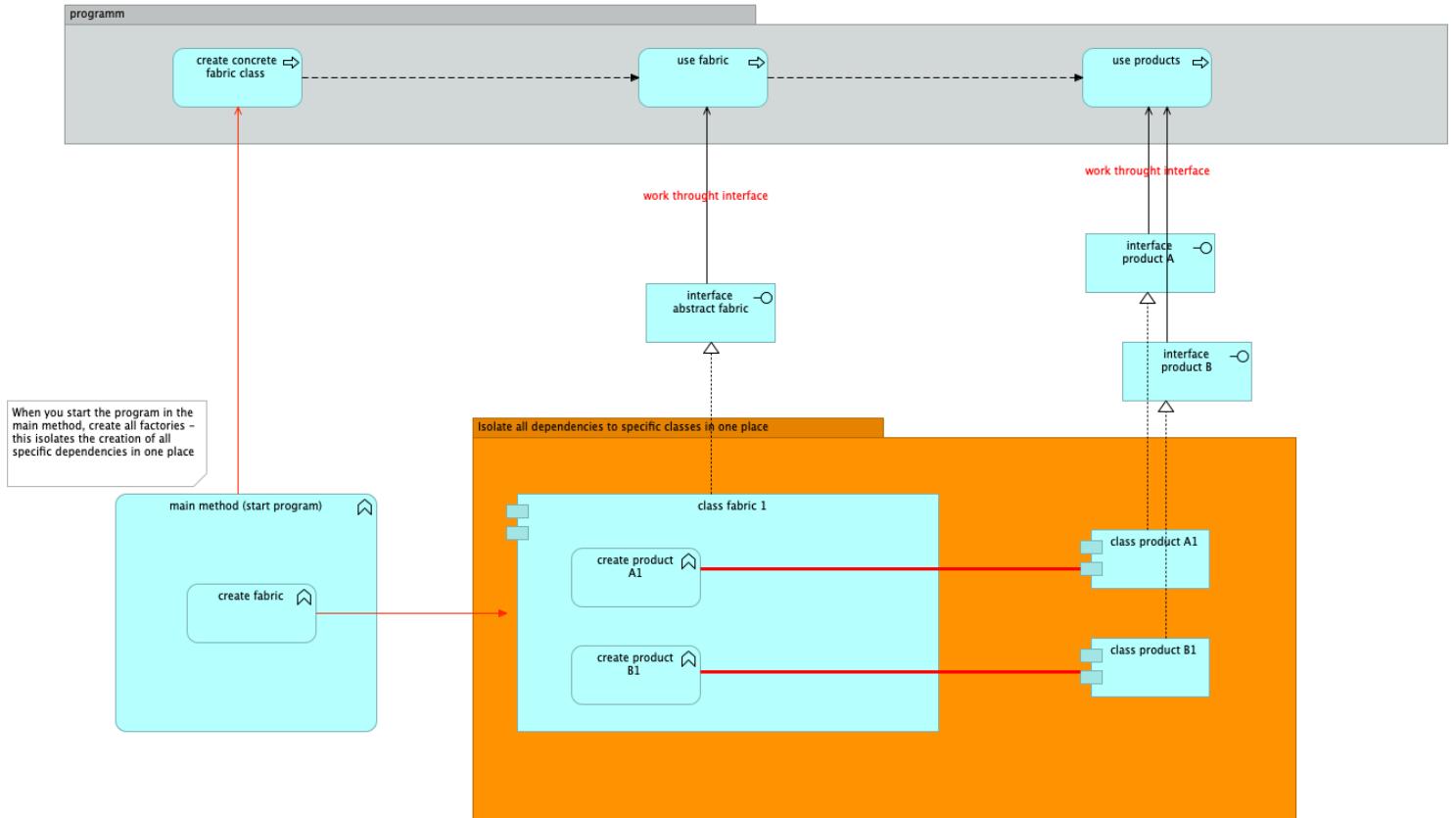
CREATIONAL PATTERNS

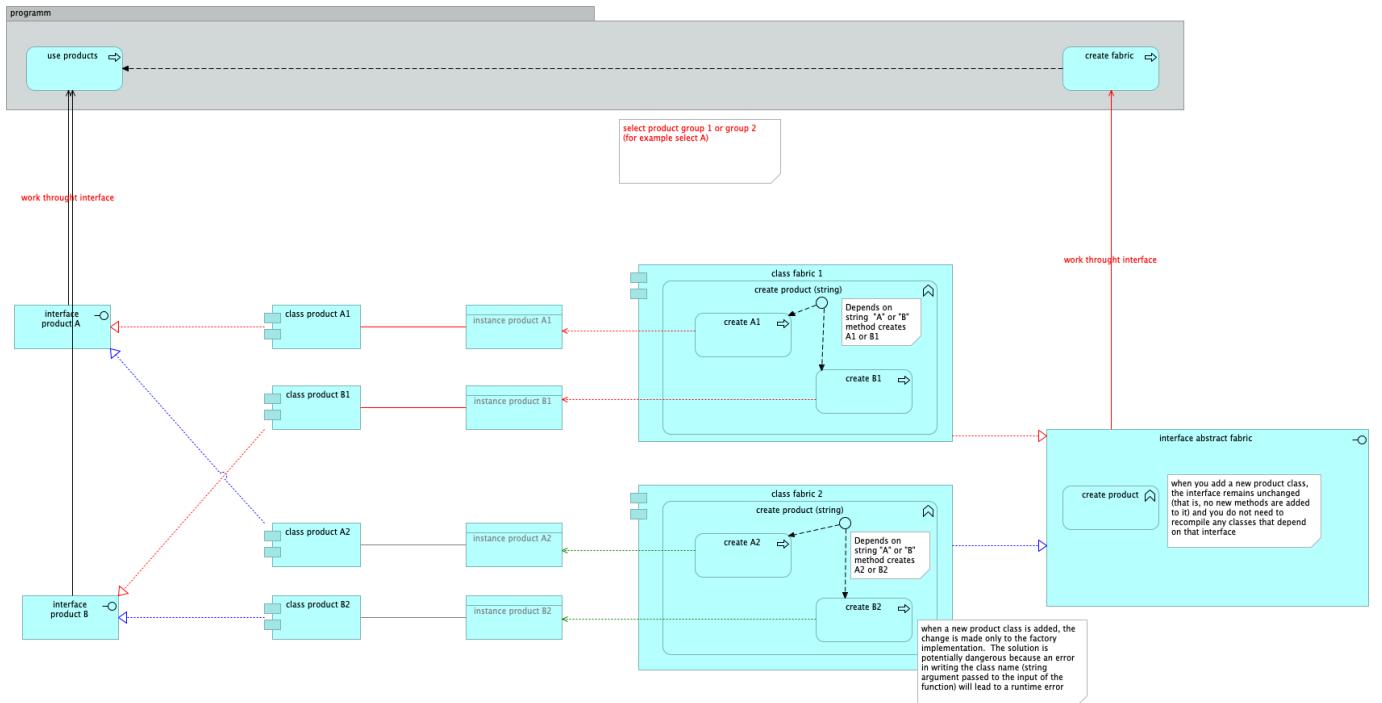
ABSTRACT FACTORY



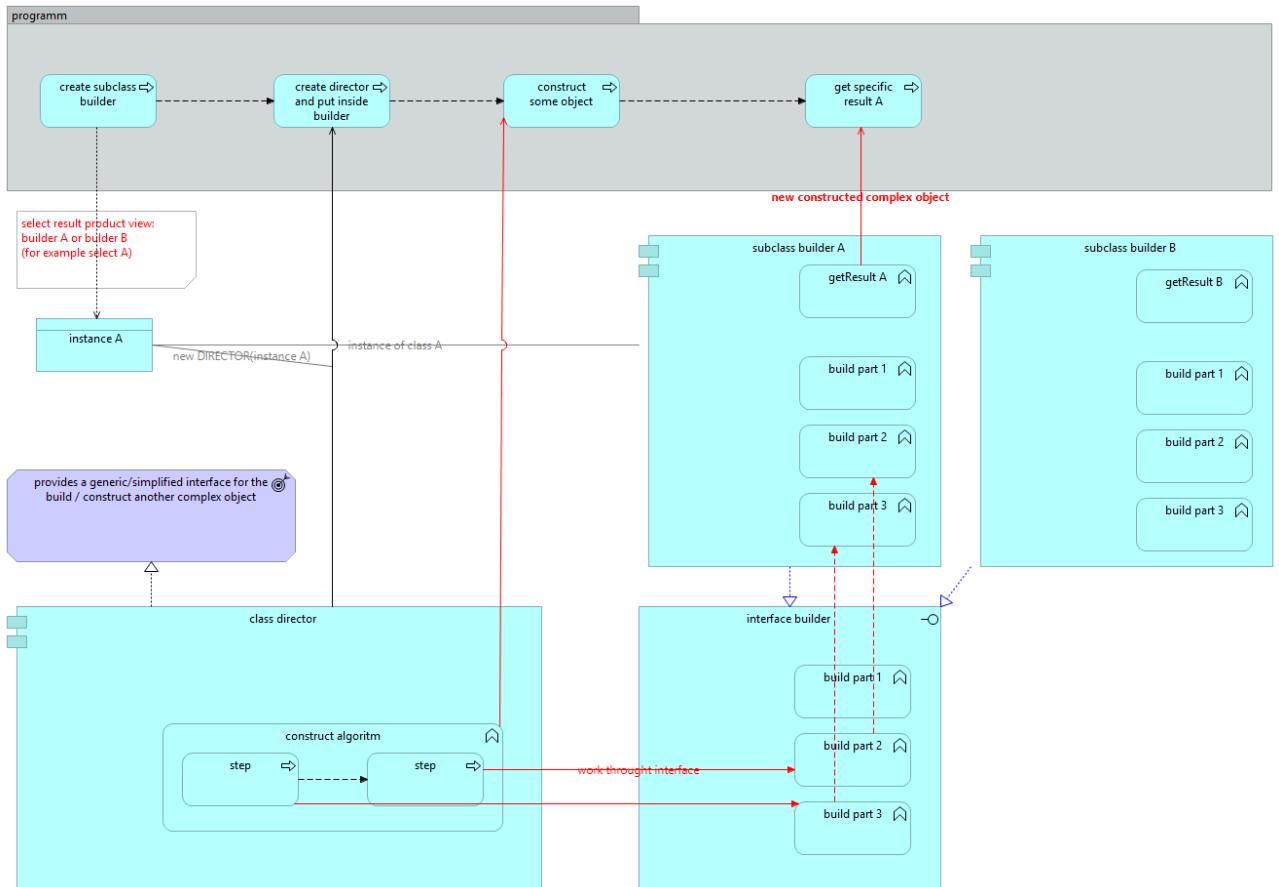


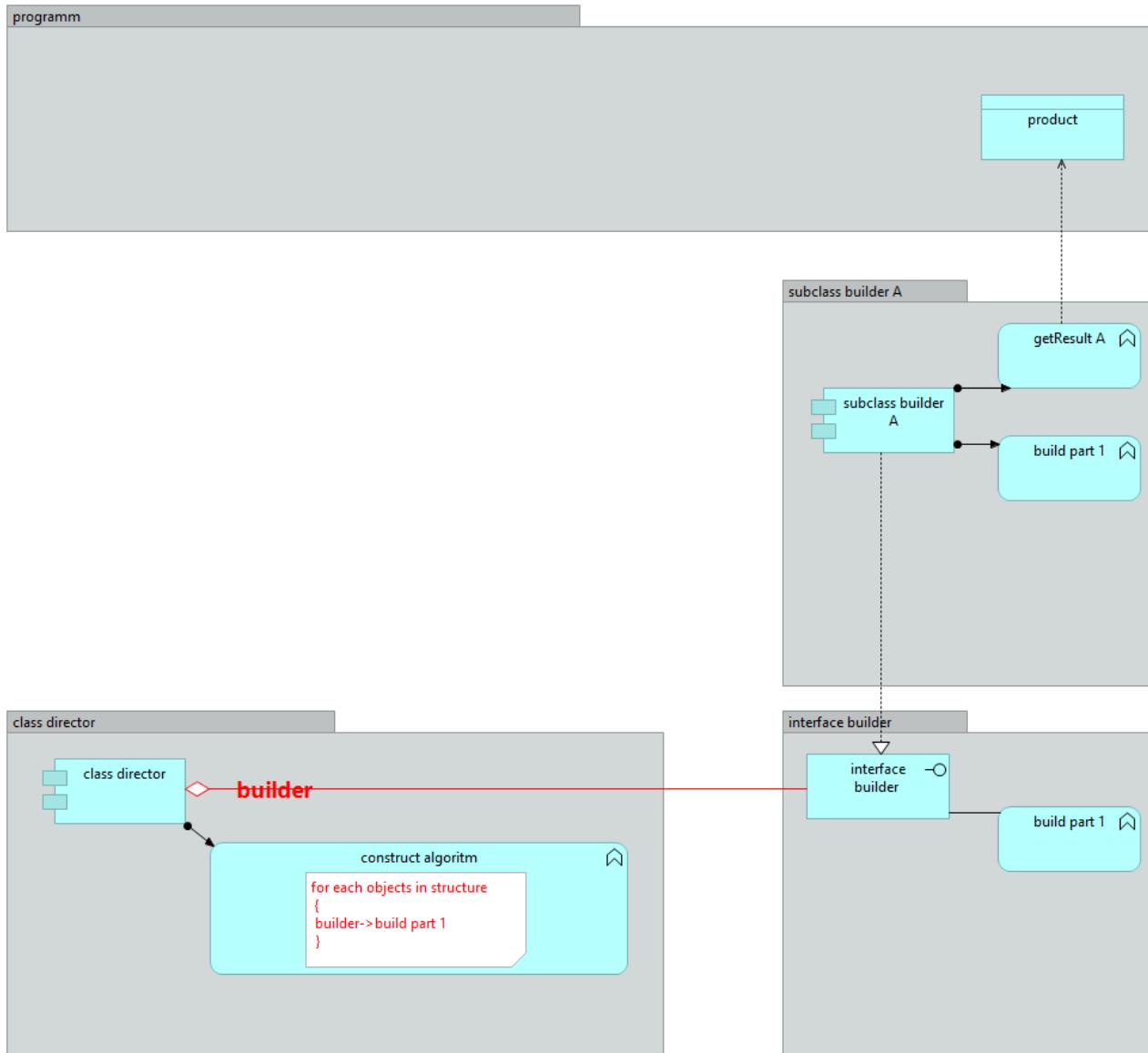
In the whole program (except for the factory inside and one factory creation) we work only with interfaces (and do not refer to specific classes)



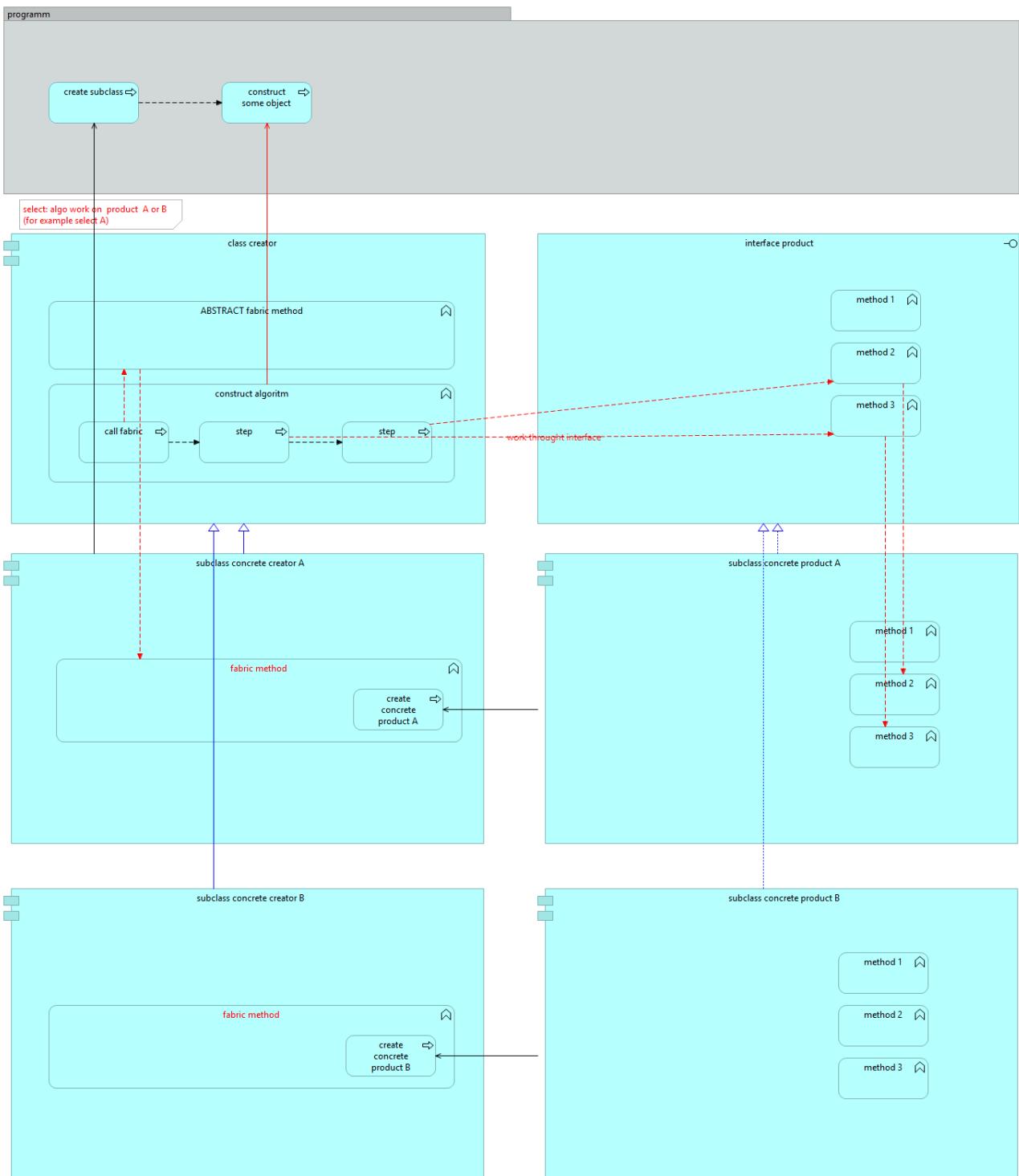


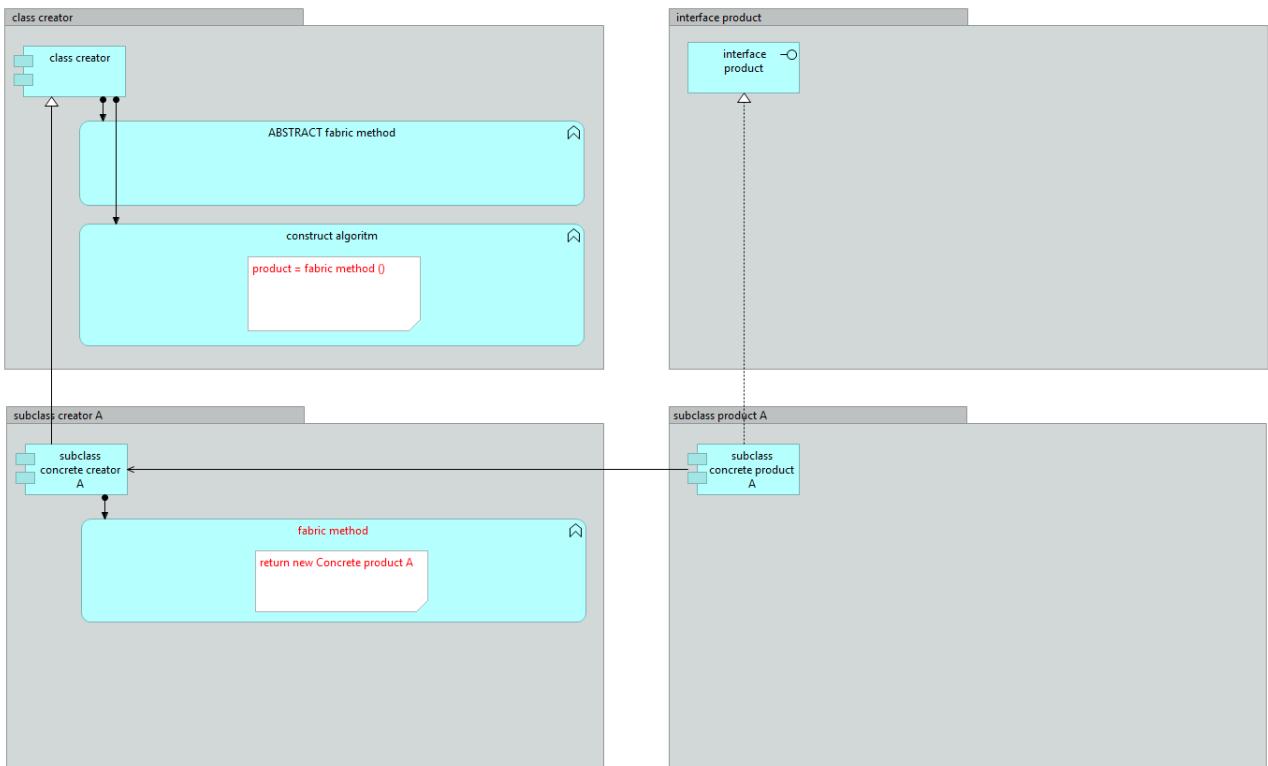
BUILDER



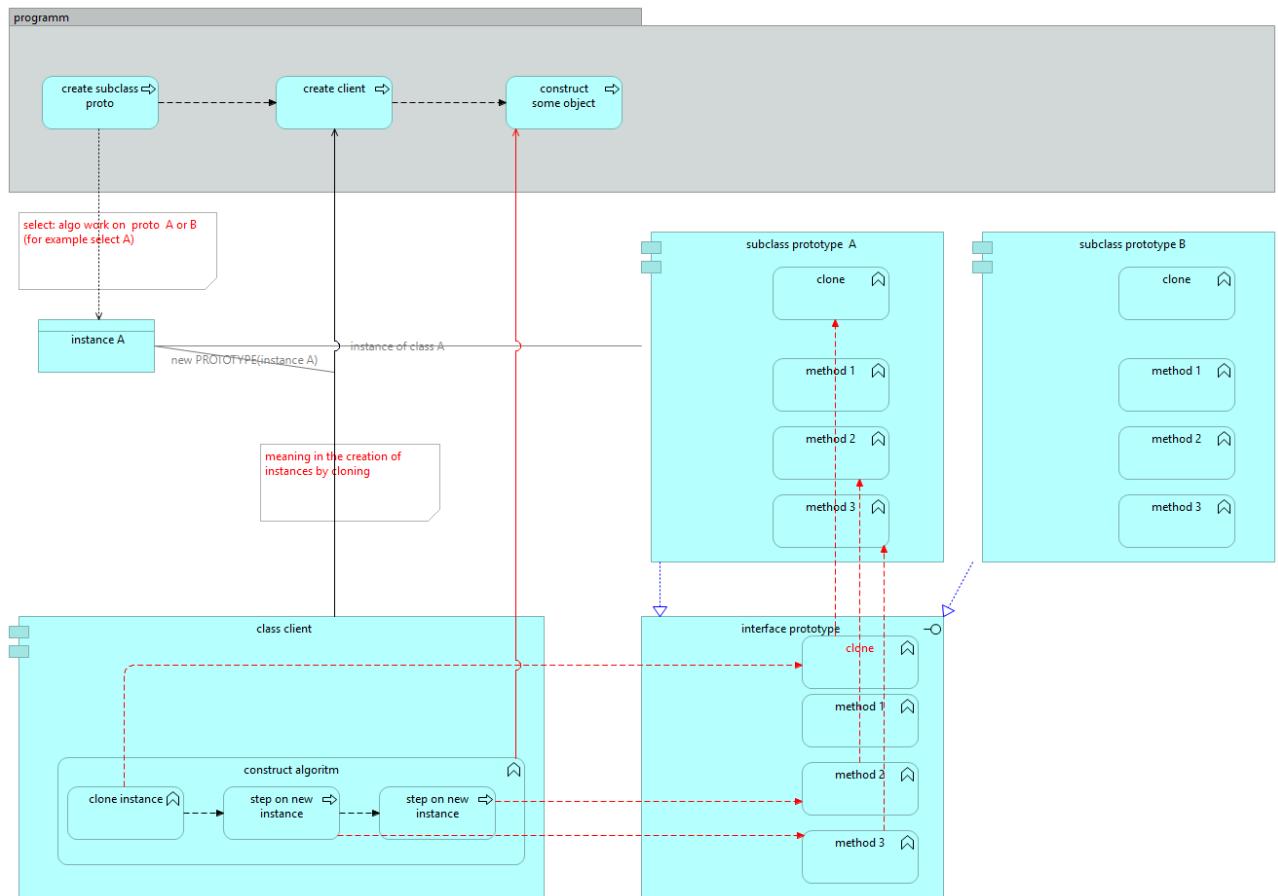


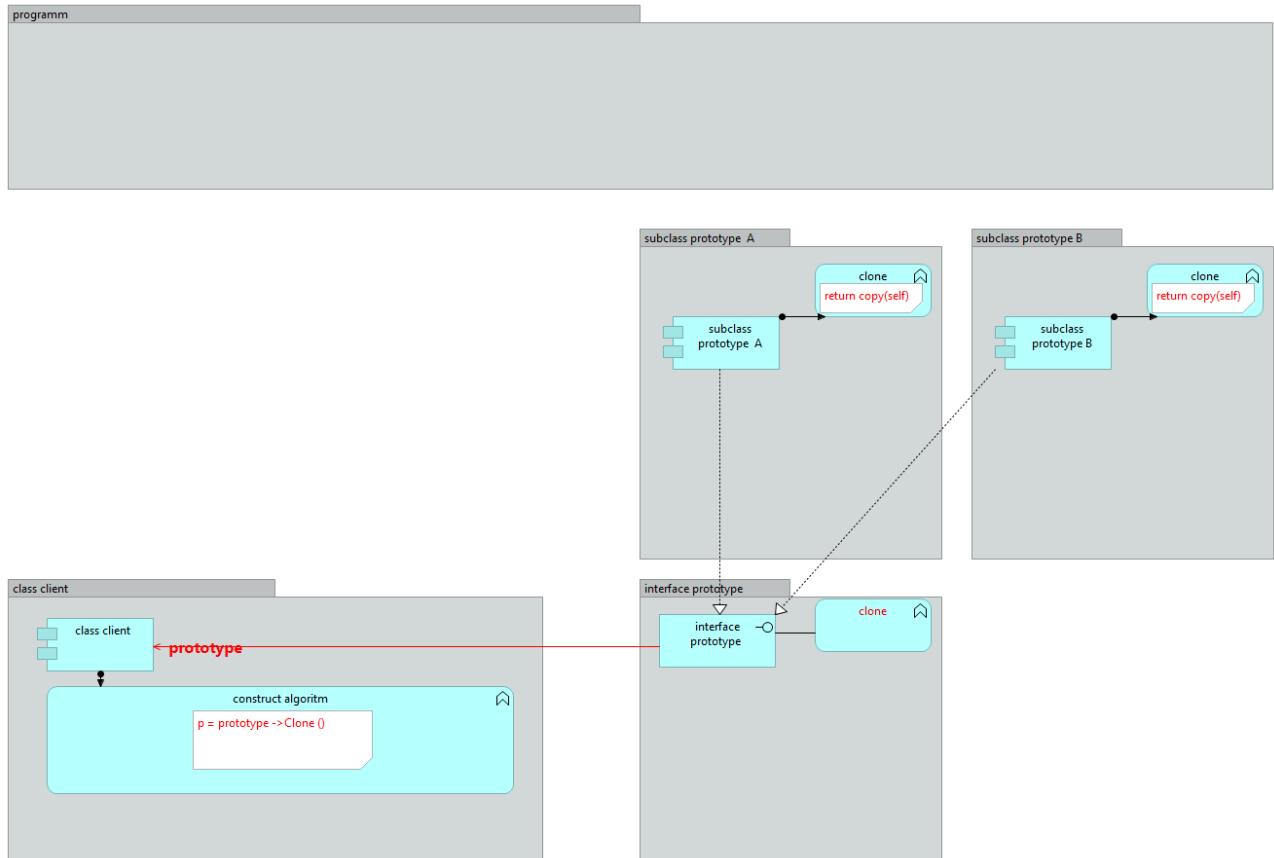
FACTORY METHOD



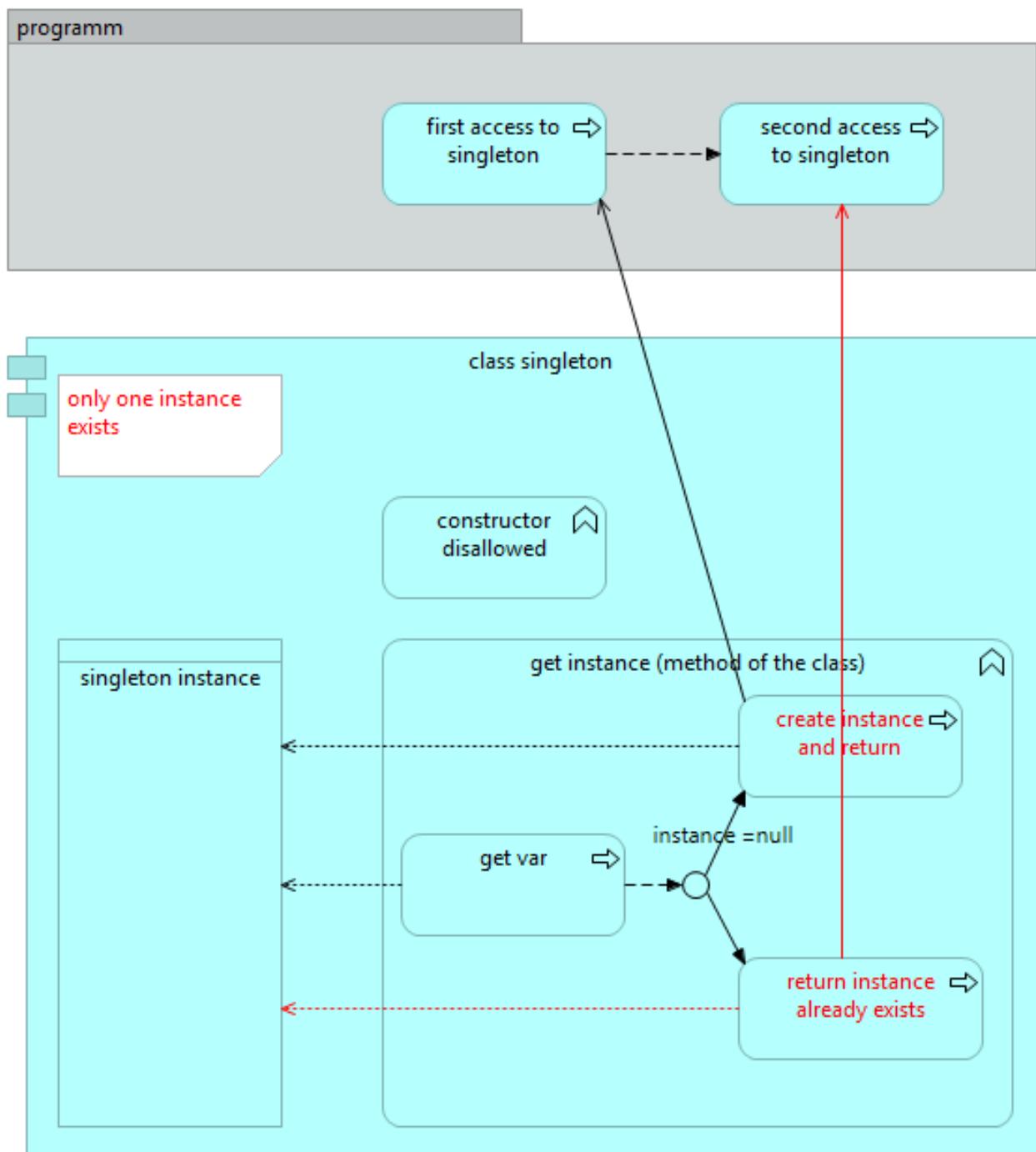


PROTOTYPE





SINGLETON



programm

class singleton

class singleton

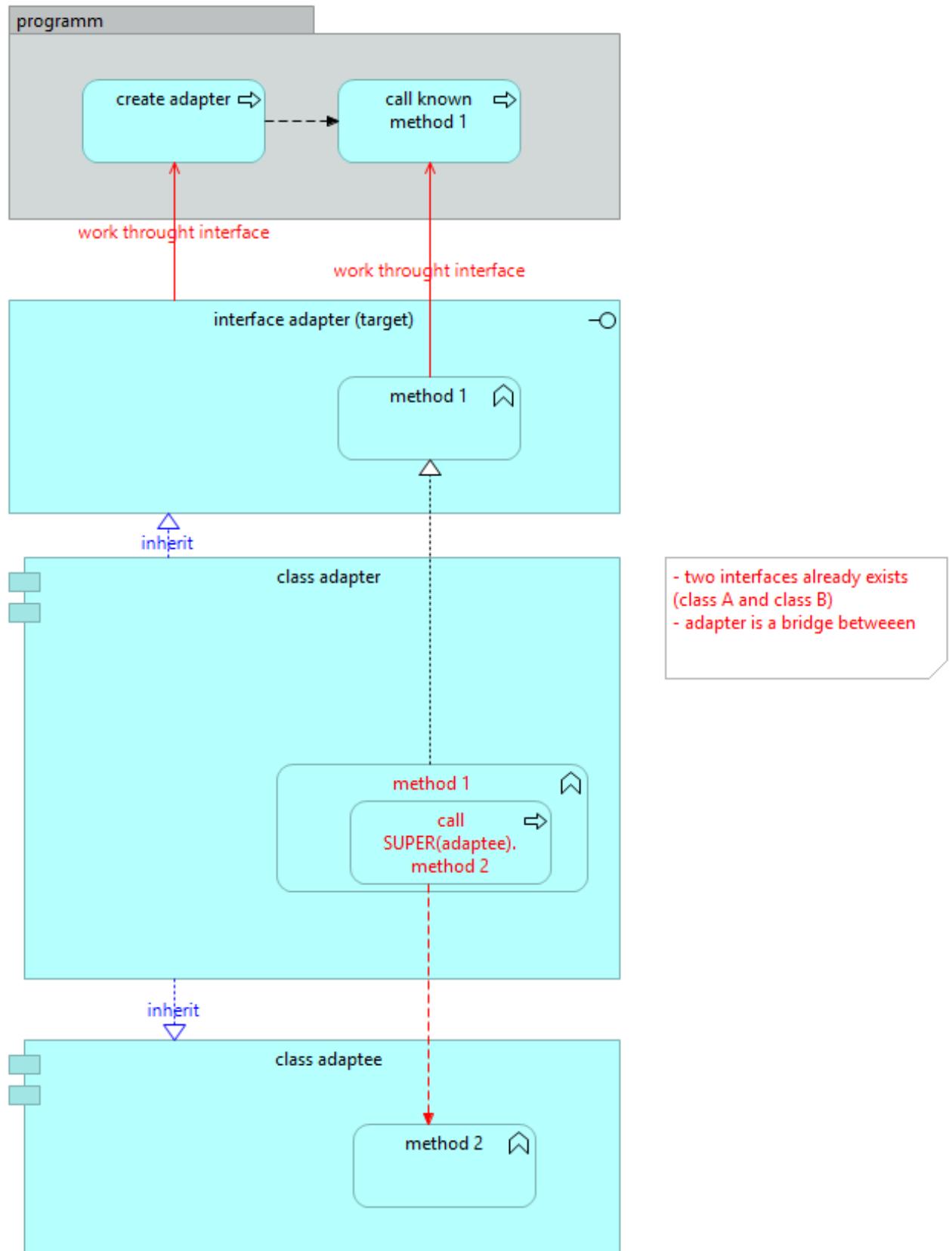
singleton instance

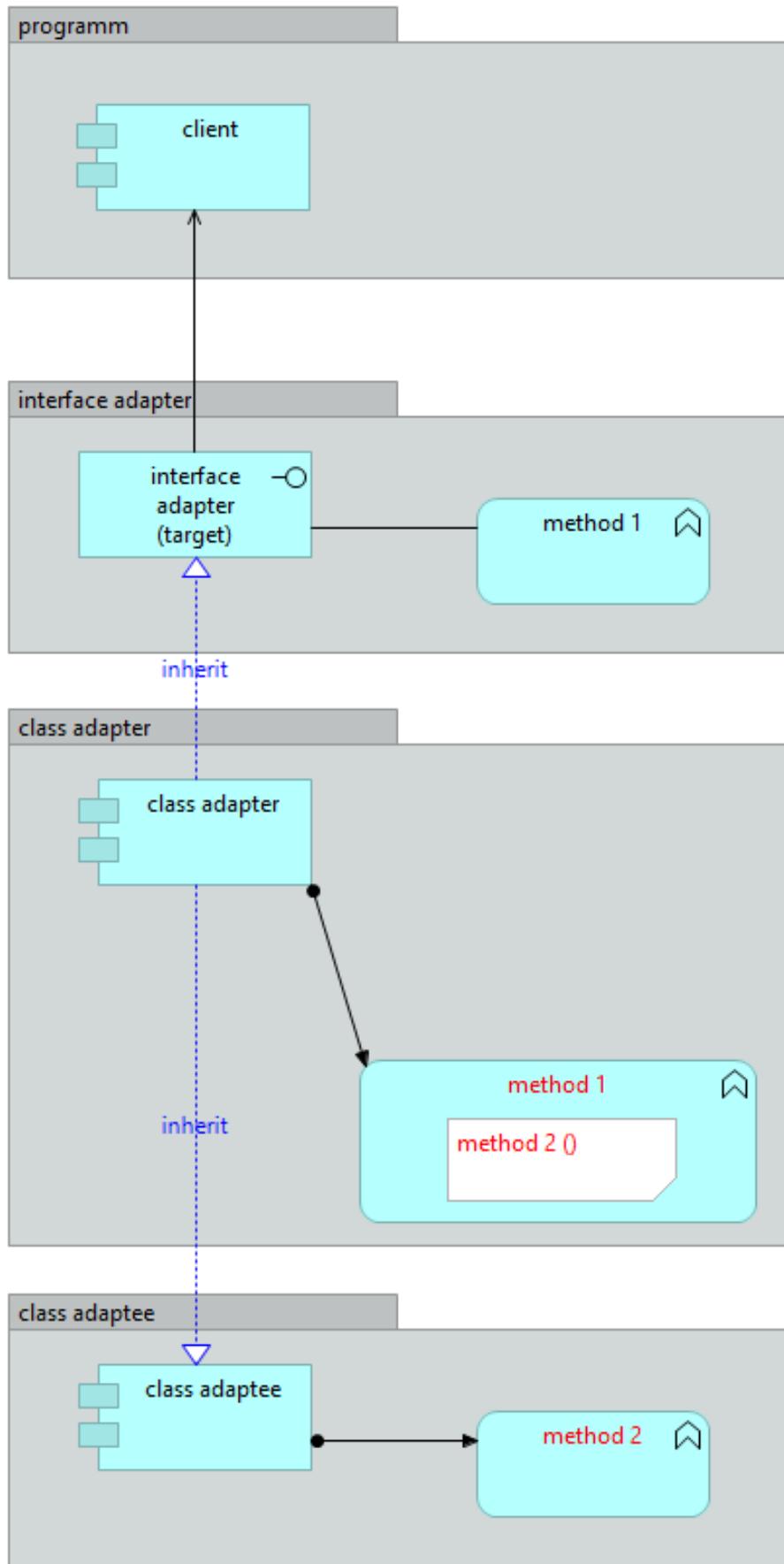
get instance (method of the class)

static method
{
return static singleton instance
}

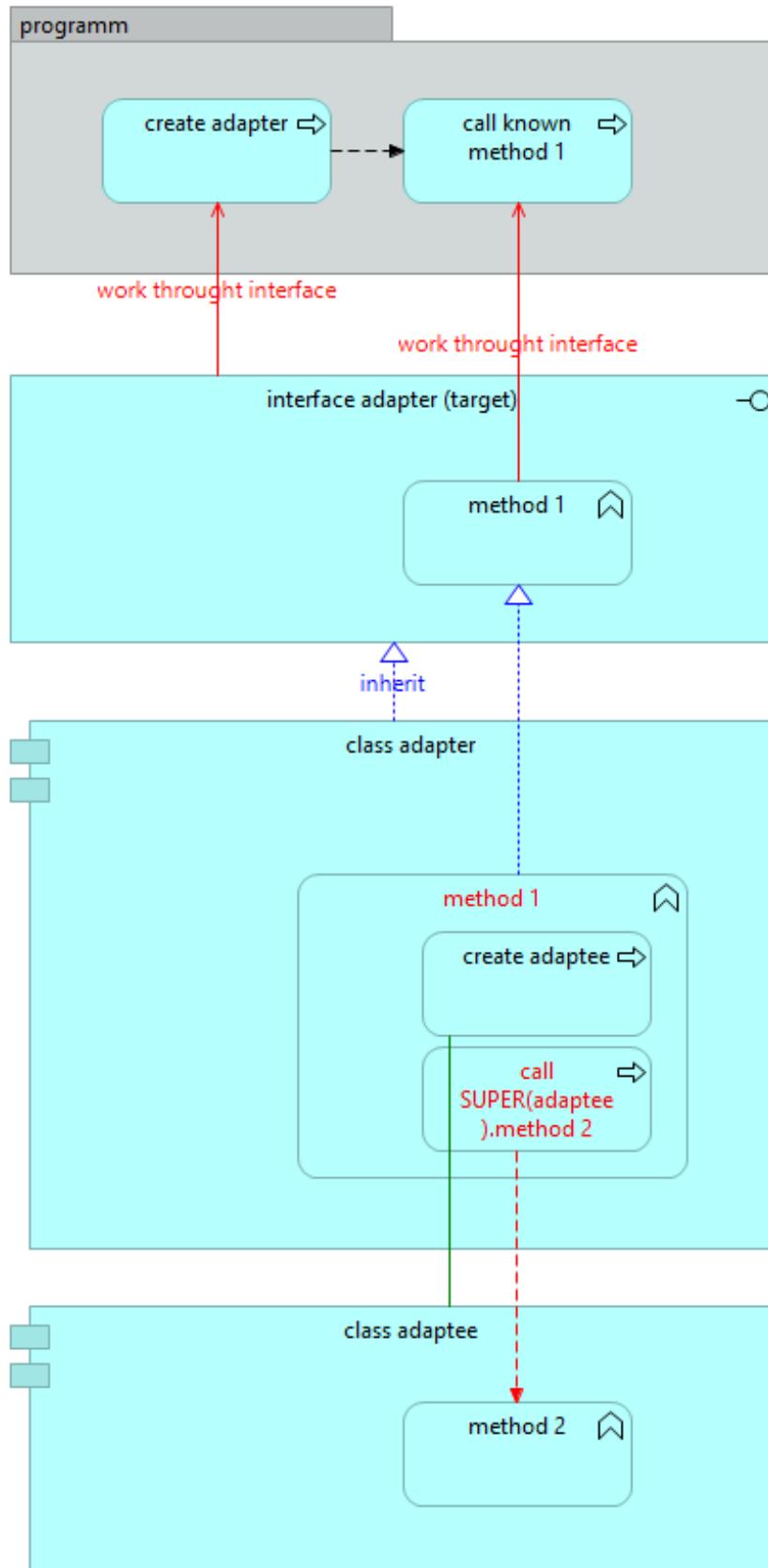
STRUCTURAL PATTERNS

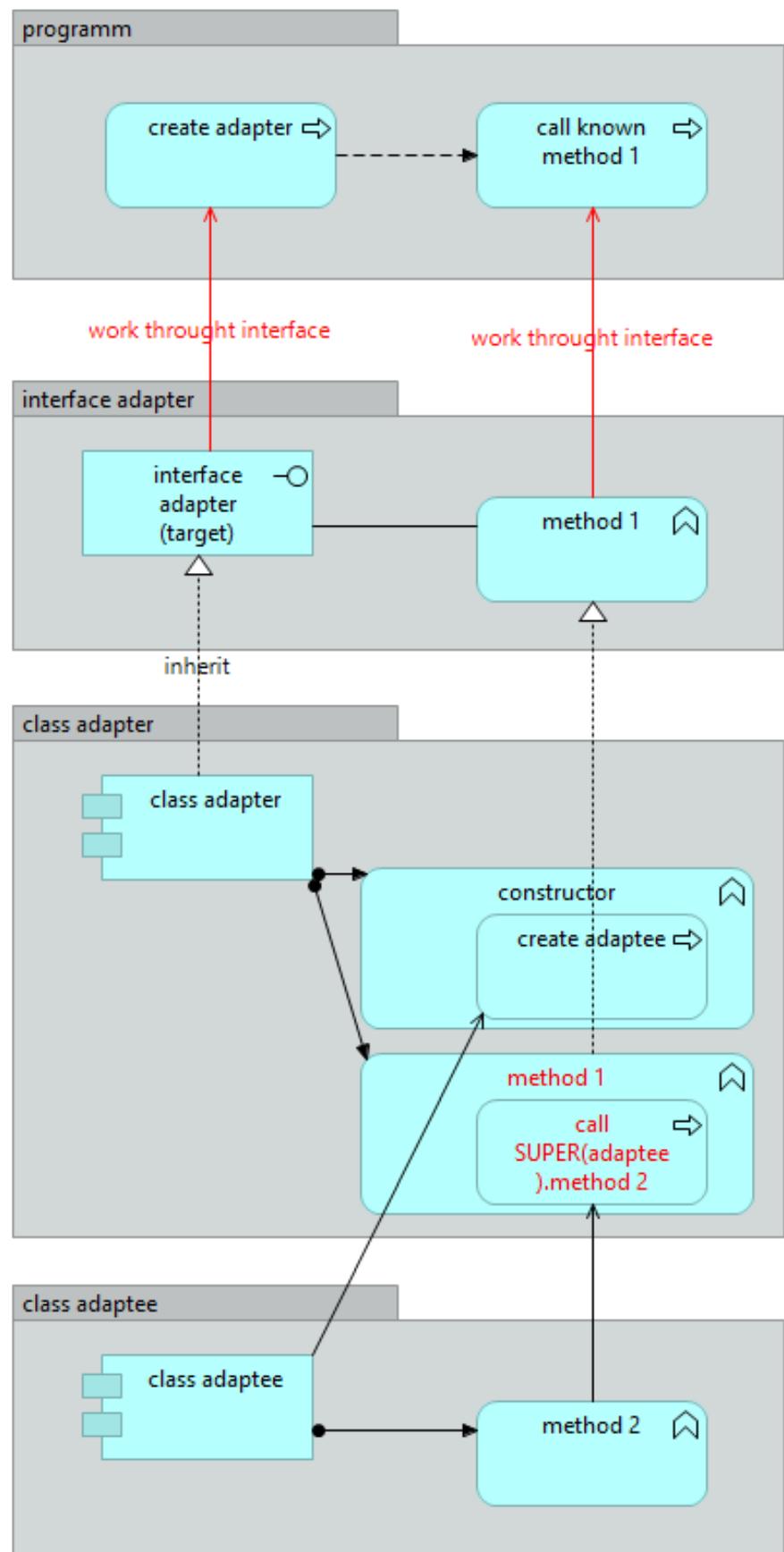
ADAPTER OF CLASS

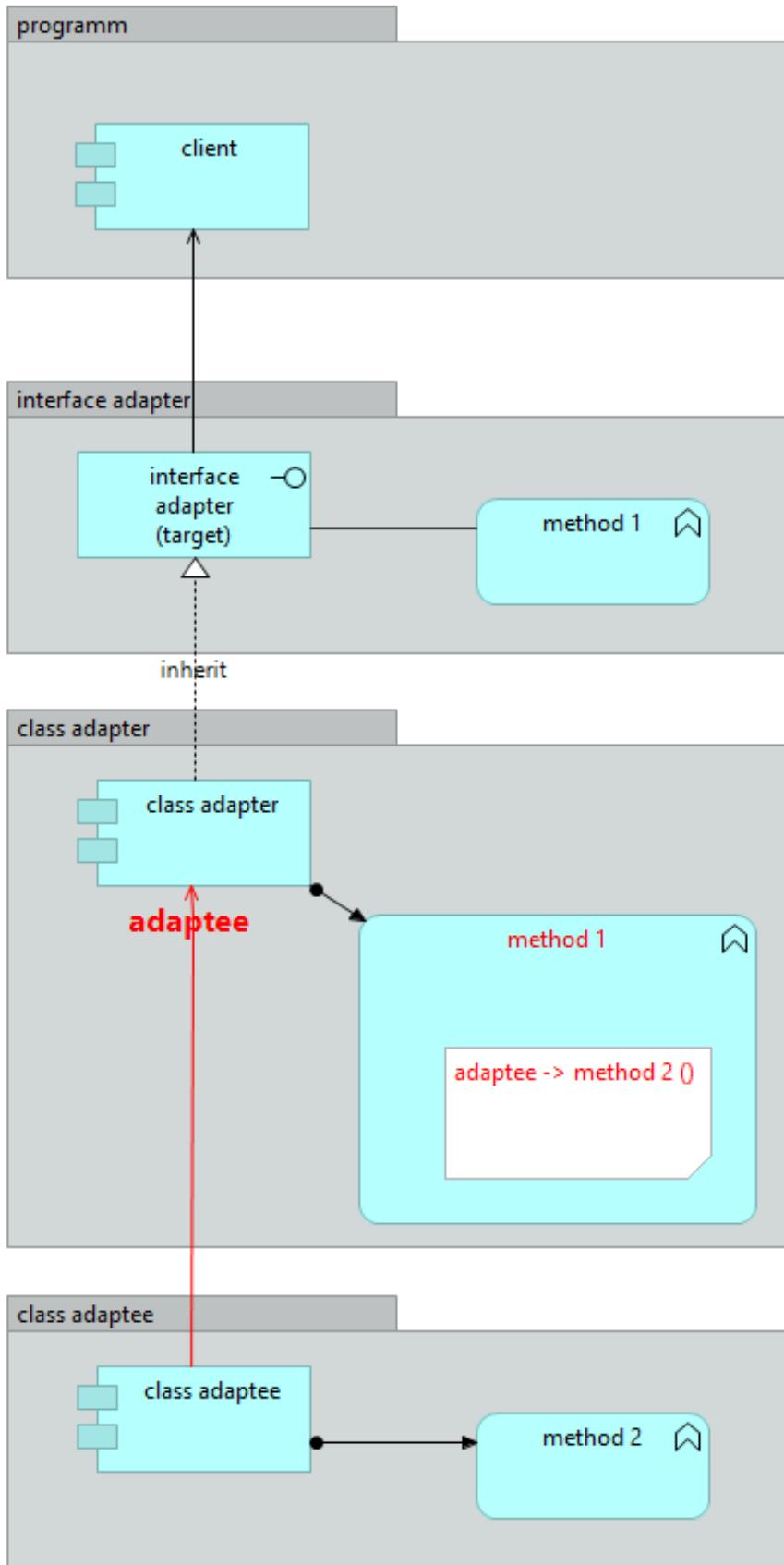




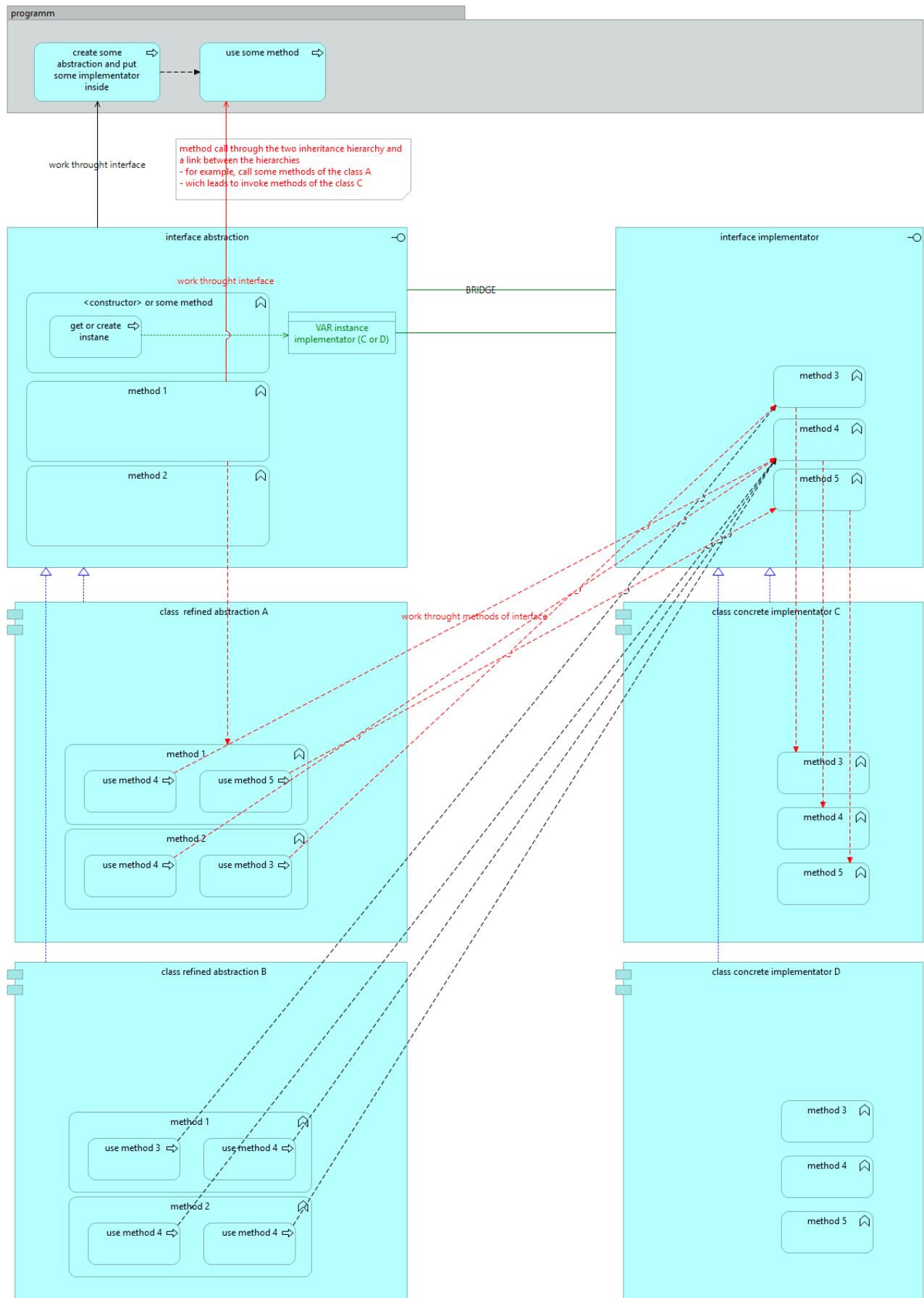
ADAPTER OF OBJECT

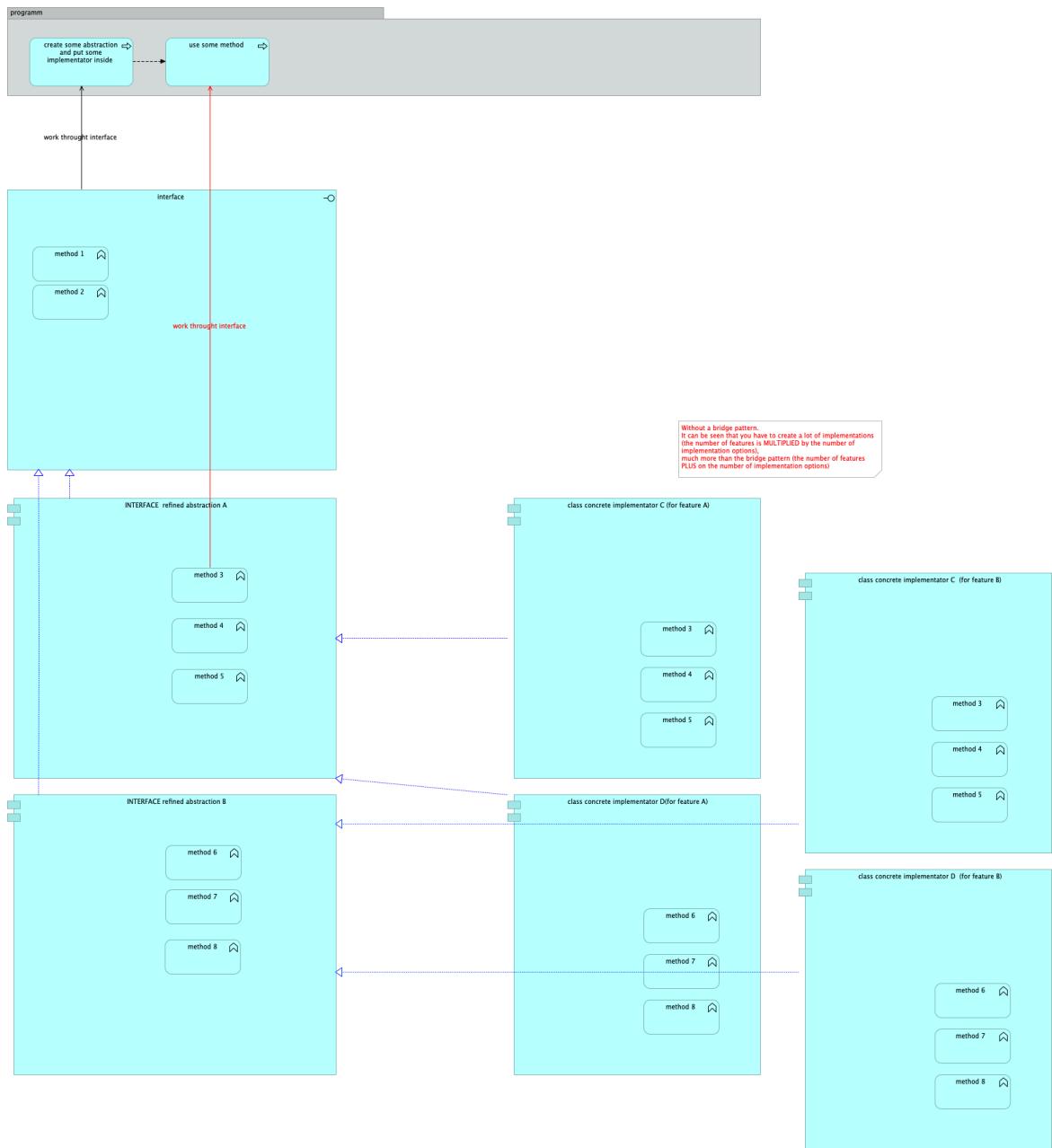


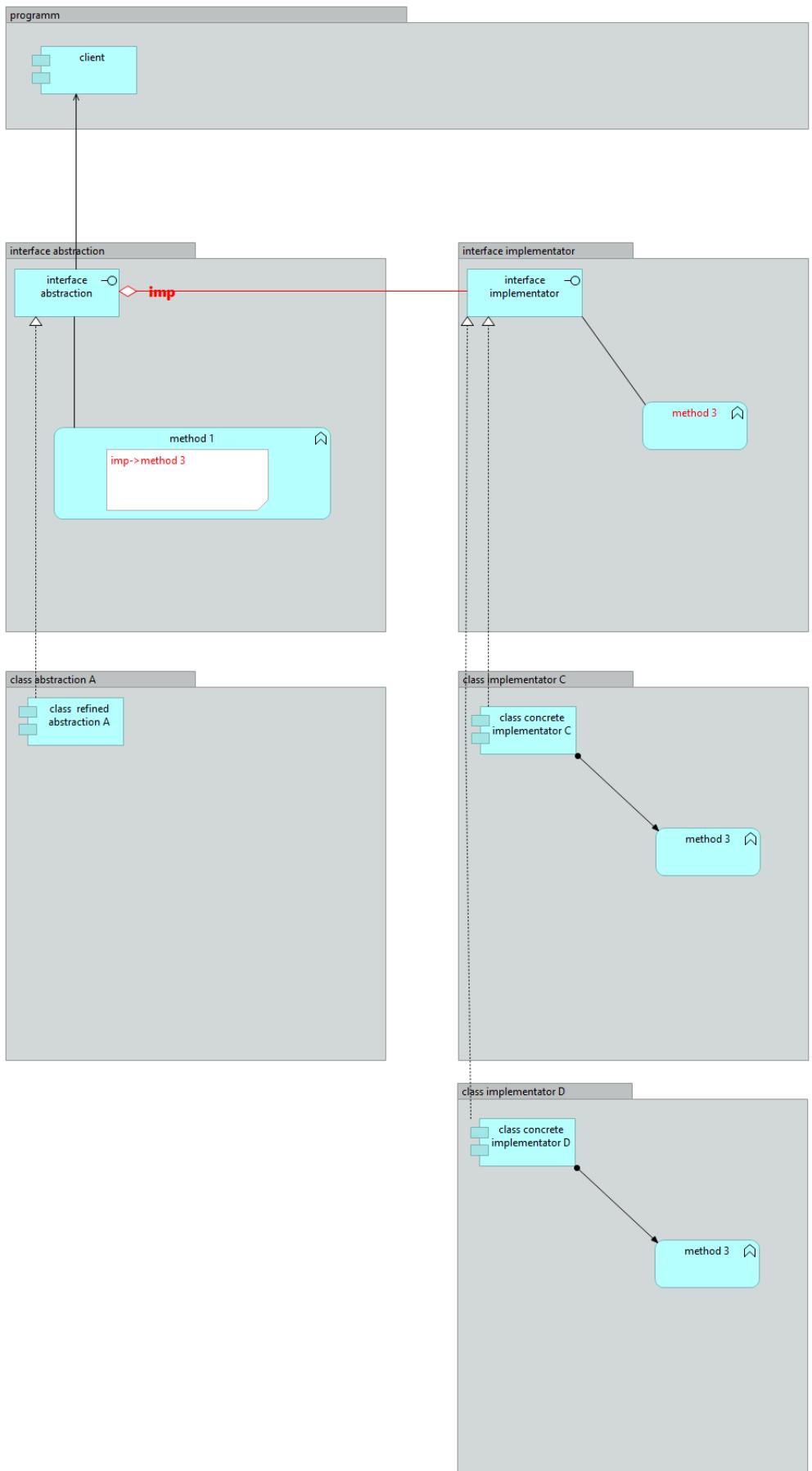


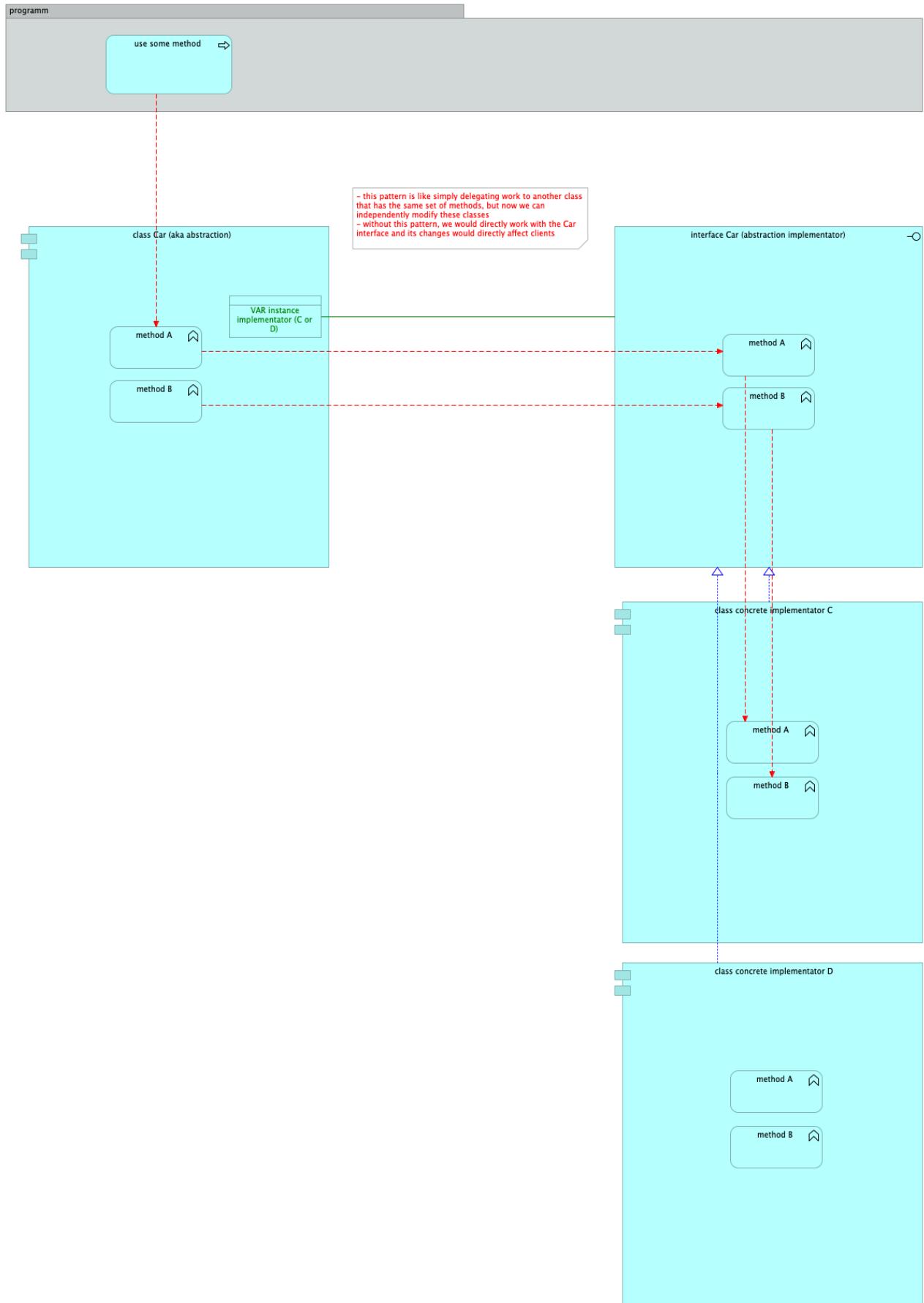


BRIDGE

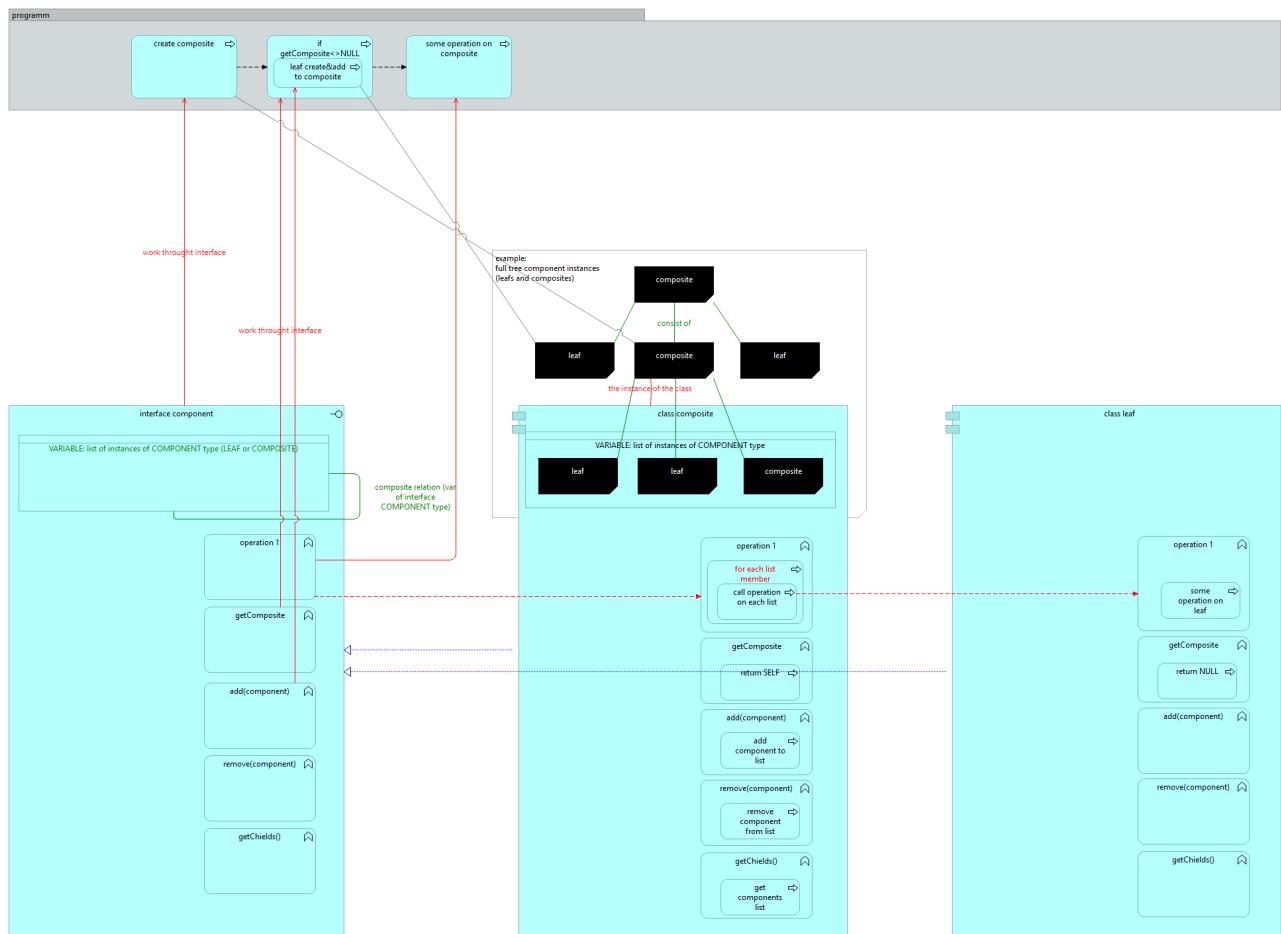


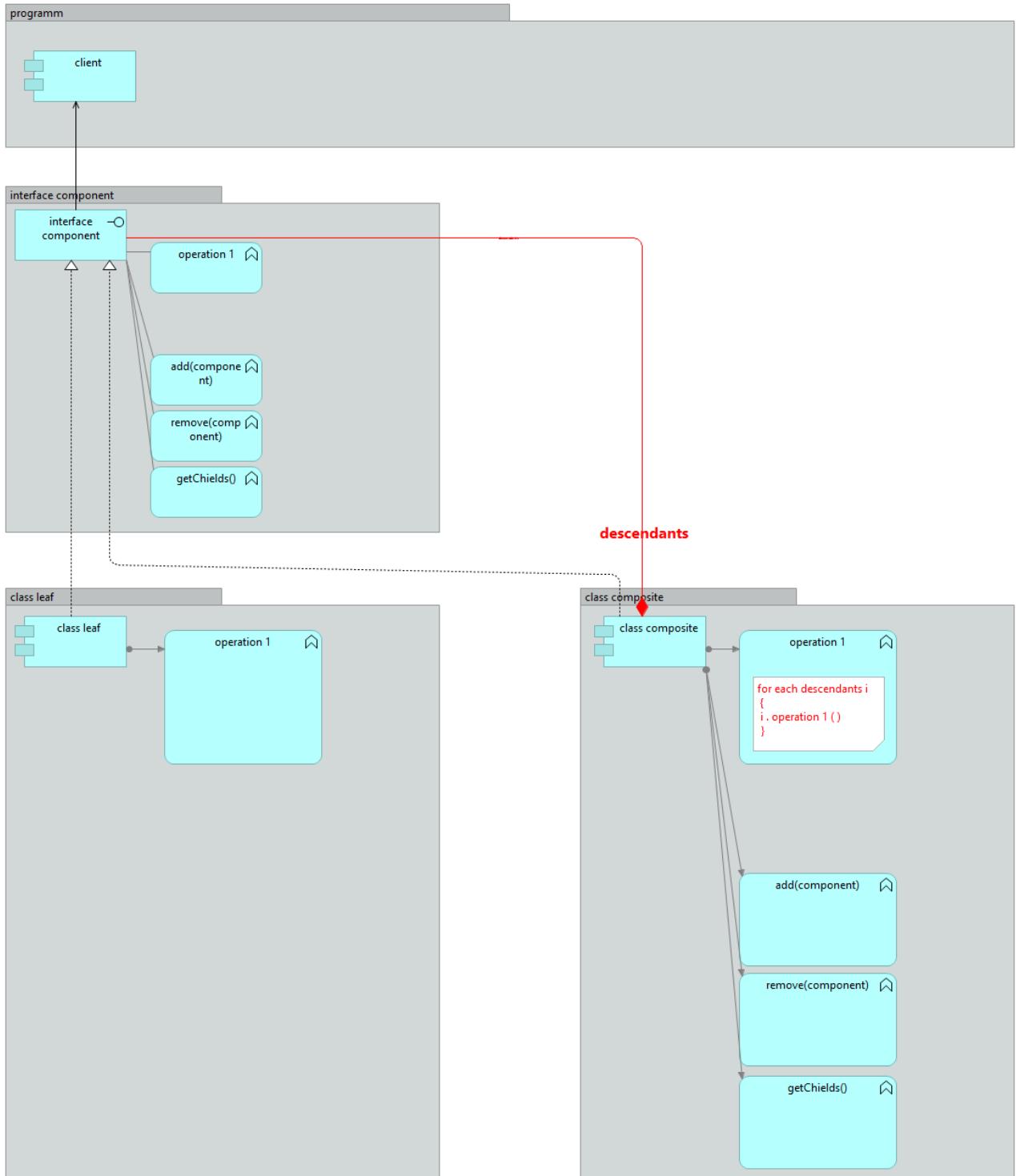




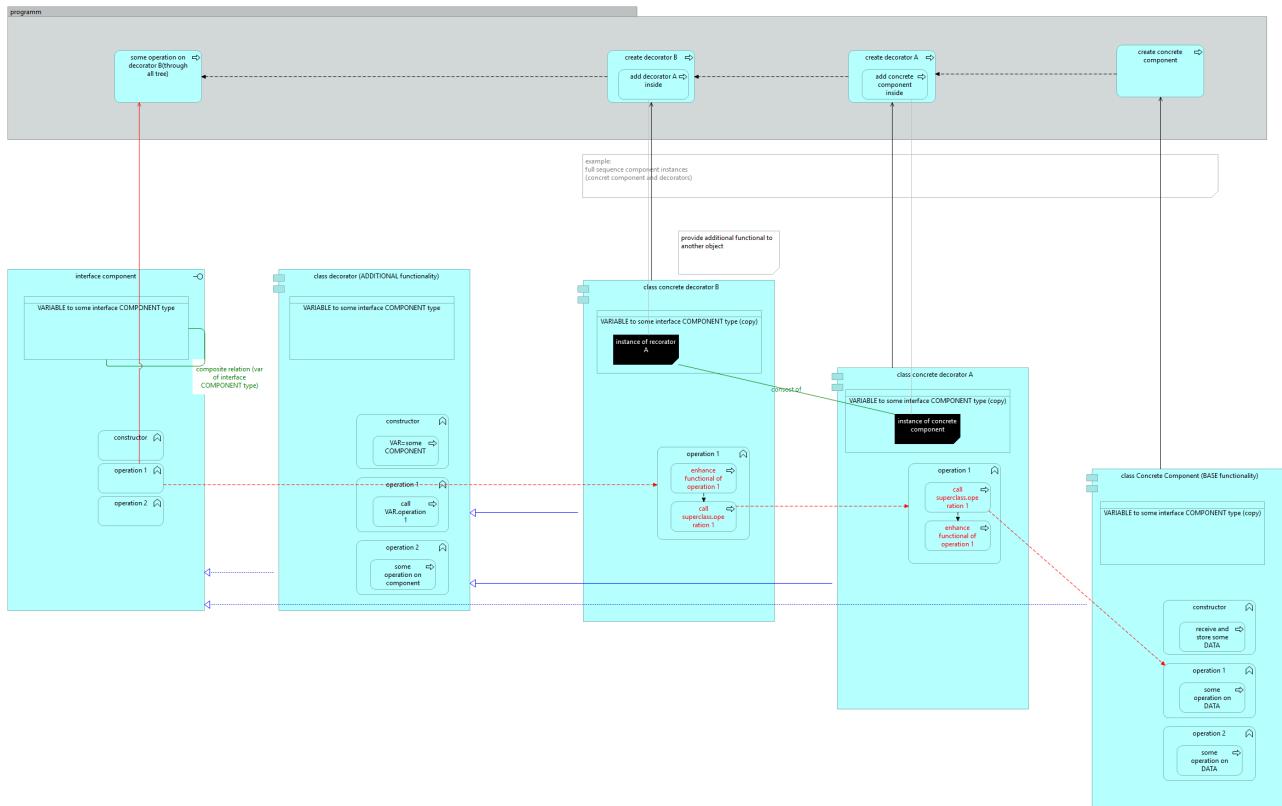


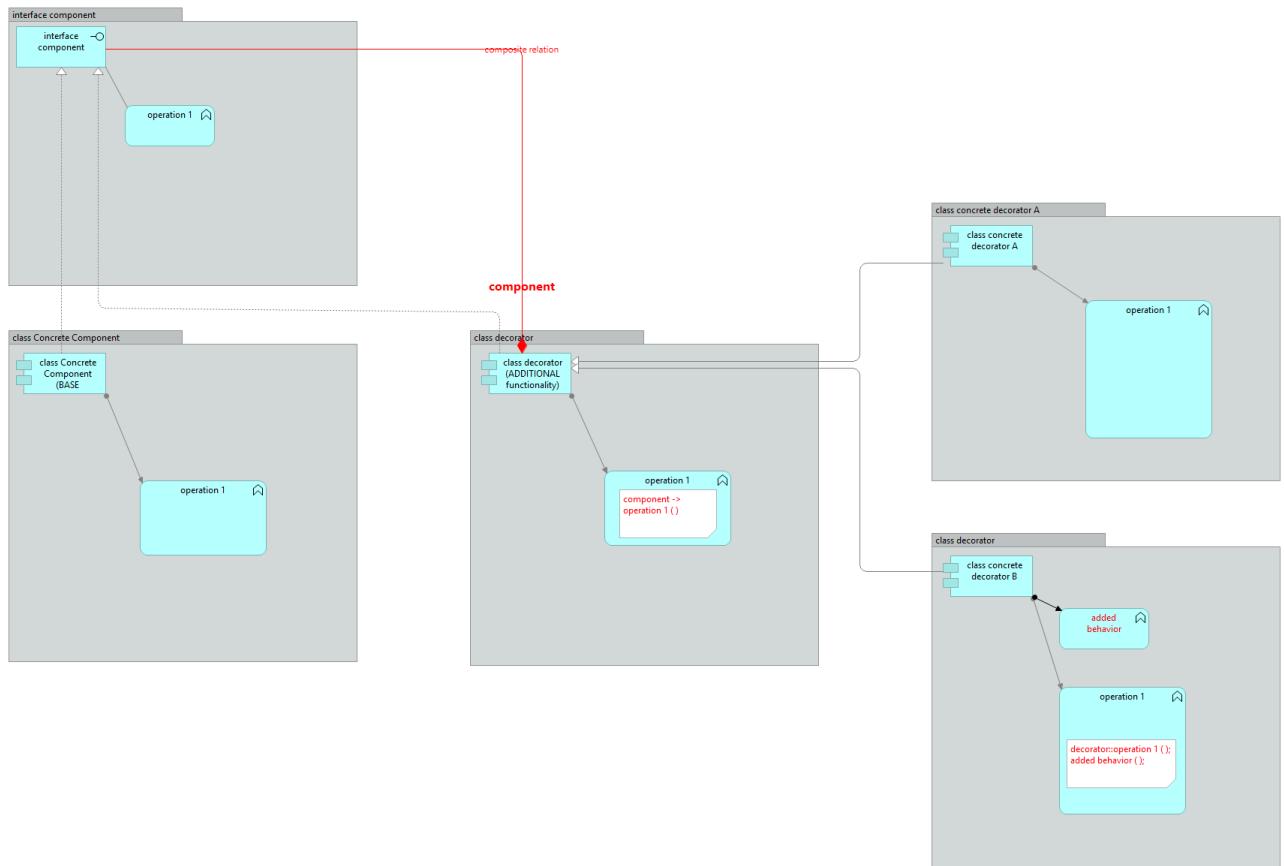
COMPOSITE



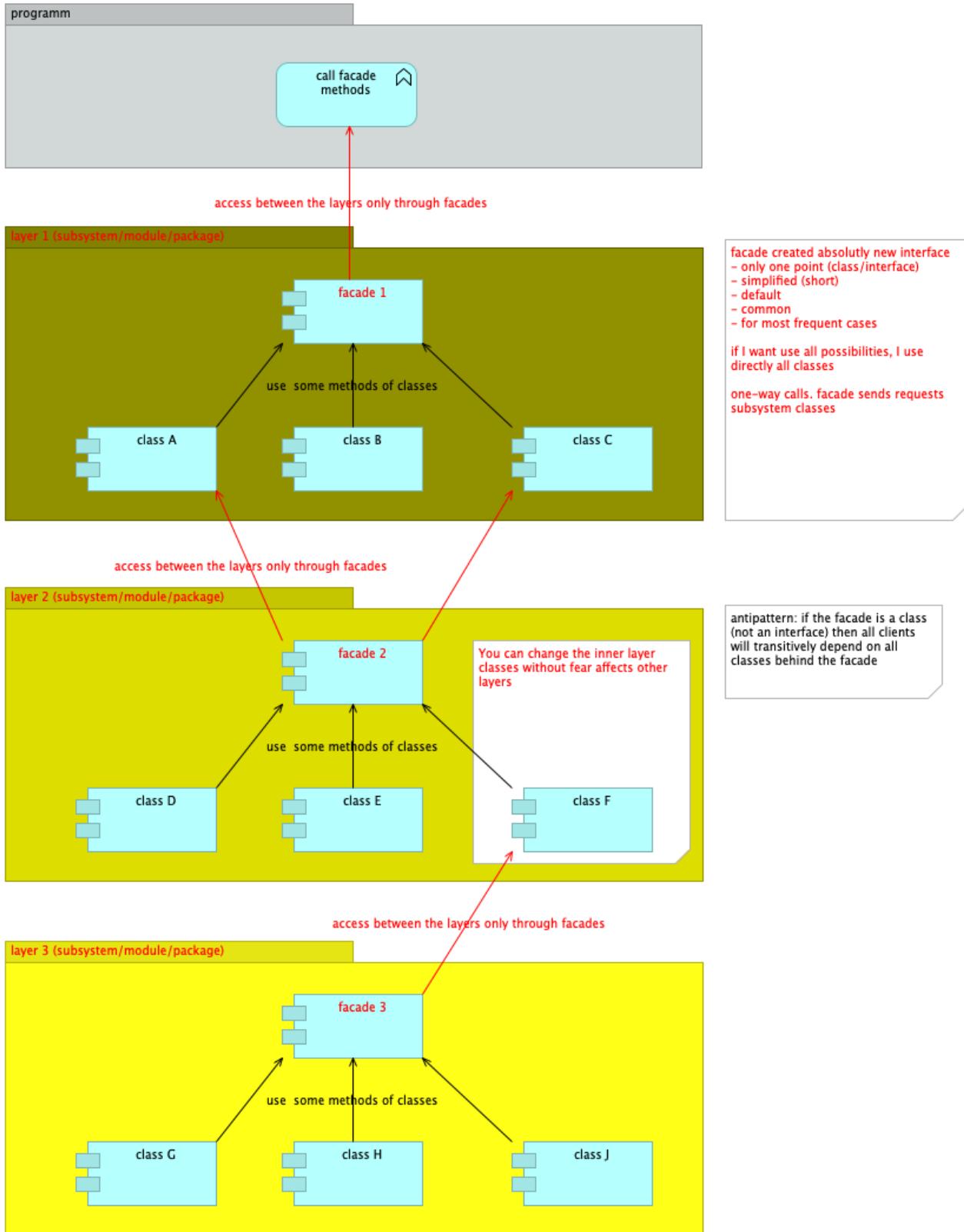


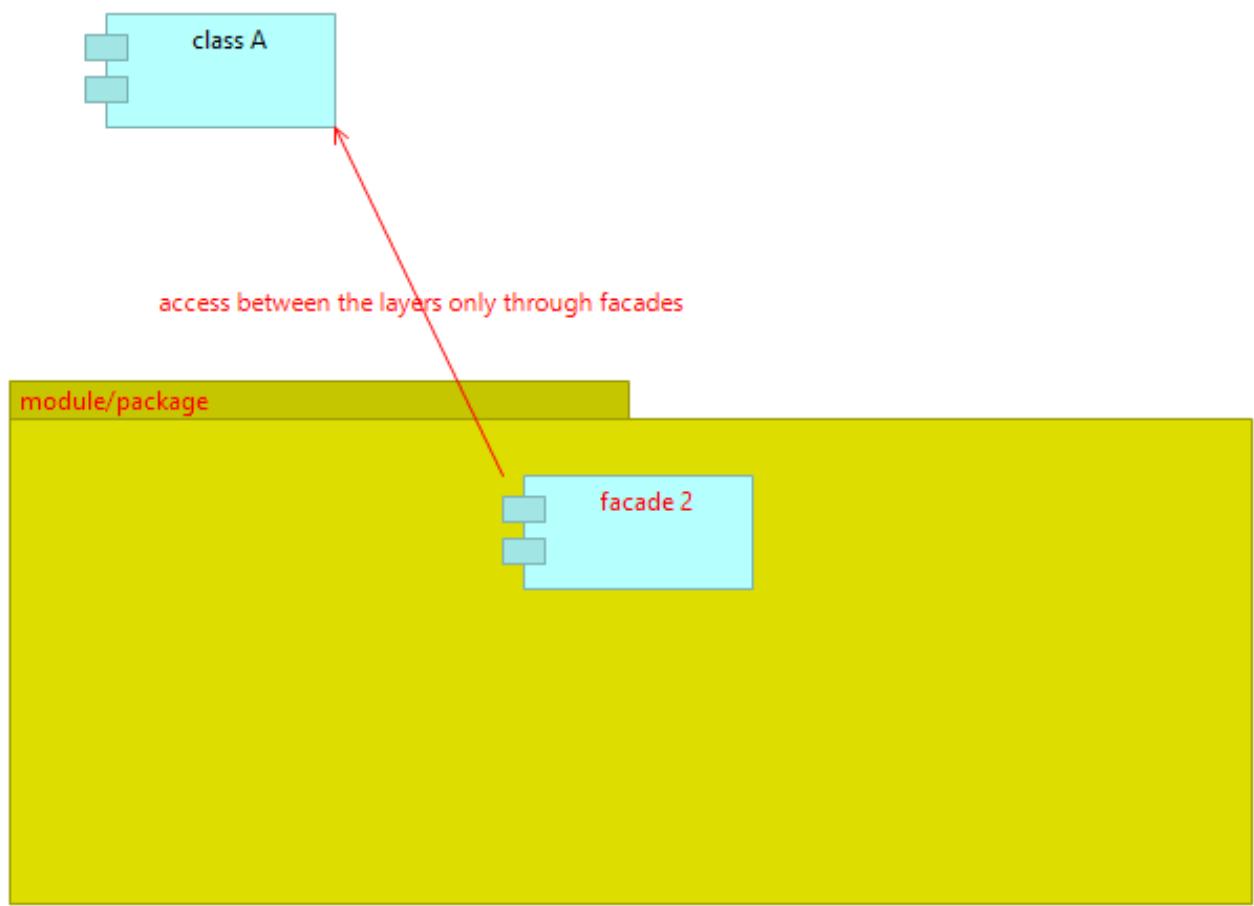
DECORATOR

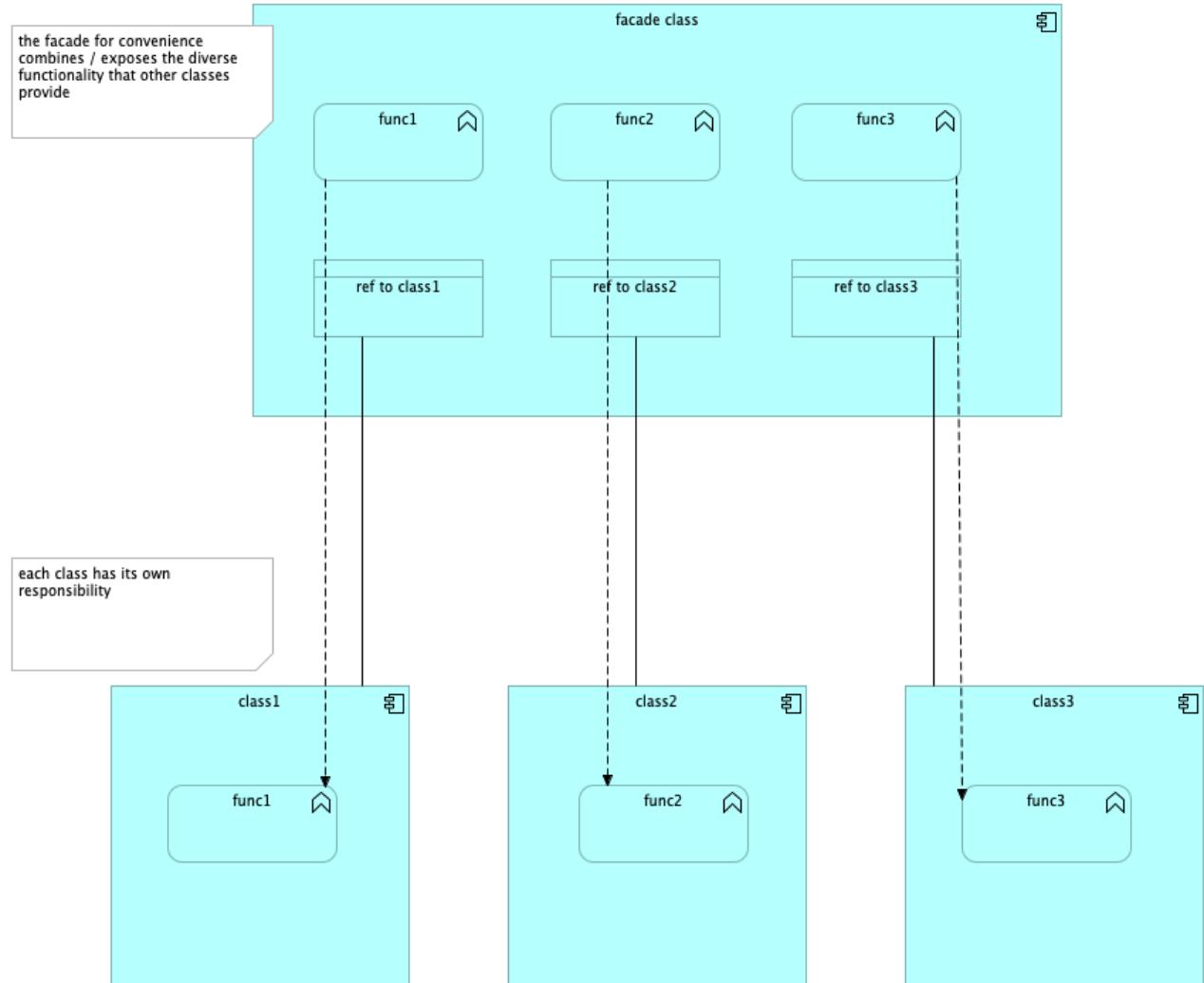




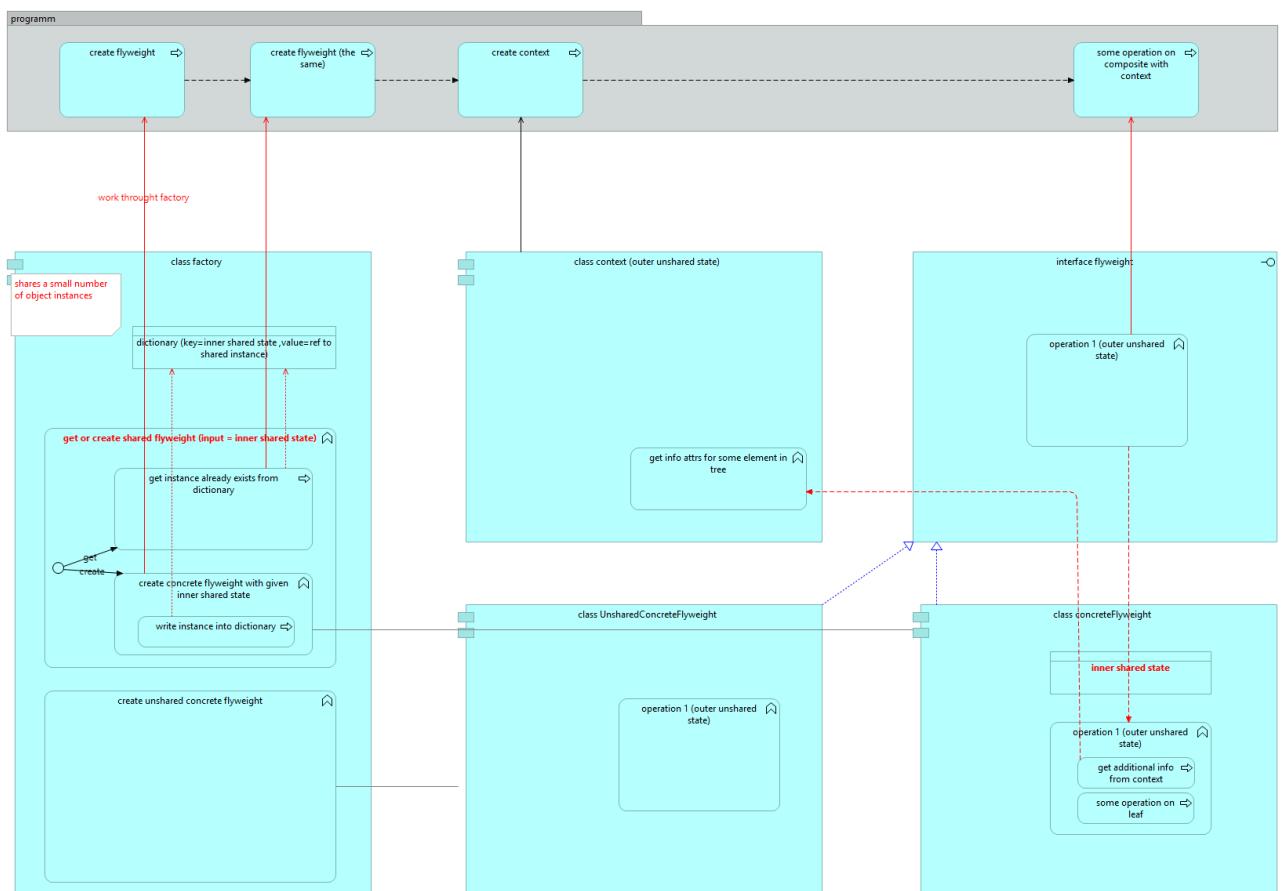
FACADE

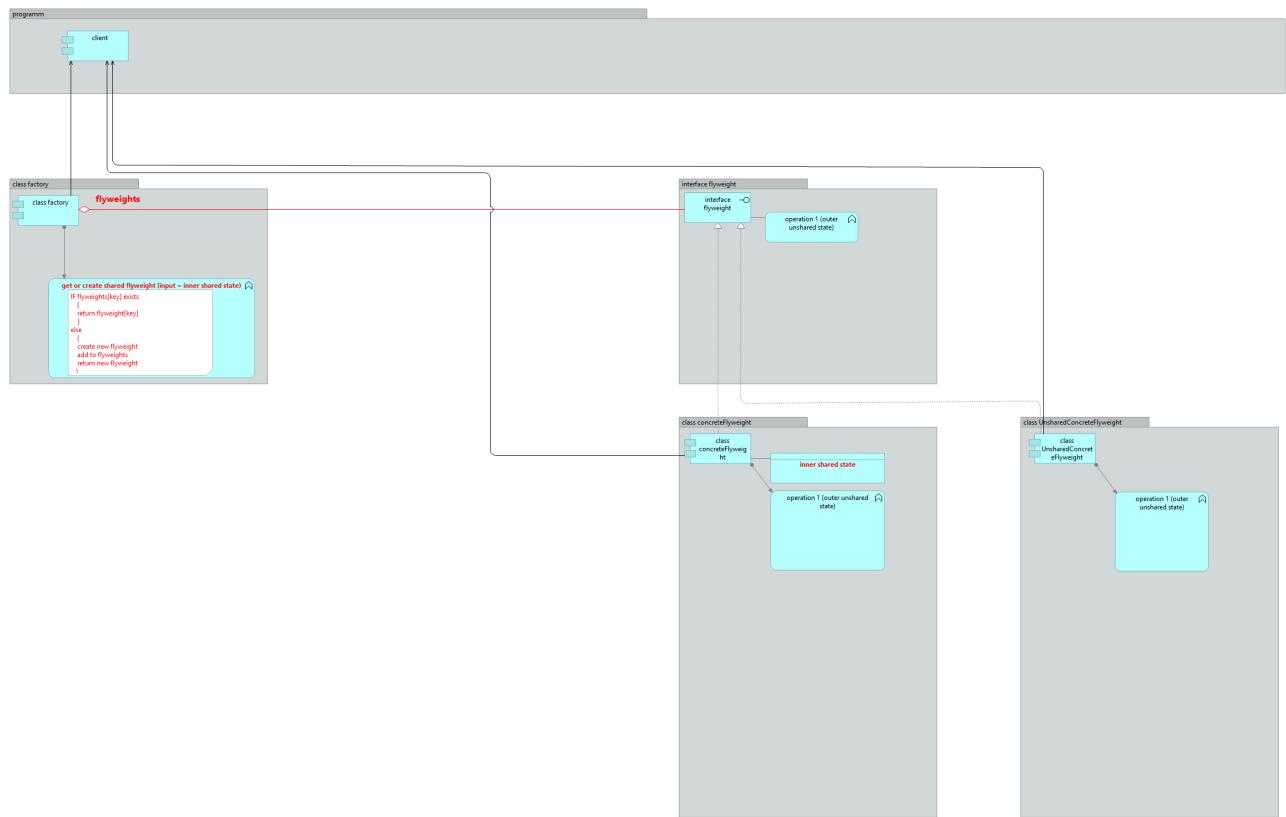




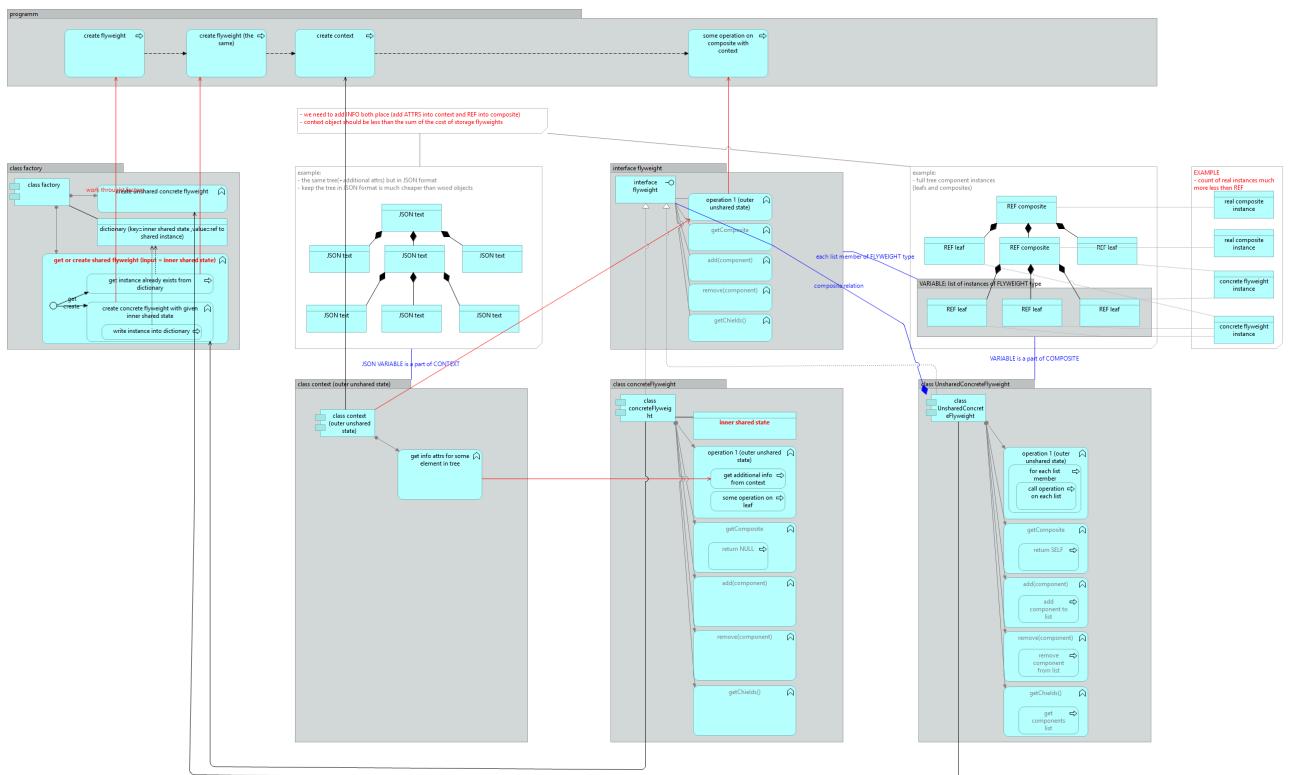


FLYWEIGHT

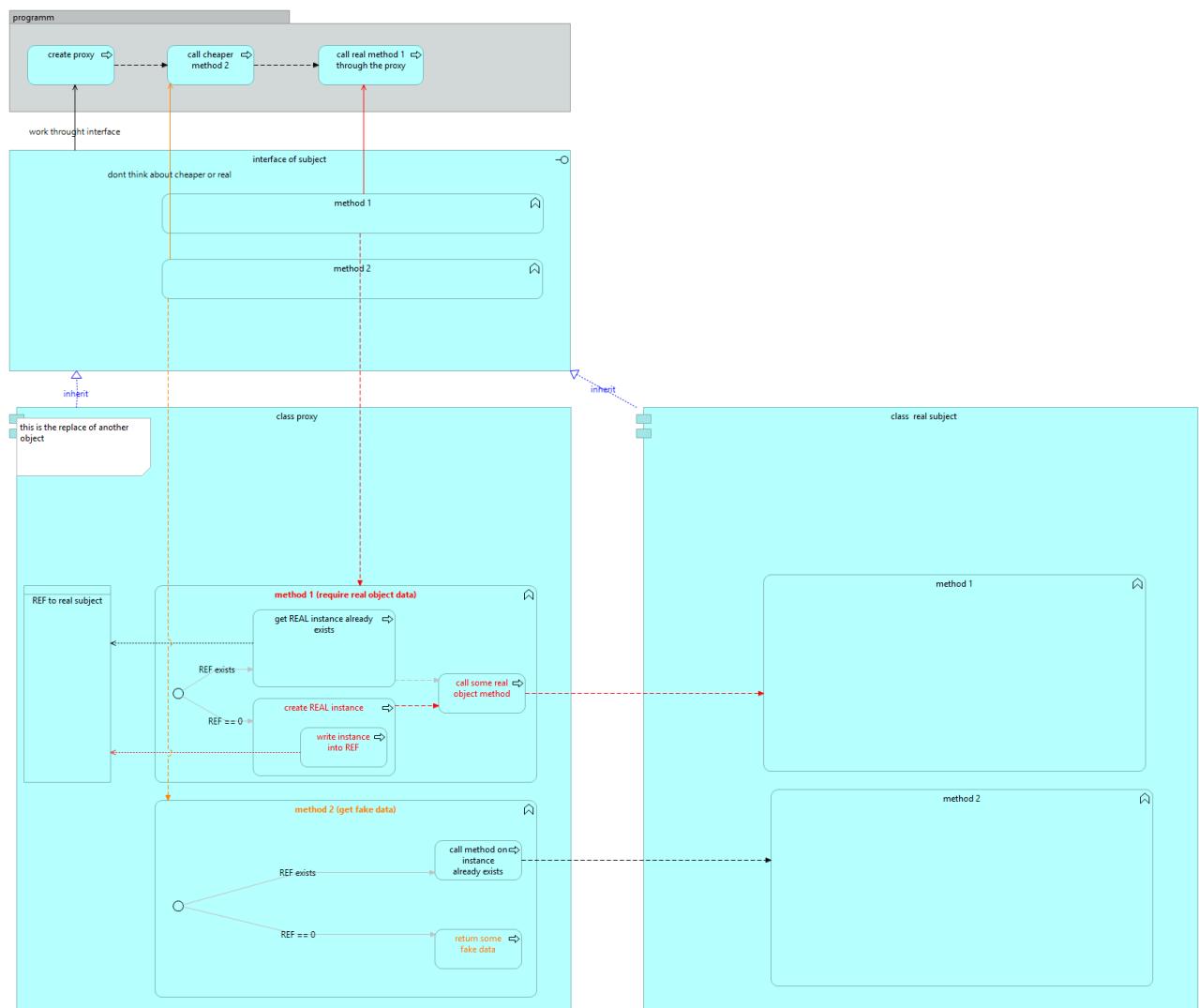


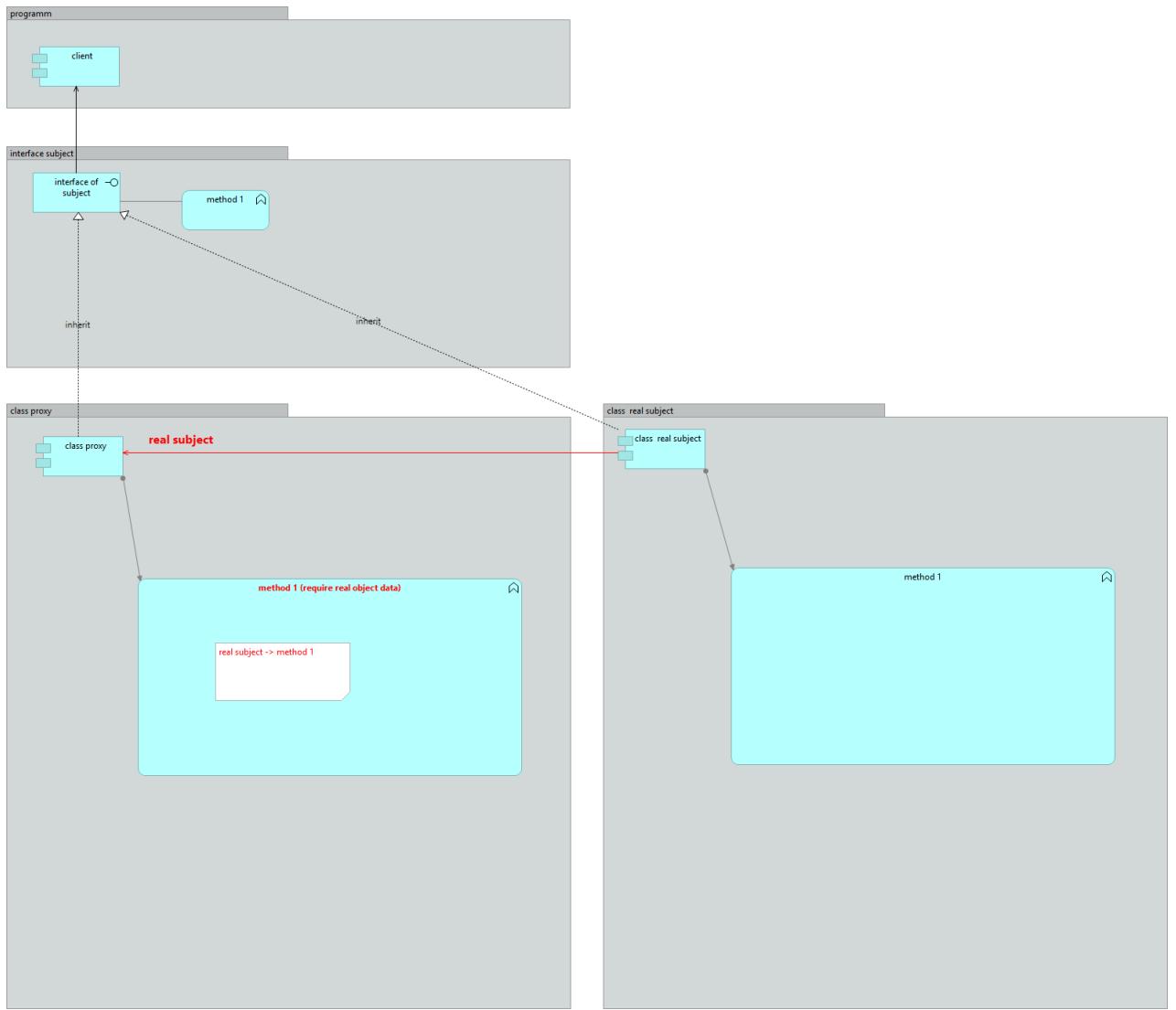


FLYWEIGHT + COMPOSITE



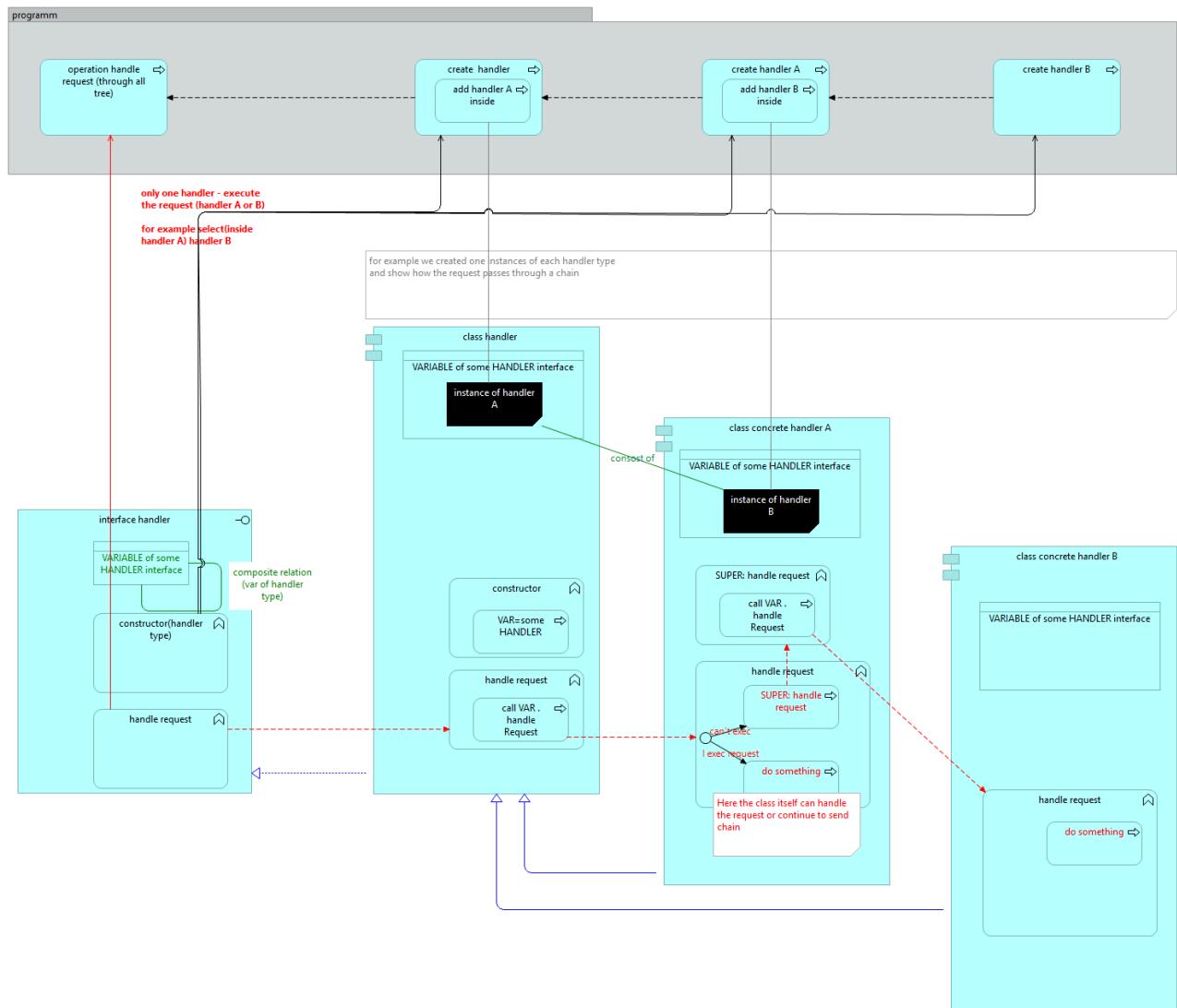
PROXY

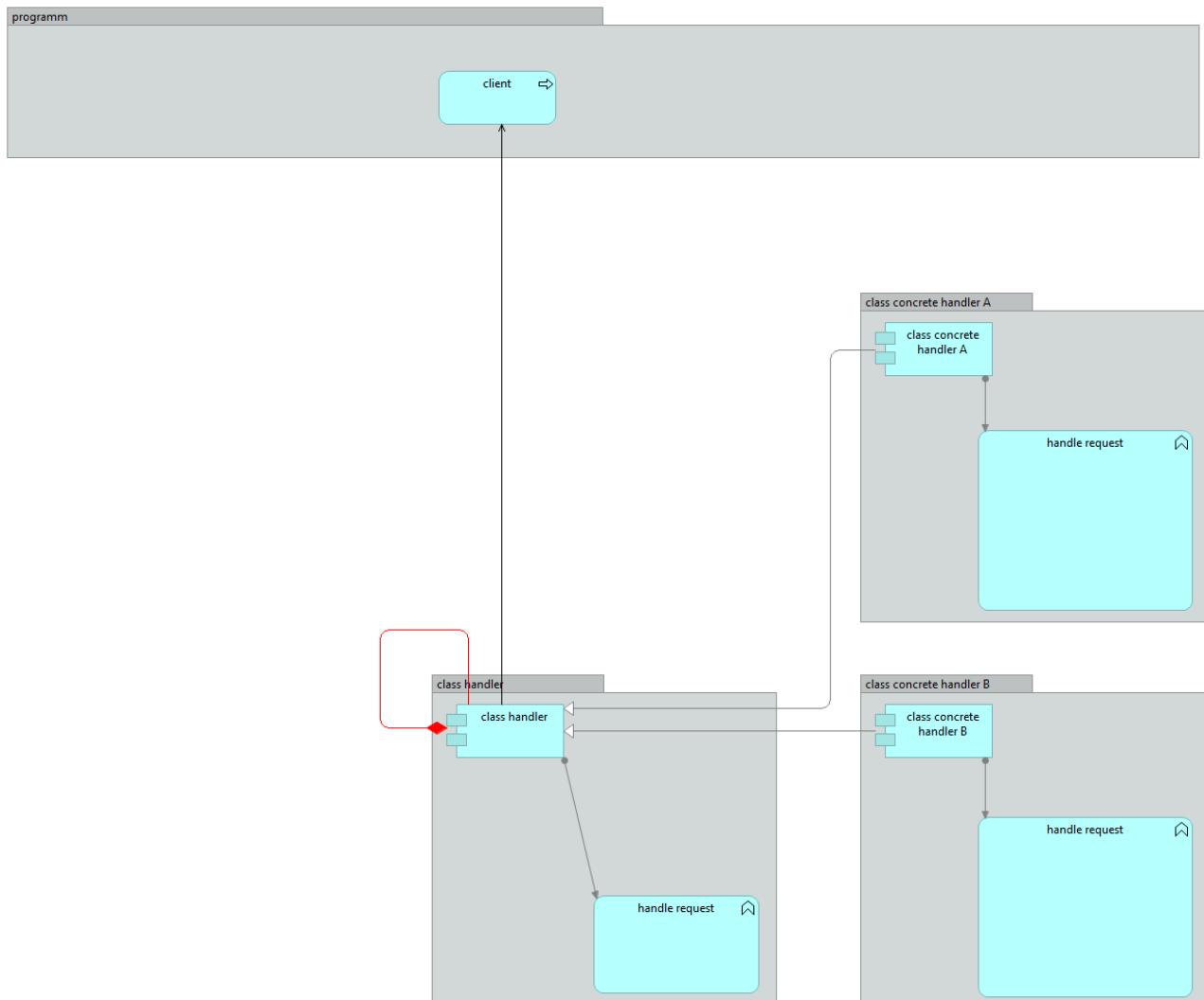




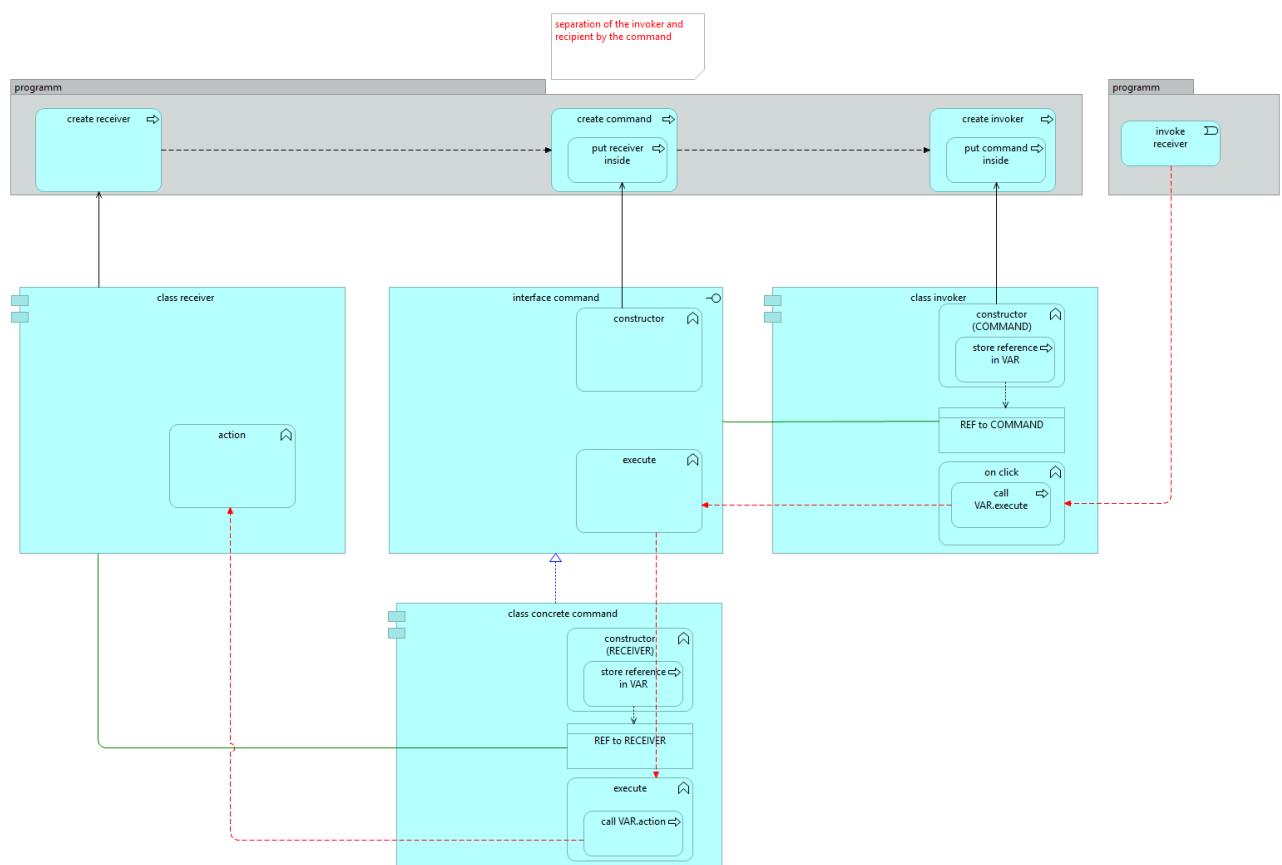
BEHAVIORAL PATTERNS

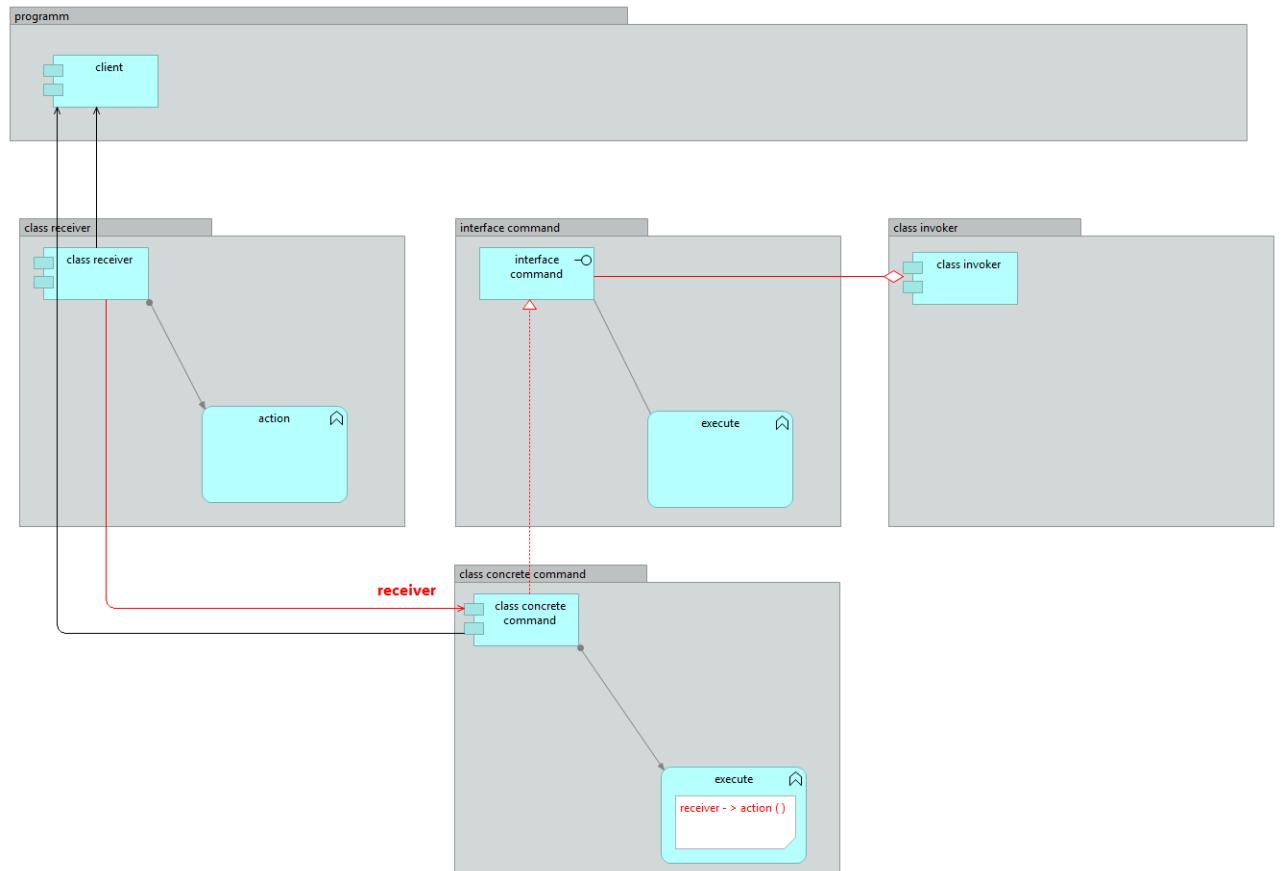
CHAIN OF RESPONSIBILITY



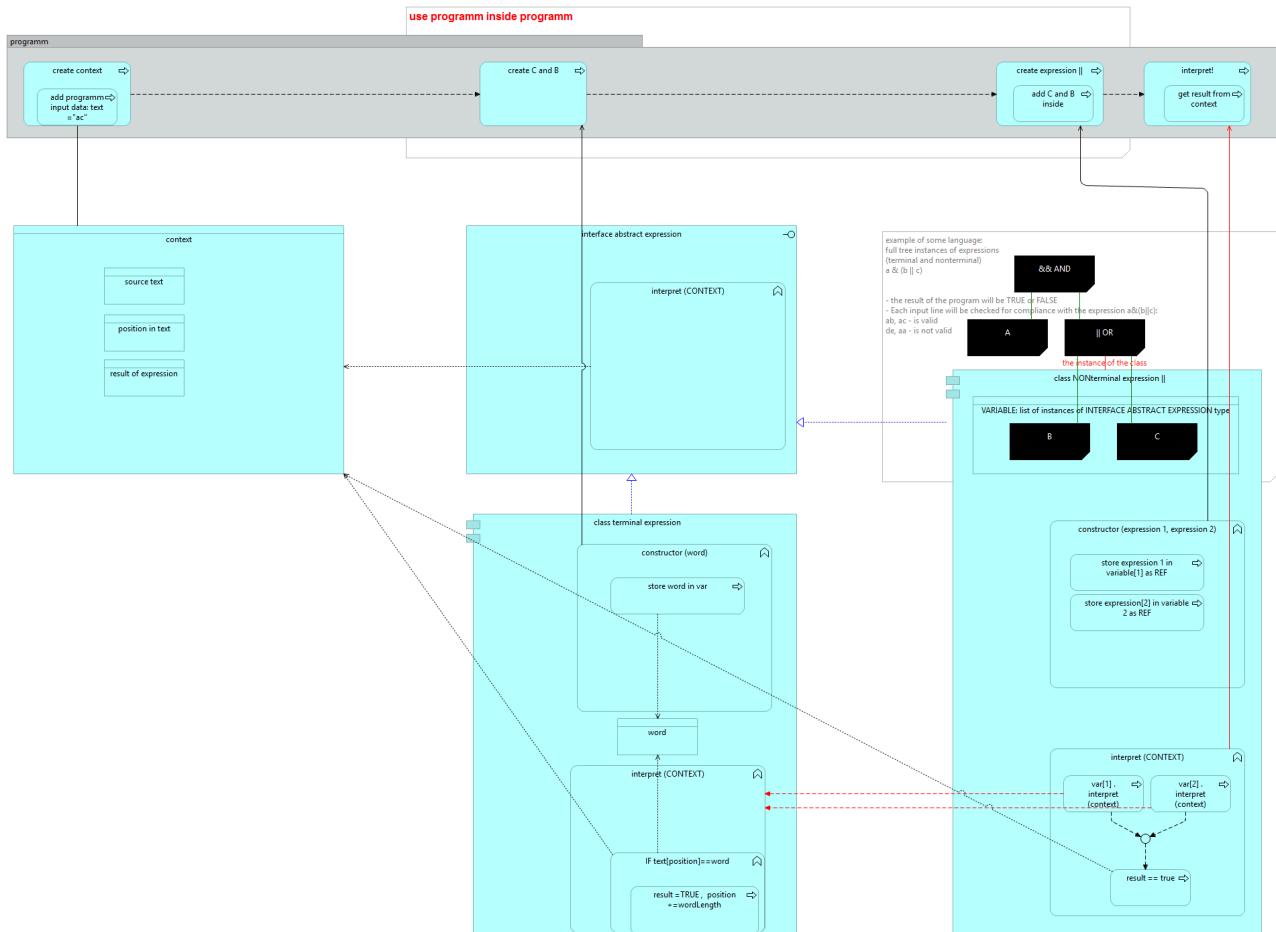


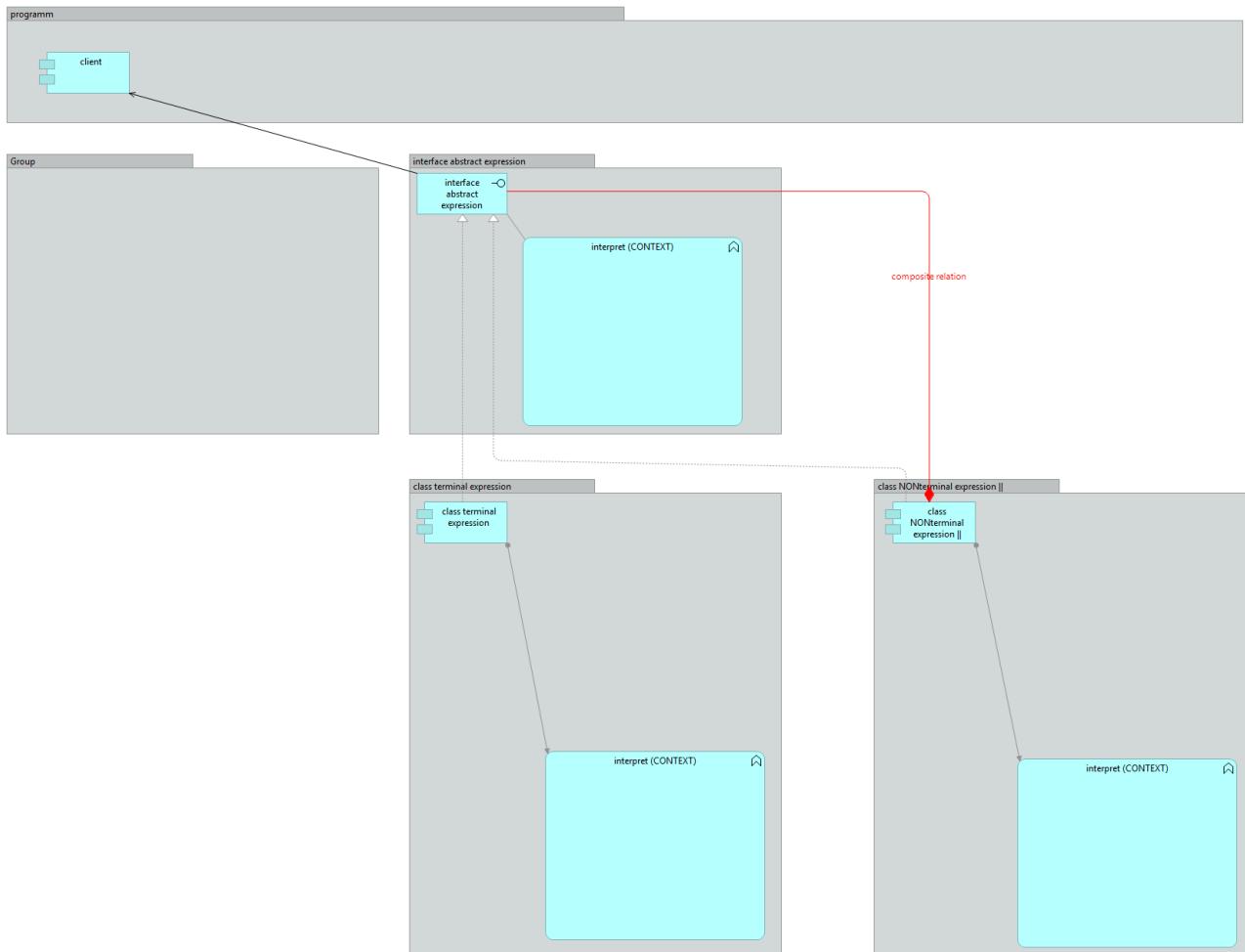
COMMAND



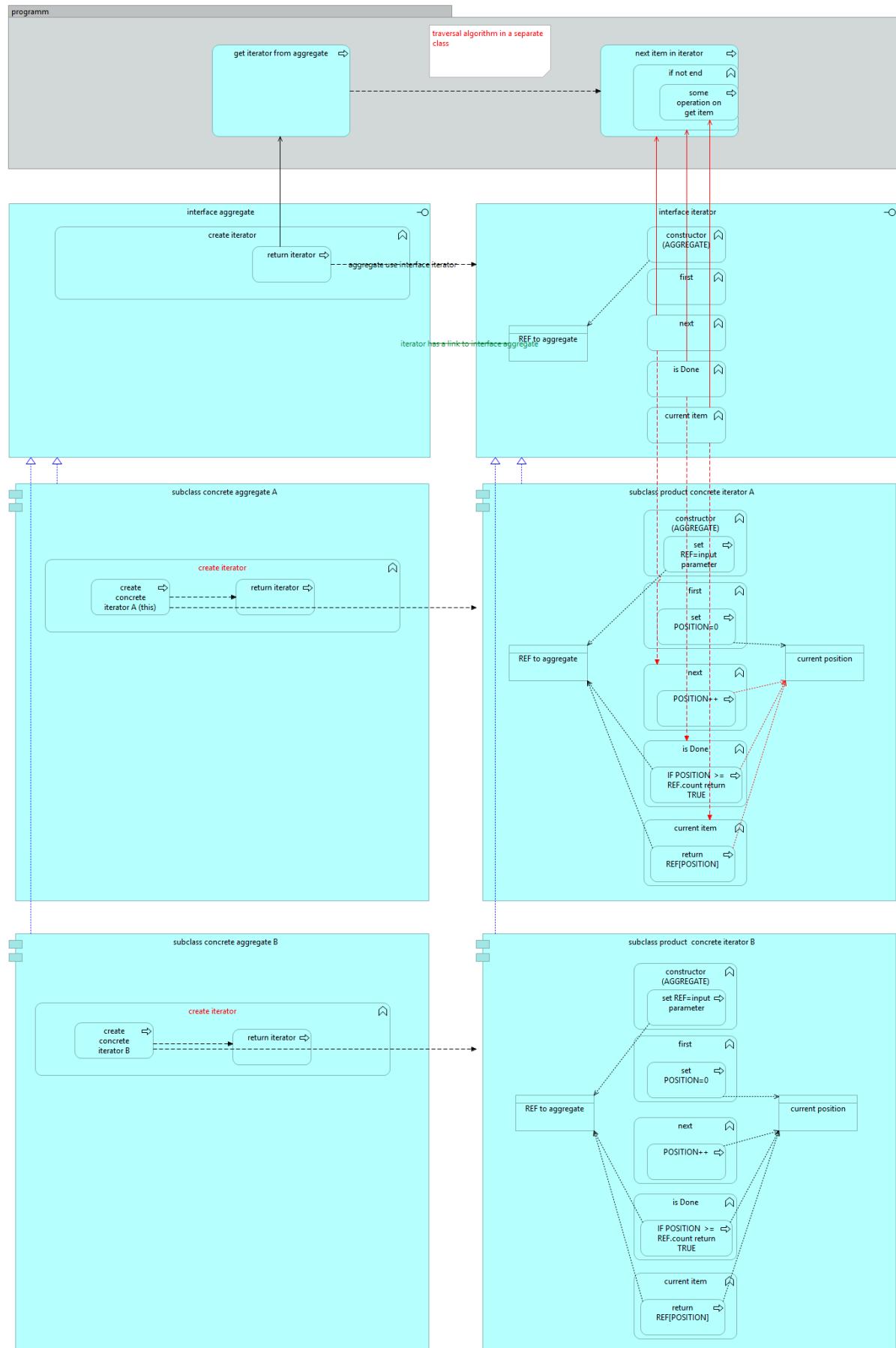


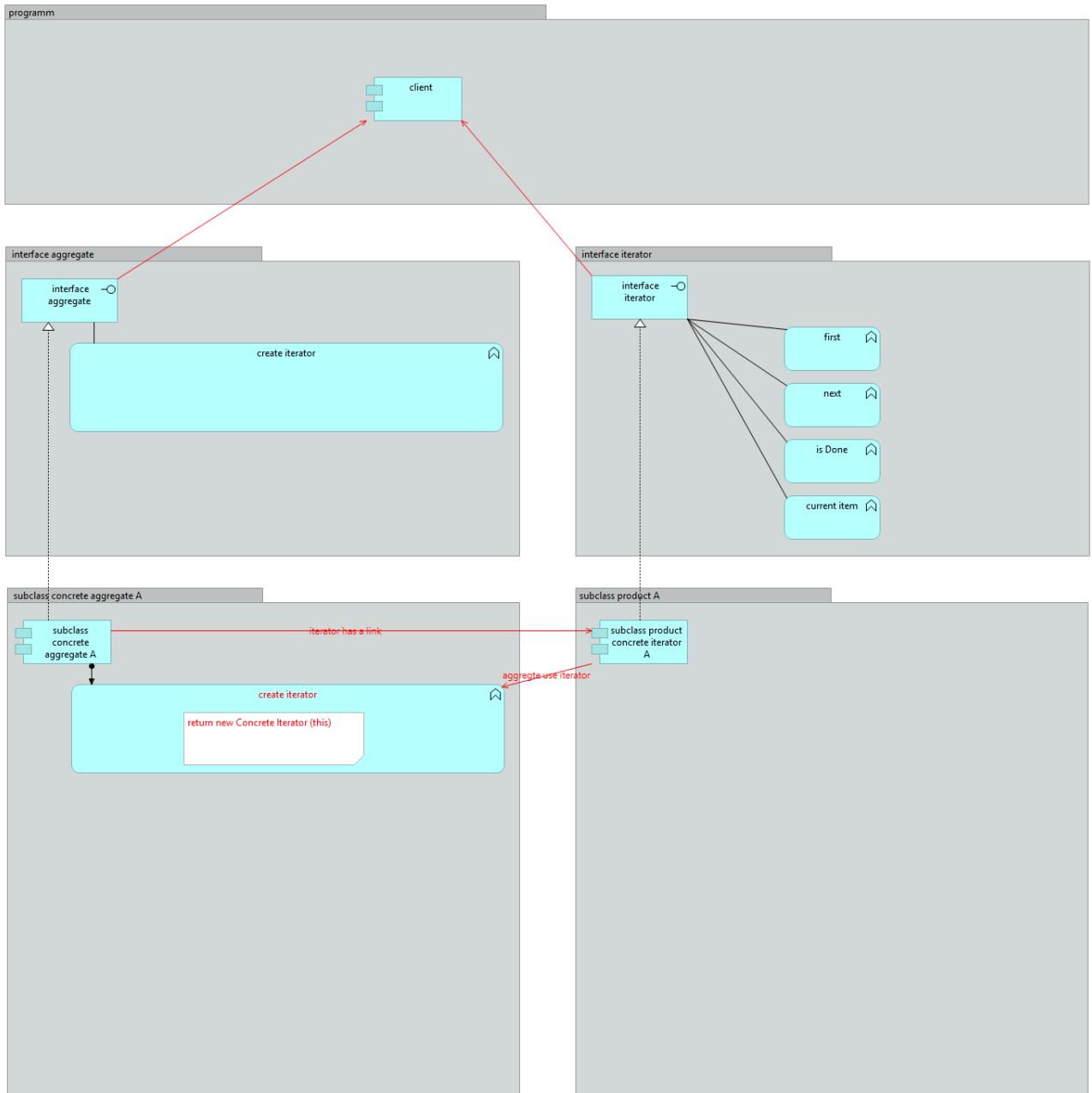
INTERPRETER



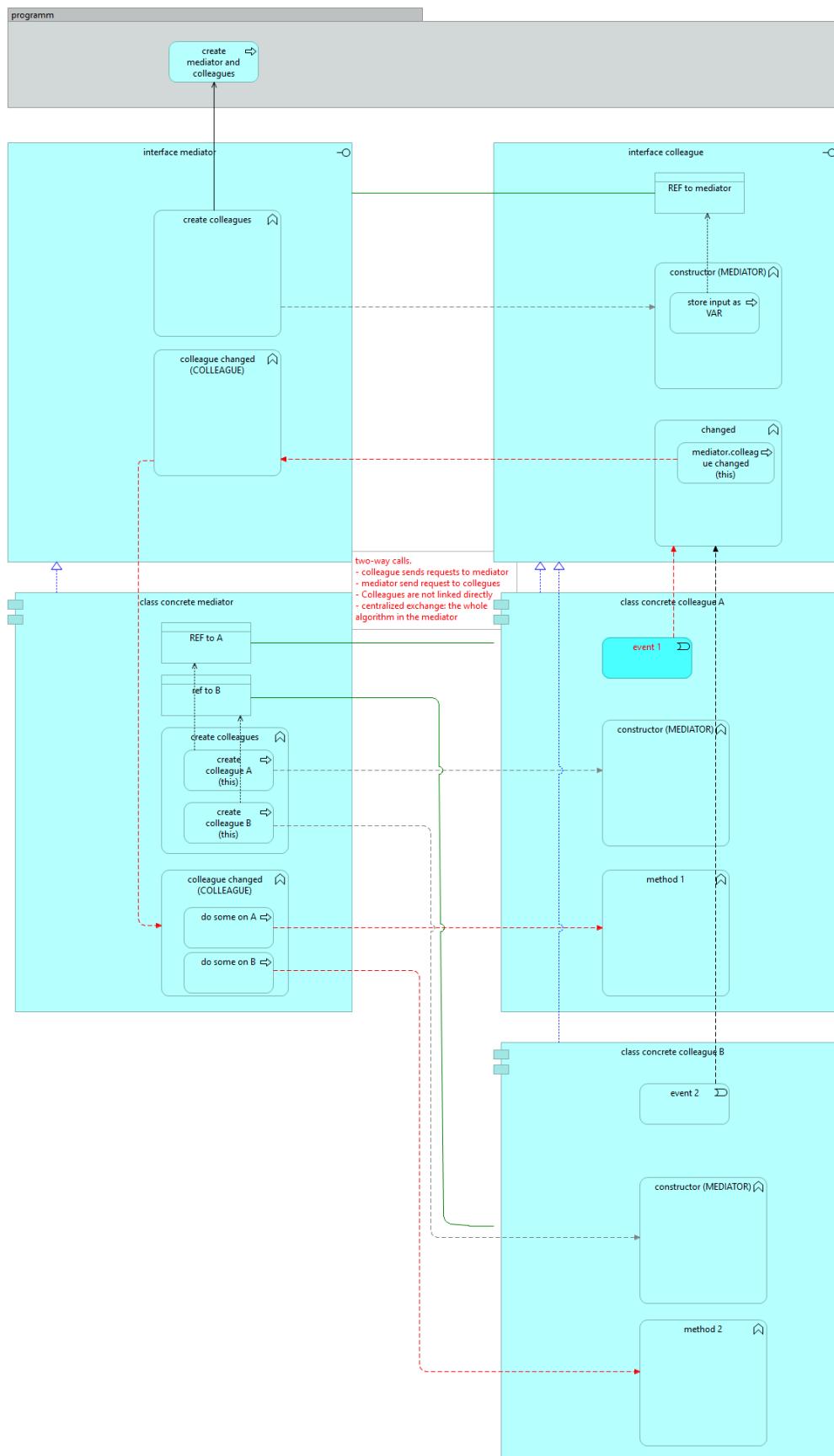


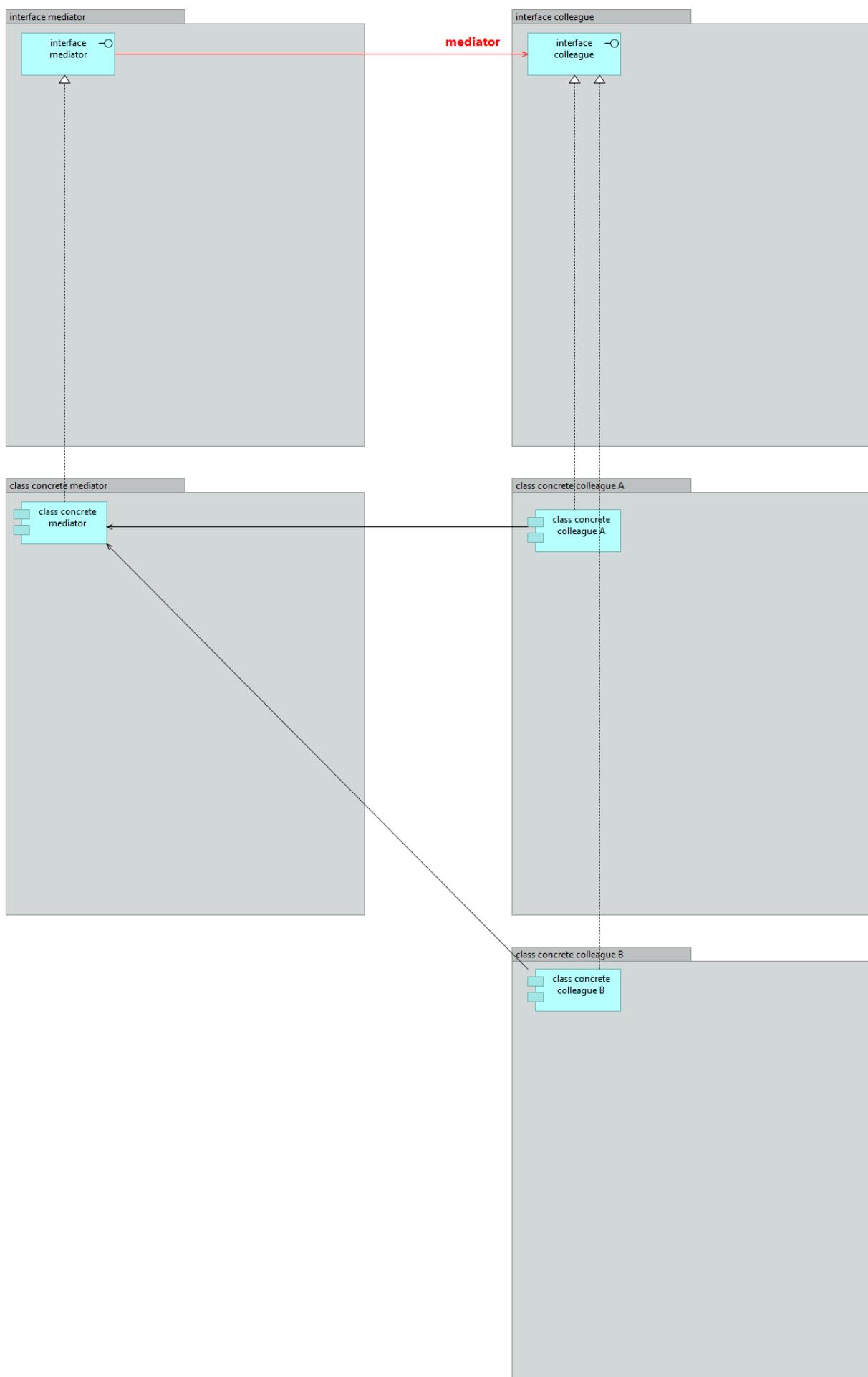
ITERATOR



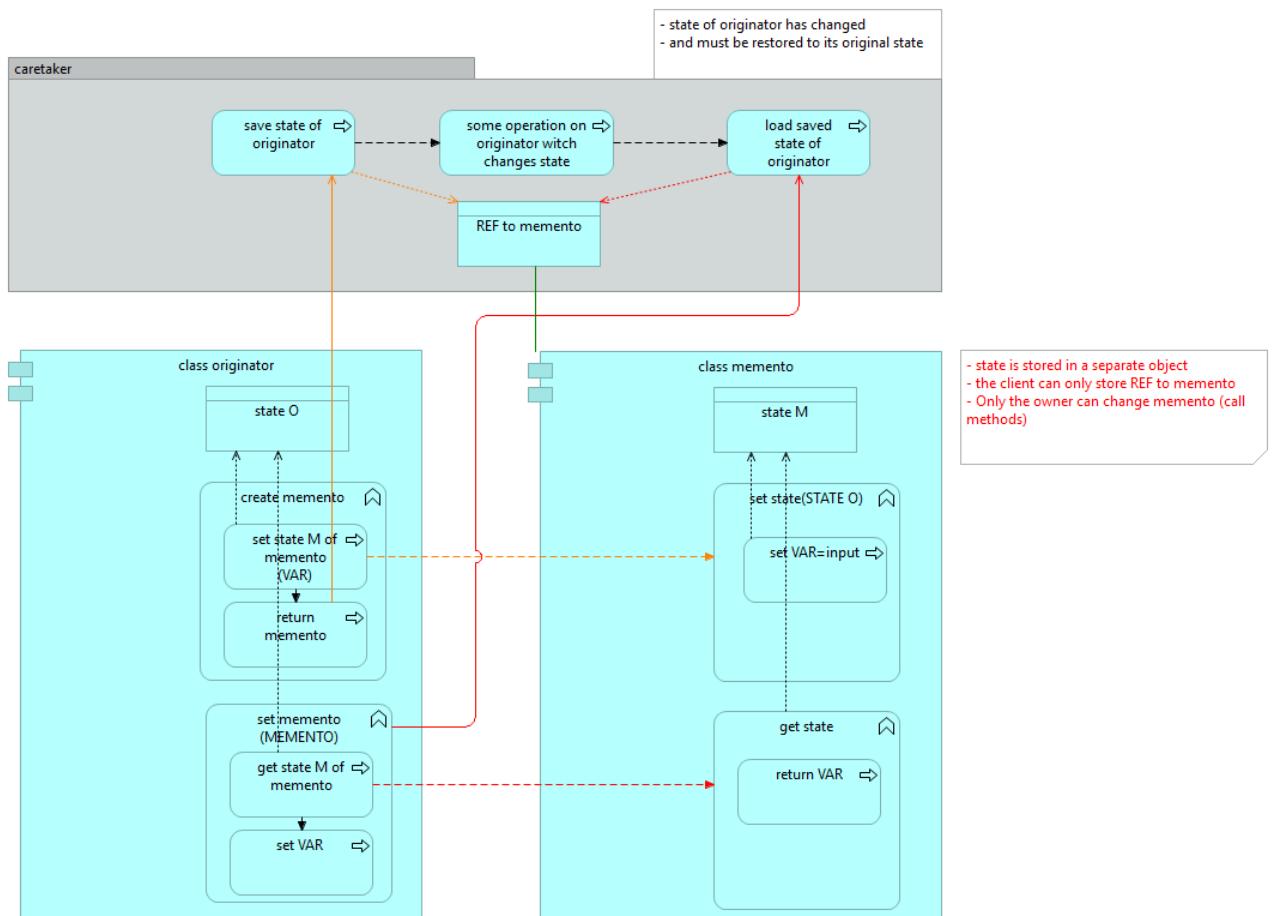


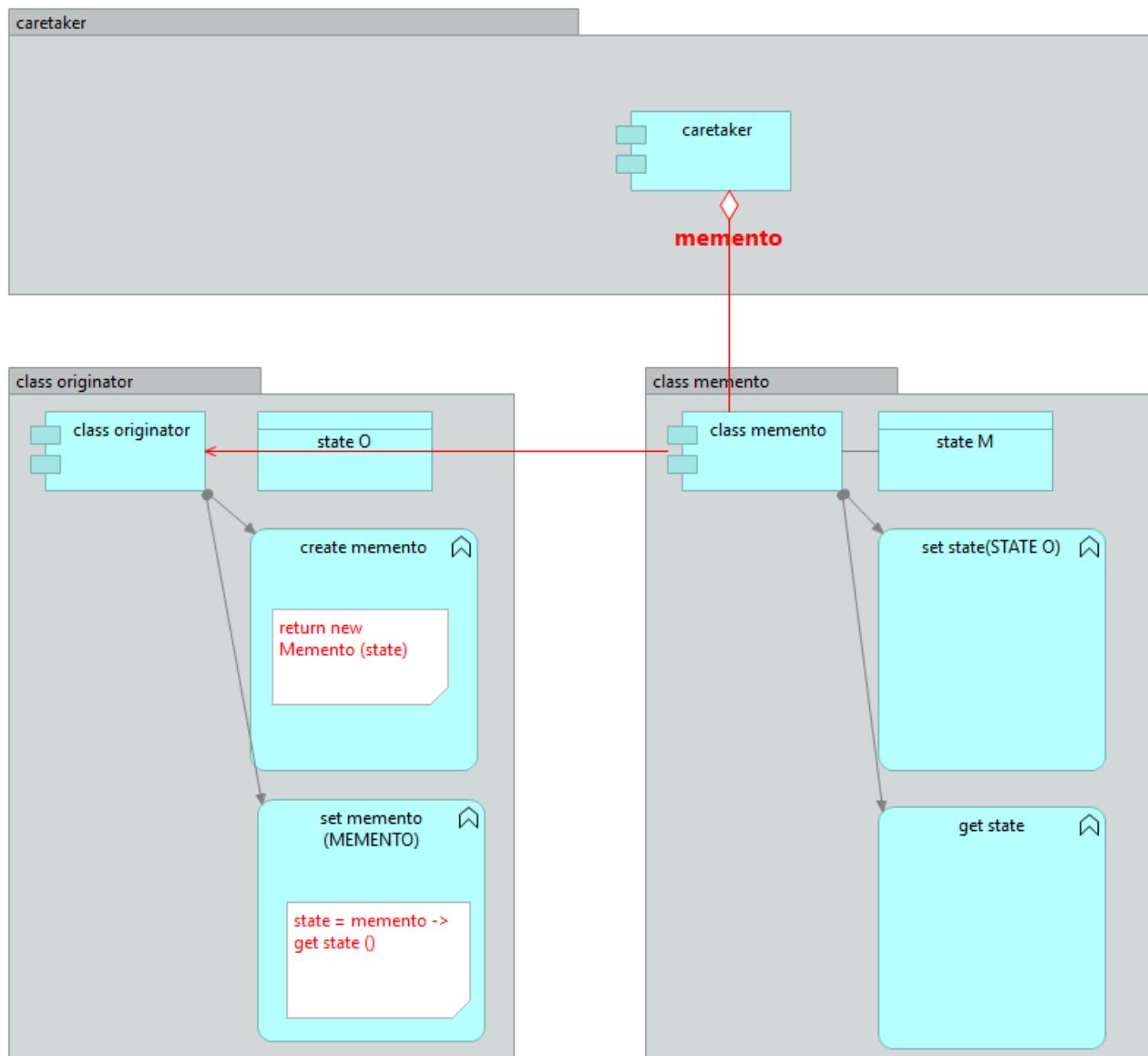
MEDIATOR



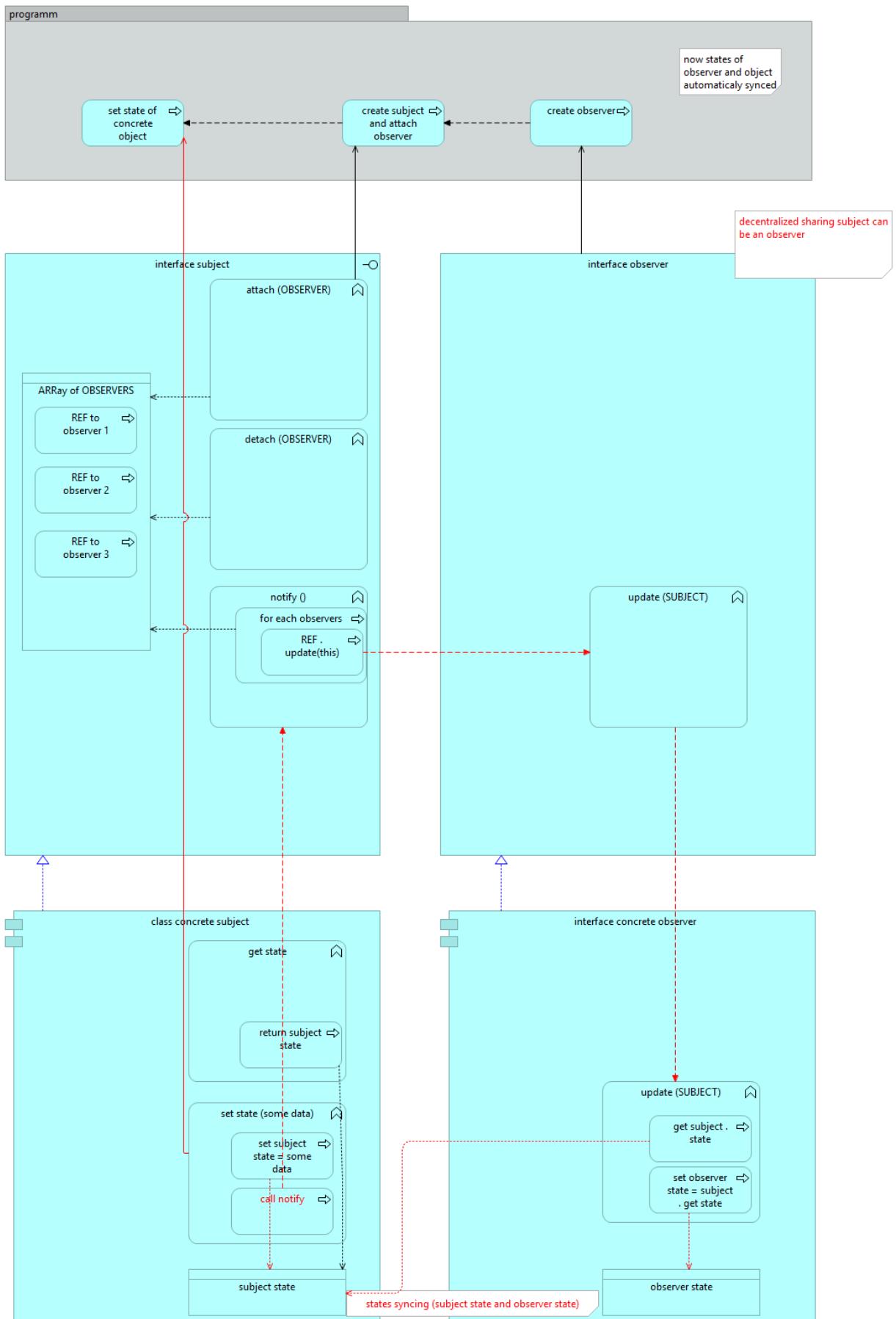


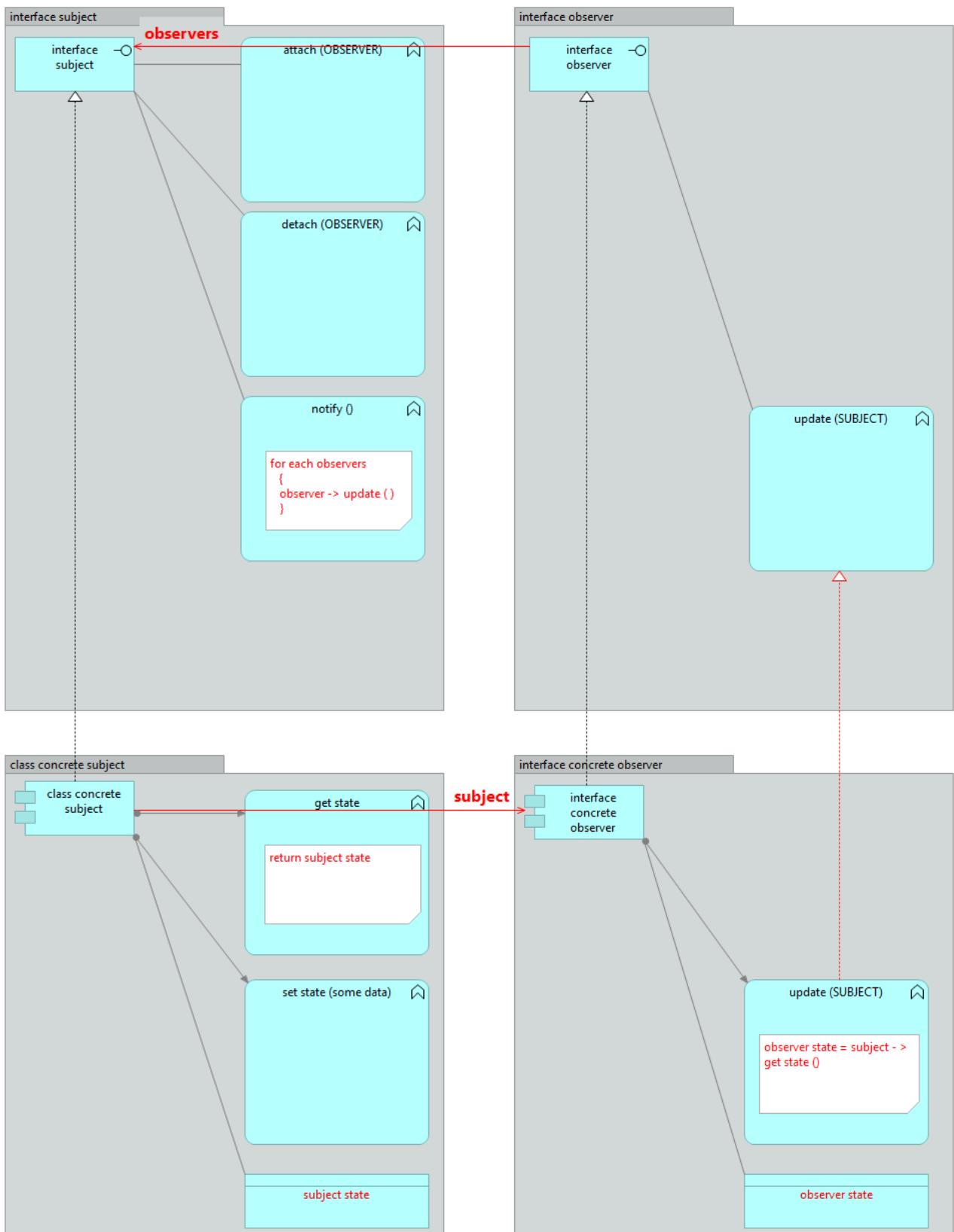
MEMENTO



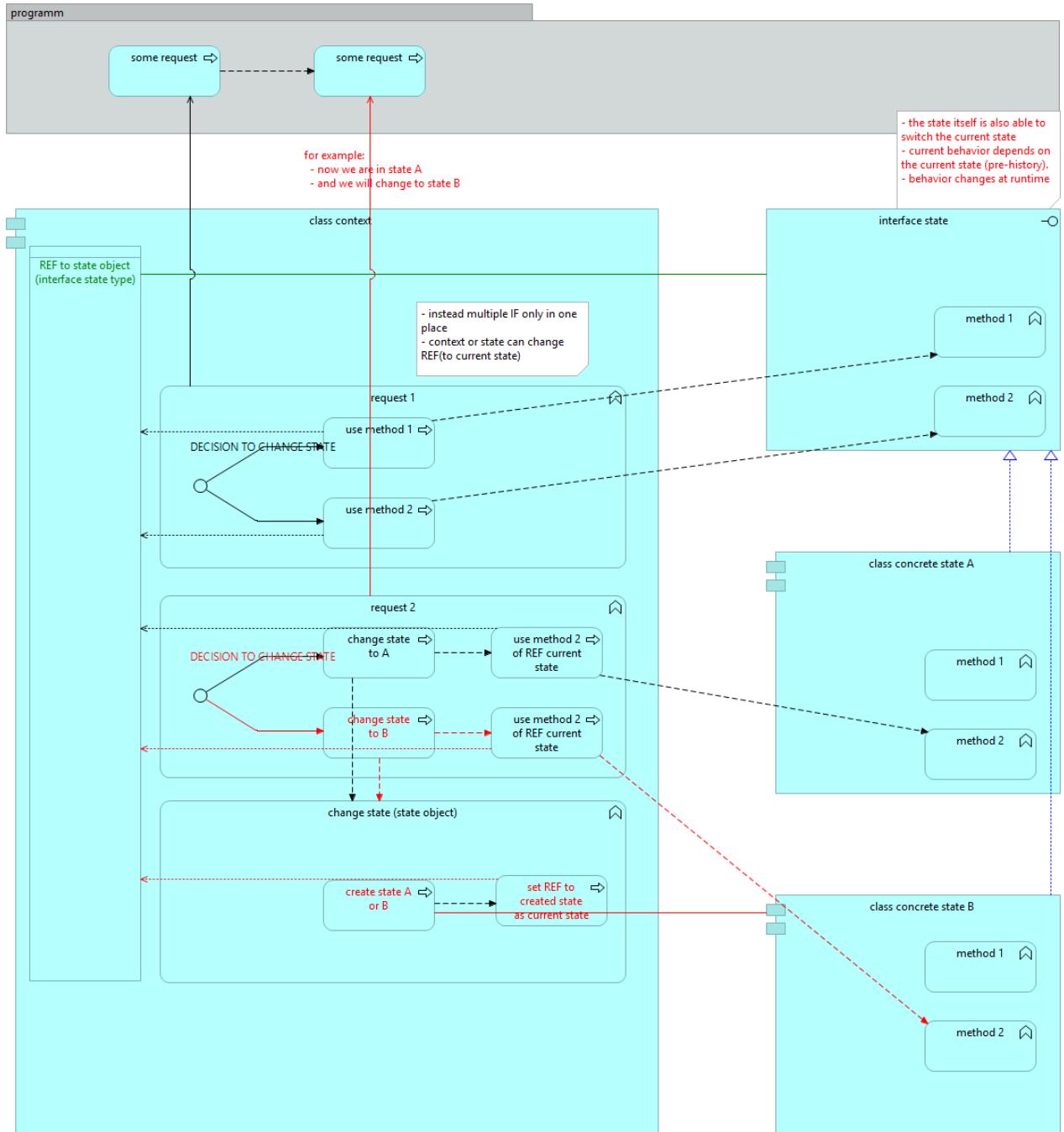


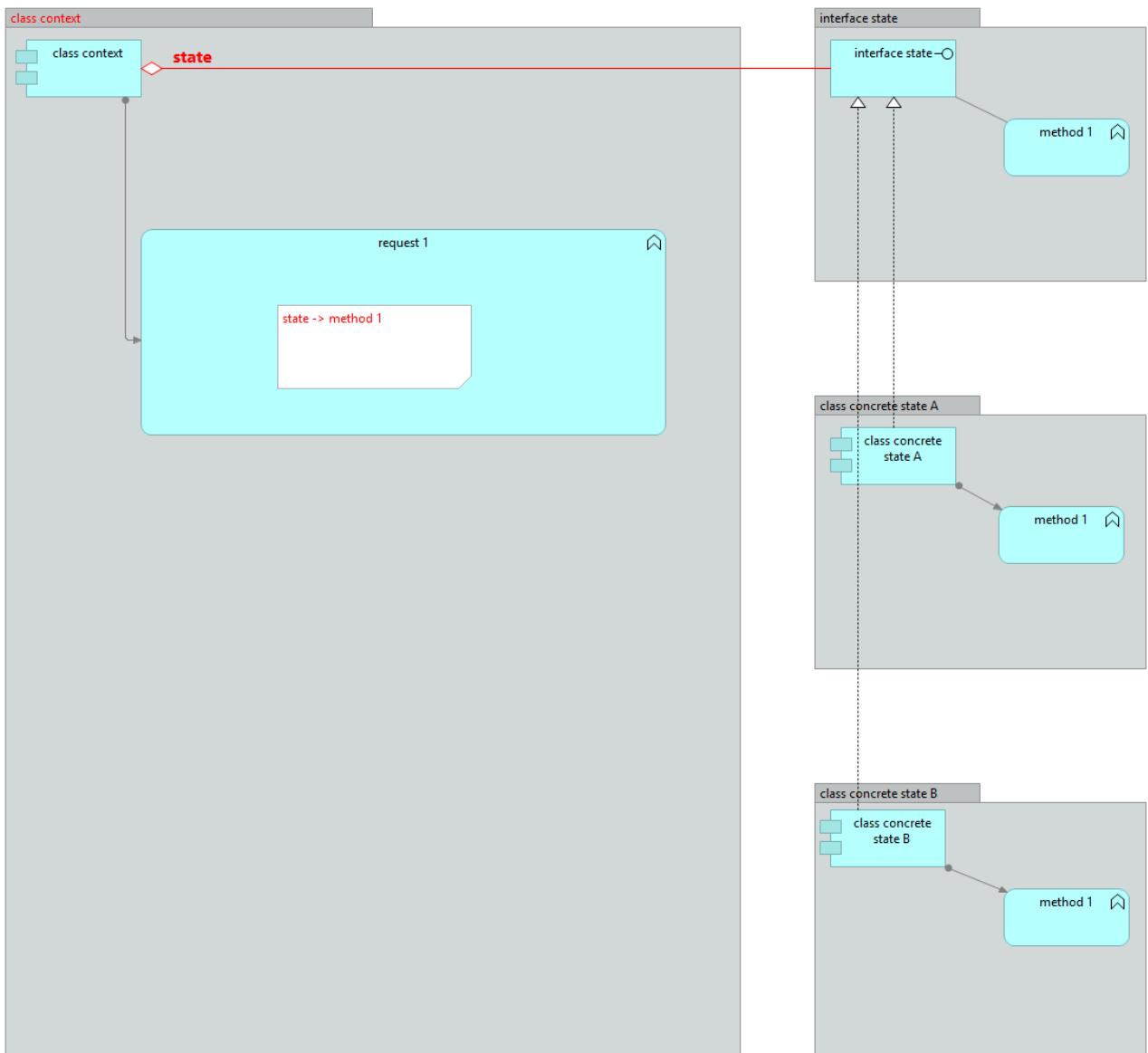
OBSERVER



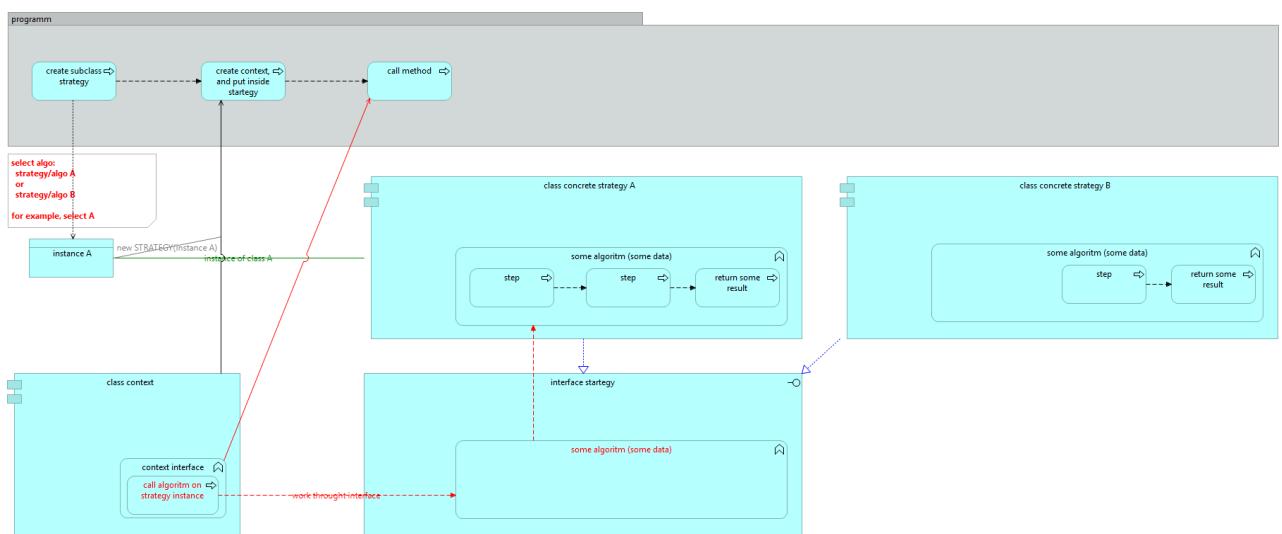


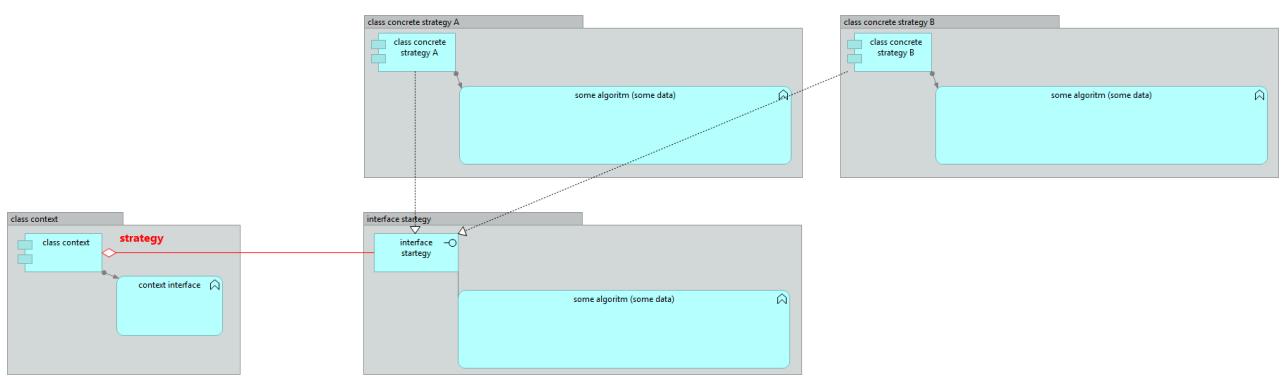
STATE



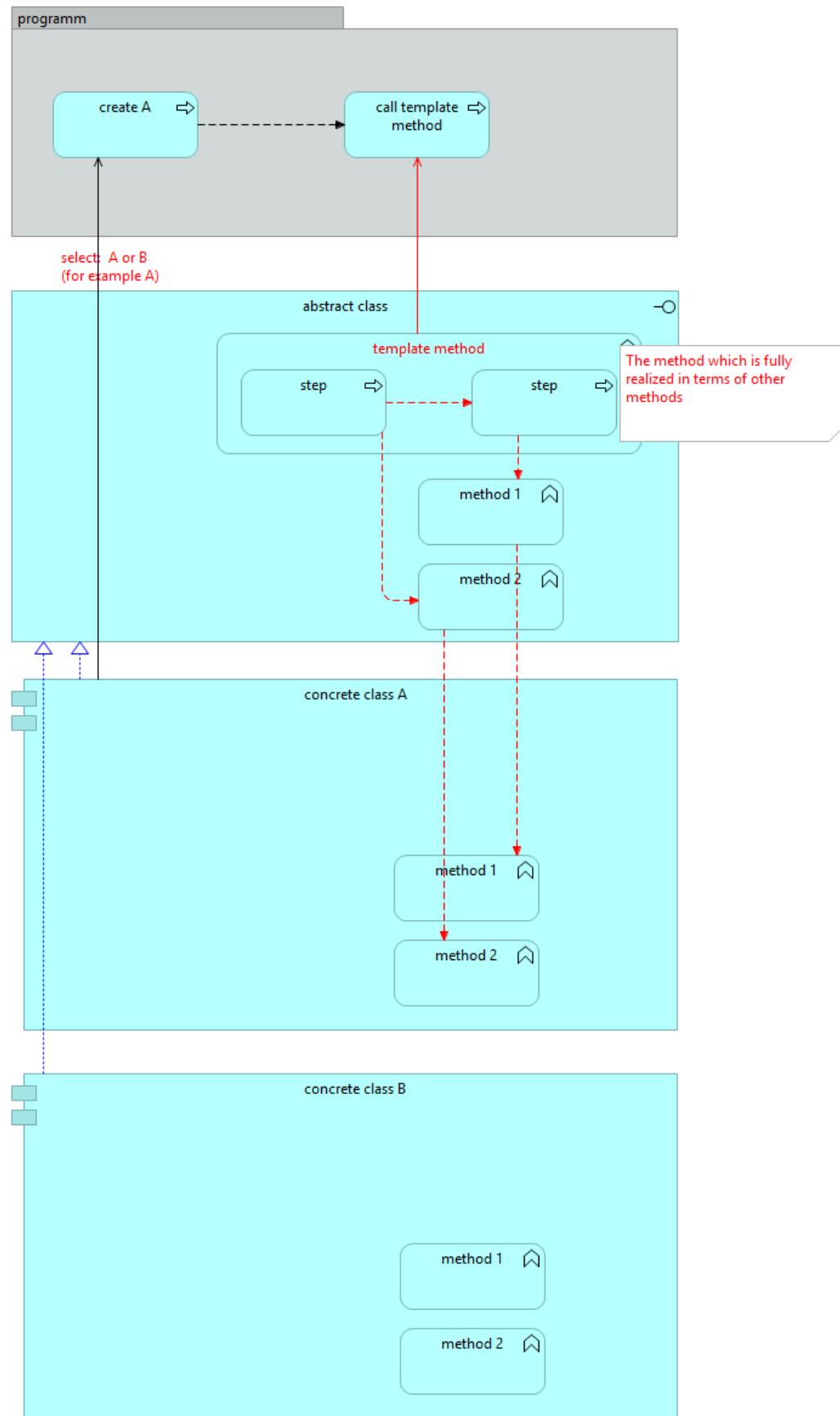


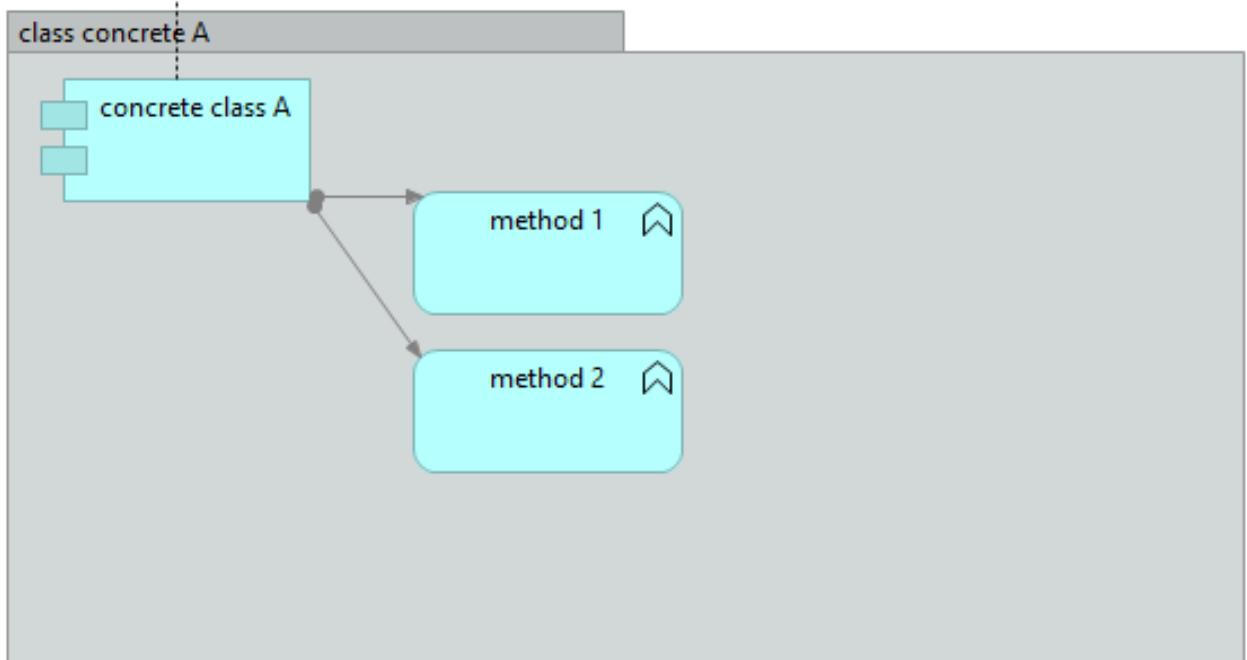
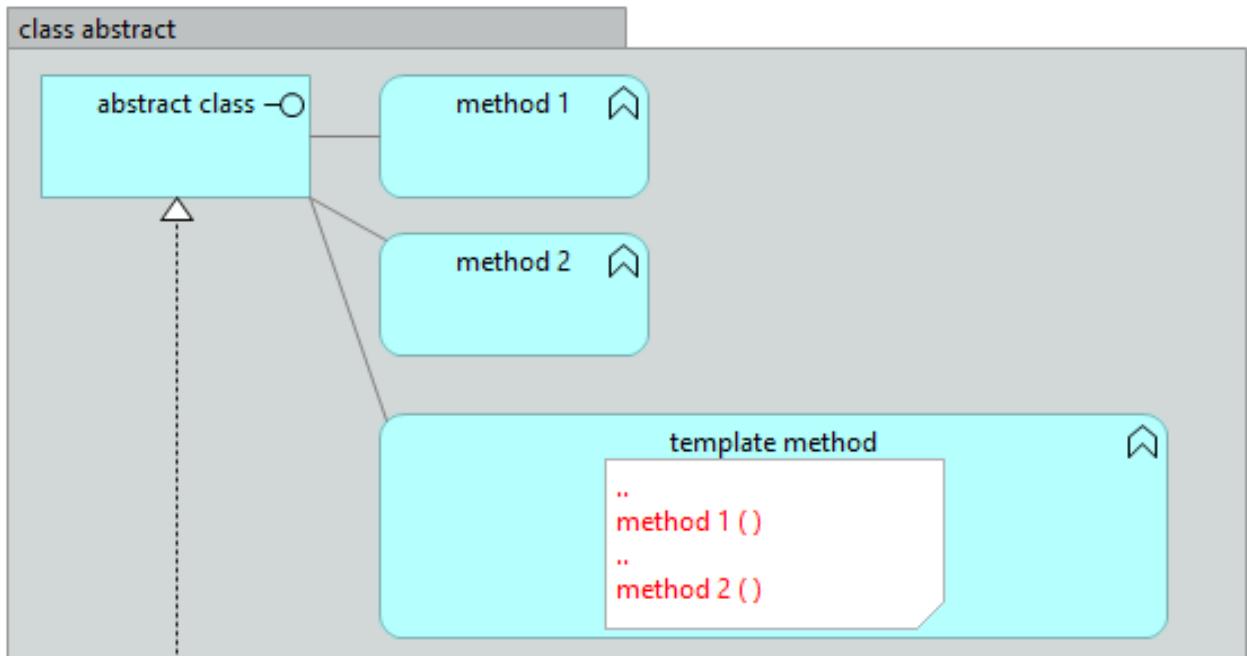
STRATEGY



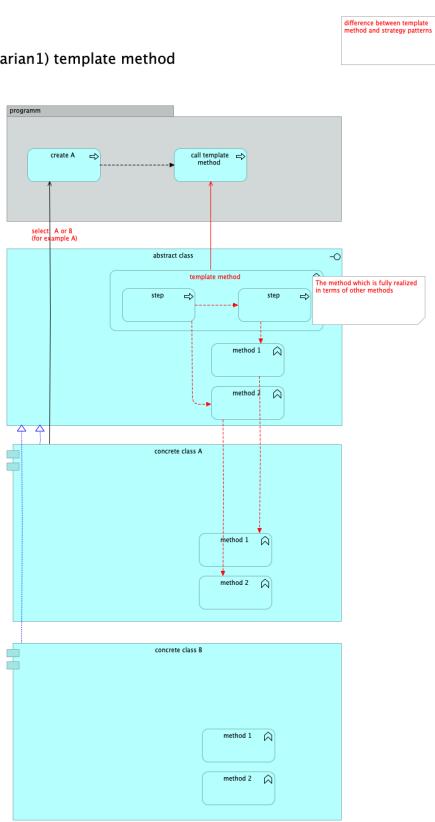


TEMPLATE METHOD

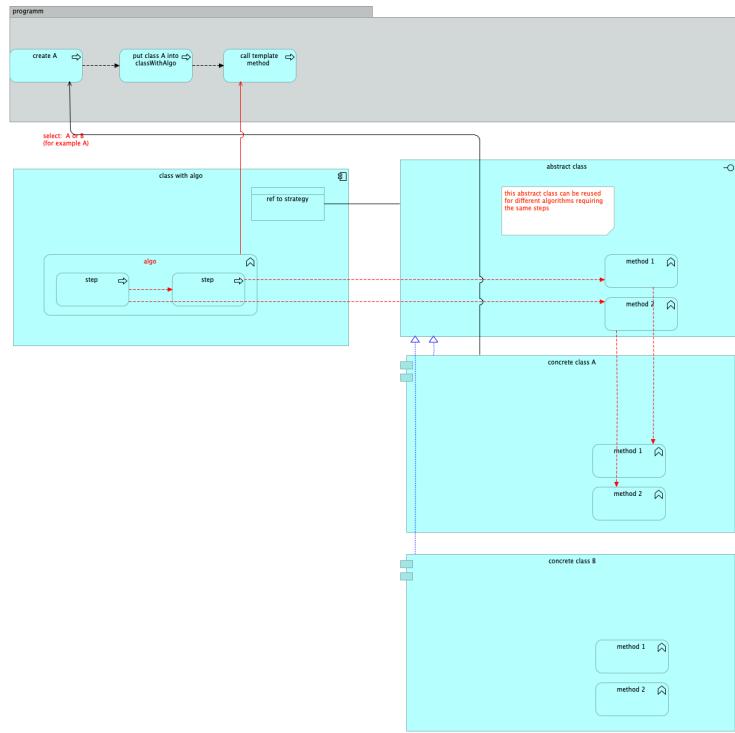




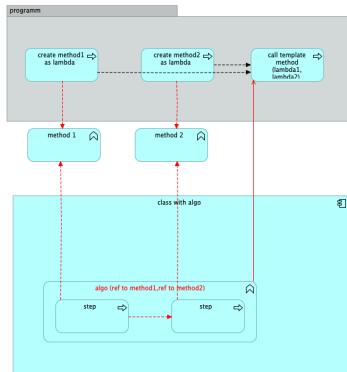
Varian1) template method



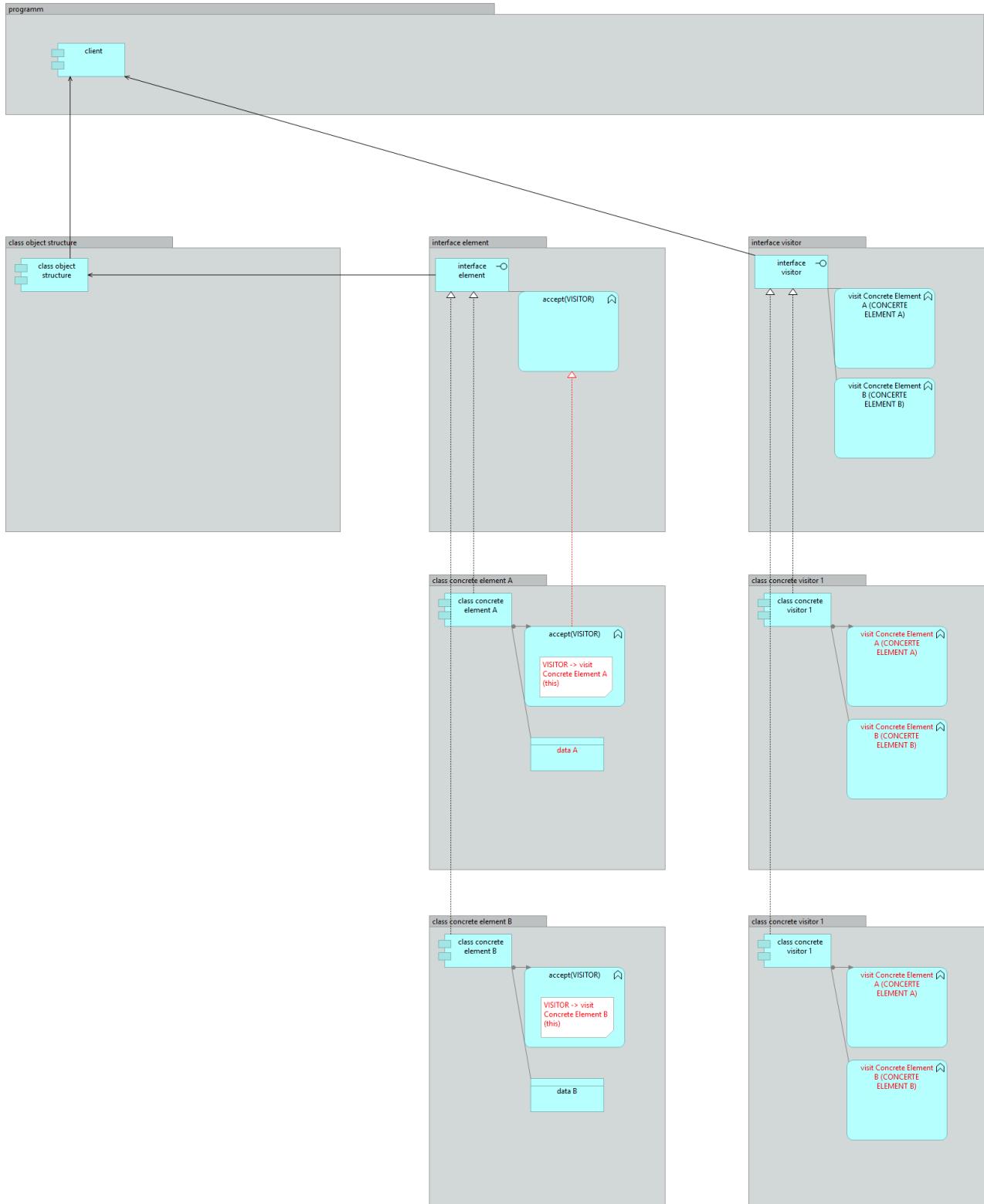
Varian2) strategy

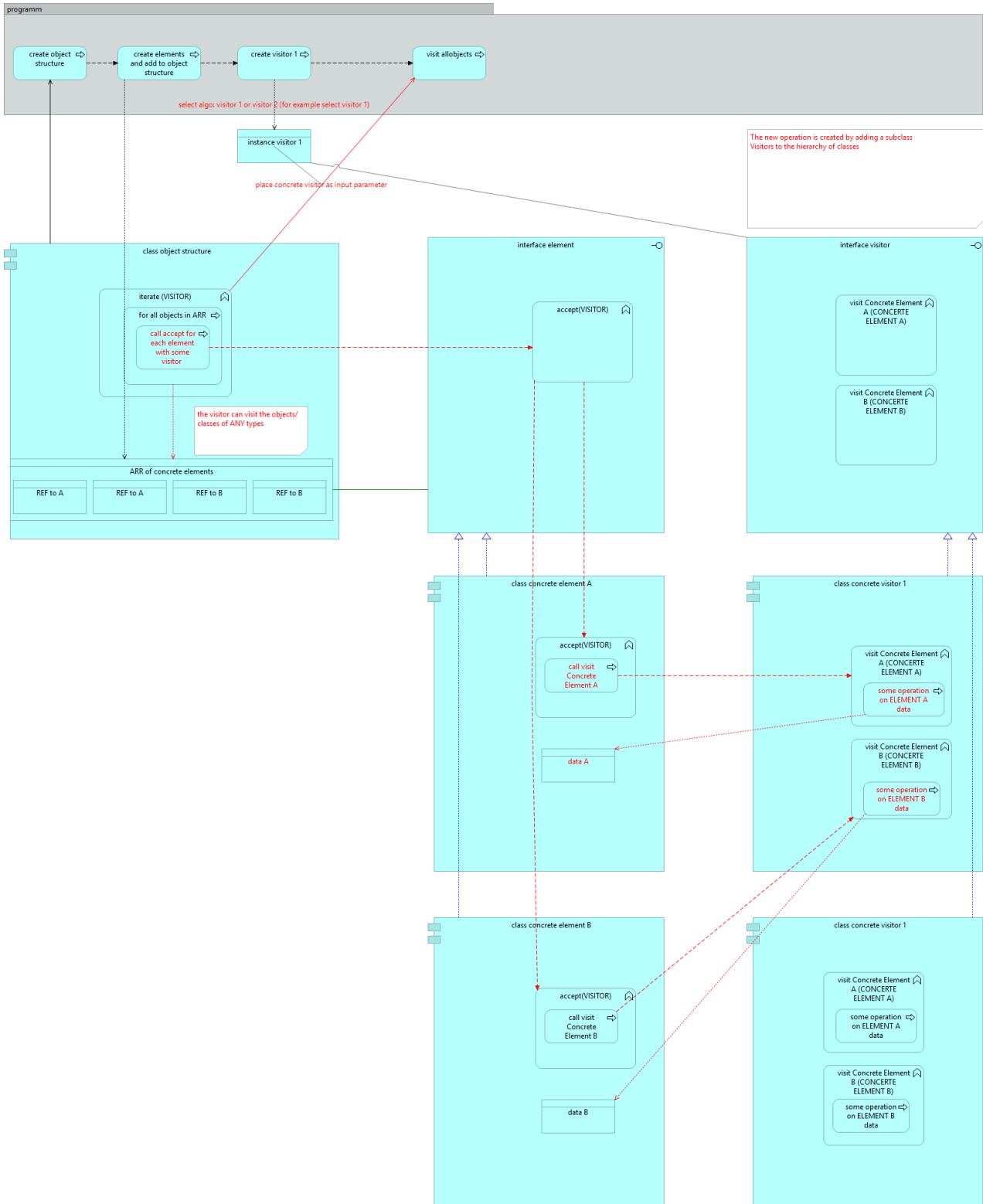


Varian3) lambdas as input parameters

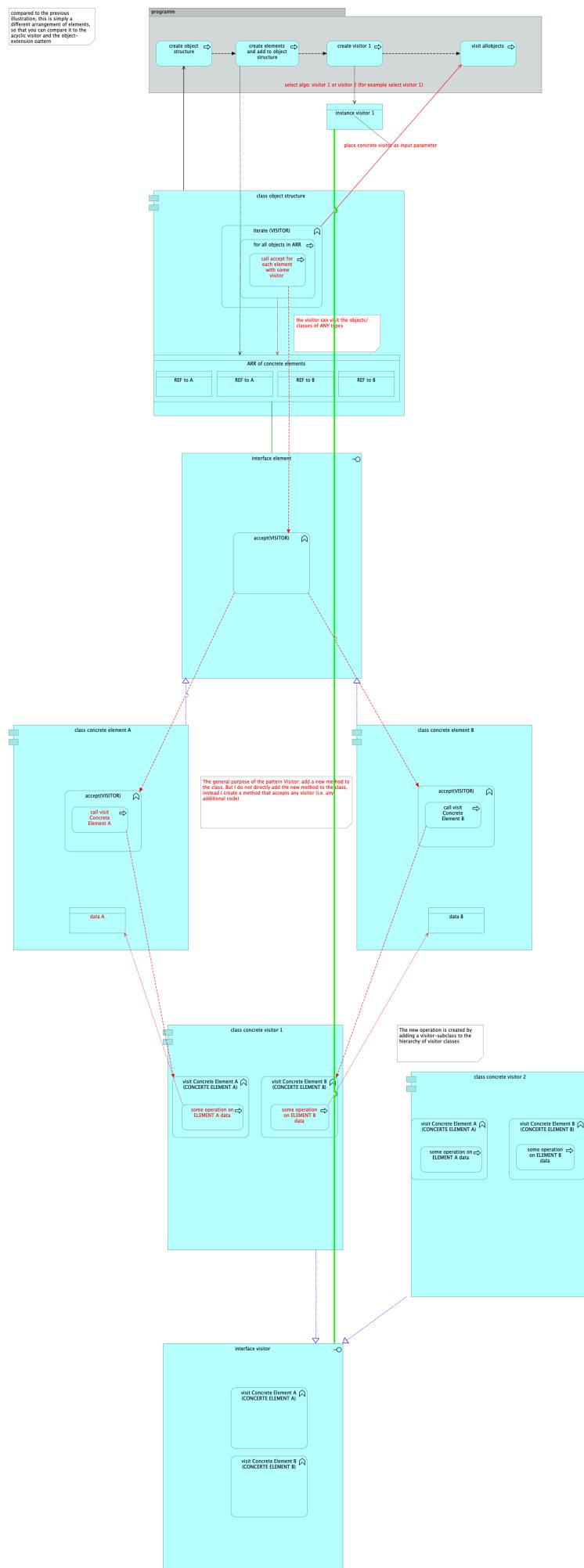


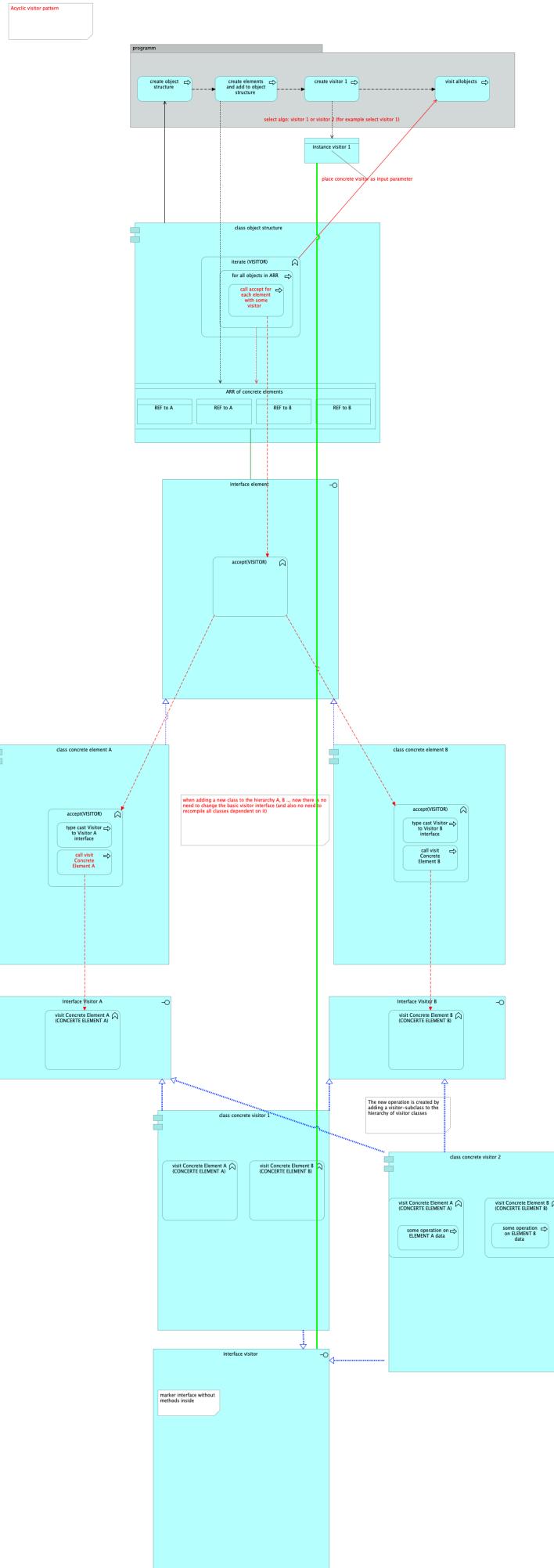
VISITOR



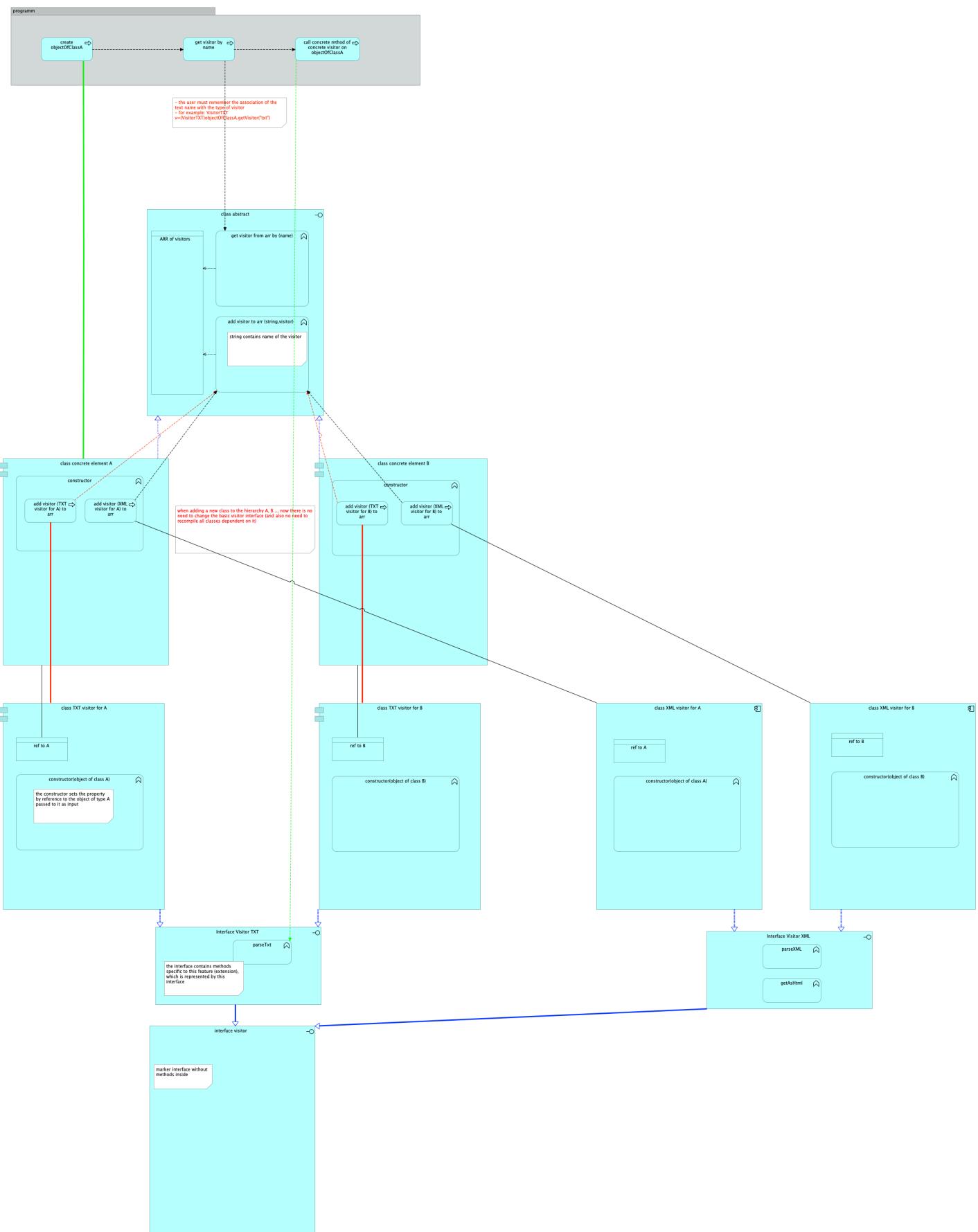


compared to the previous illustration, this one shows a different arrangement of elements, so that you can compare it to the application of the object-extension pattern.





Extension object pattern
(for explanation you can imagine extension as visitor)

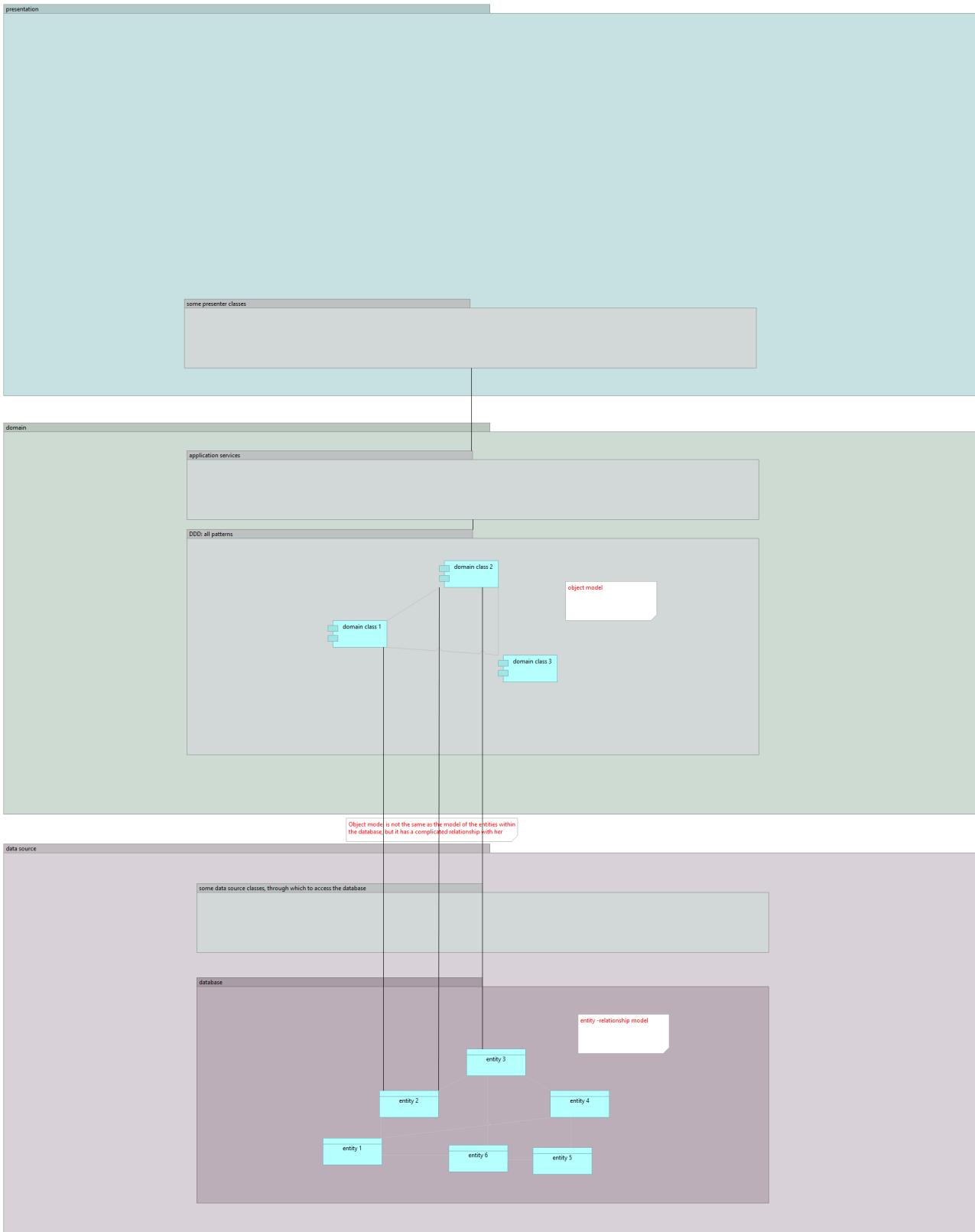


2 Enterprise Patterns

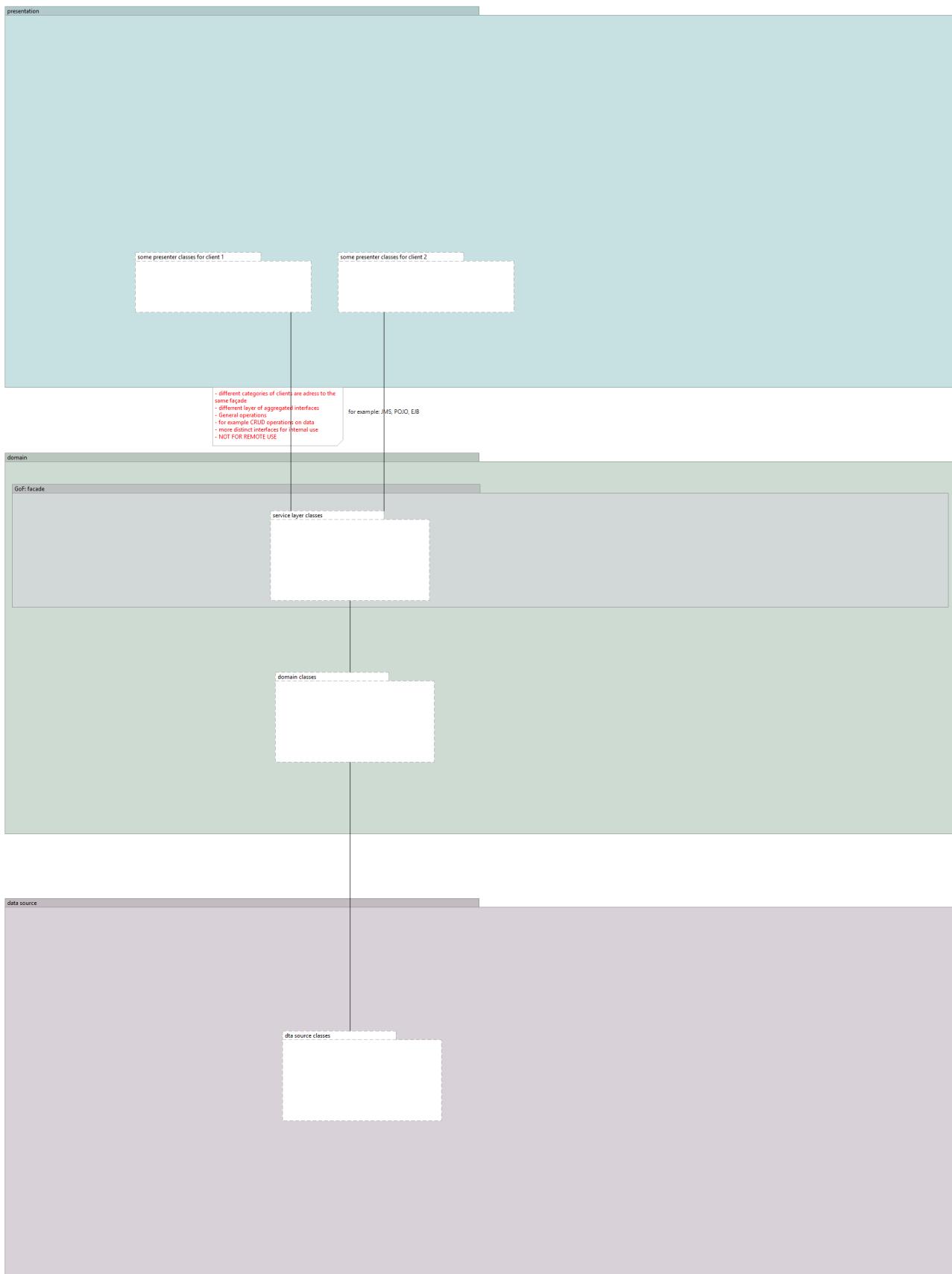


BUSINESS LOGIC

DOMAIN MODEL



SERVICE LAYER



TRANSACTION SCRIPT

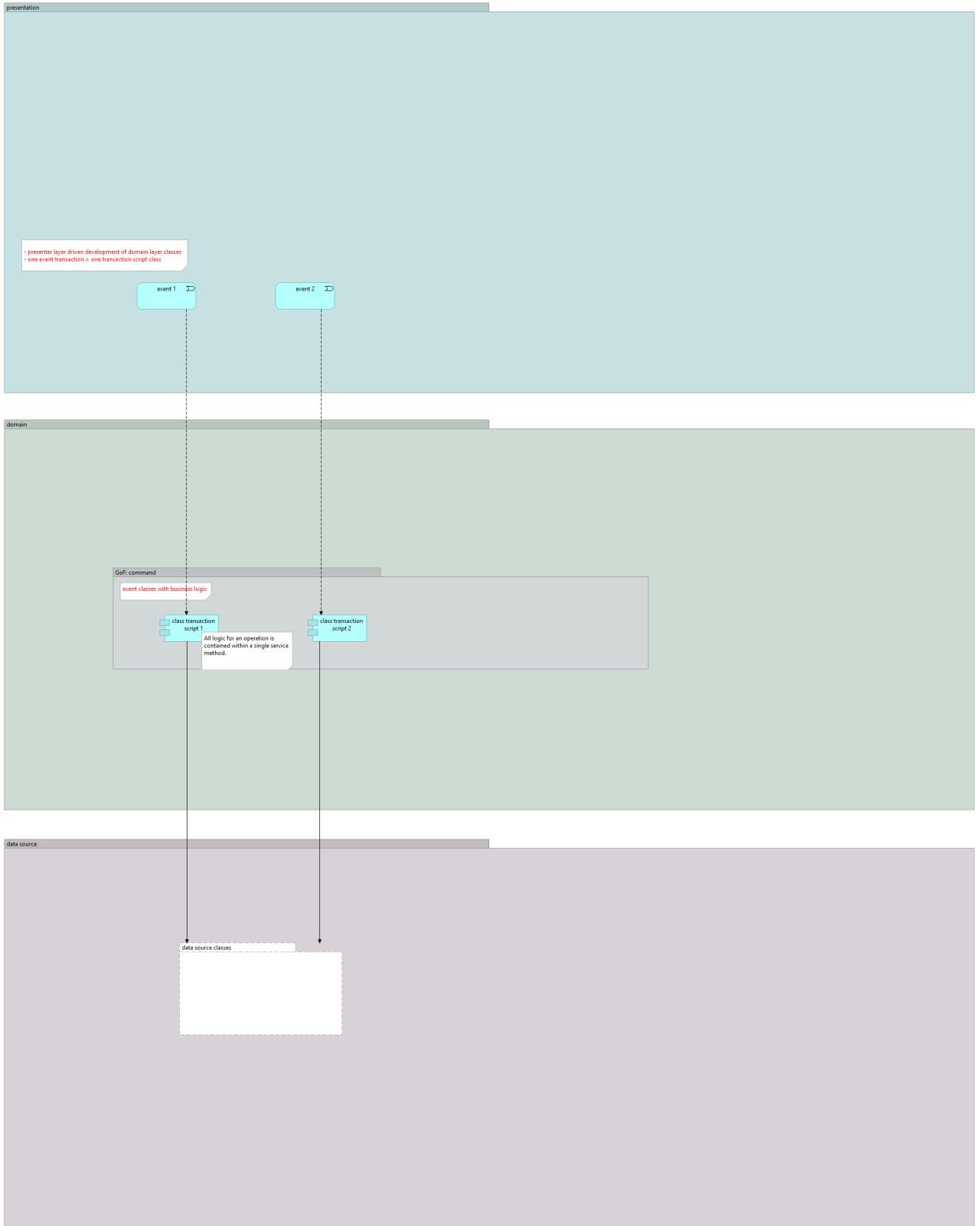
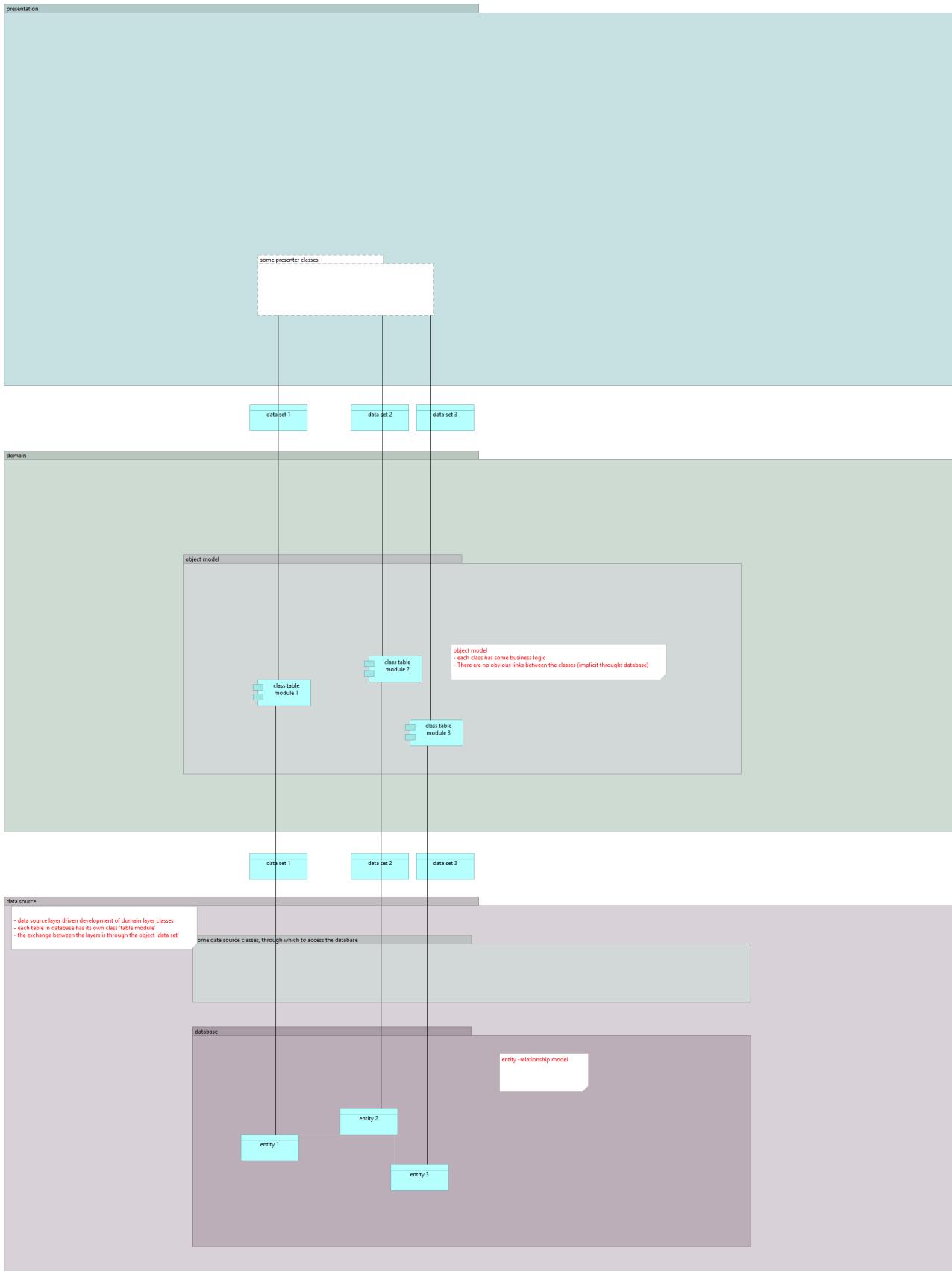
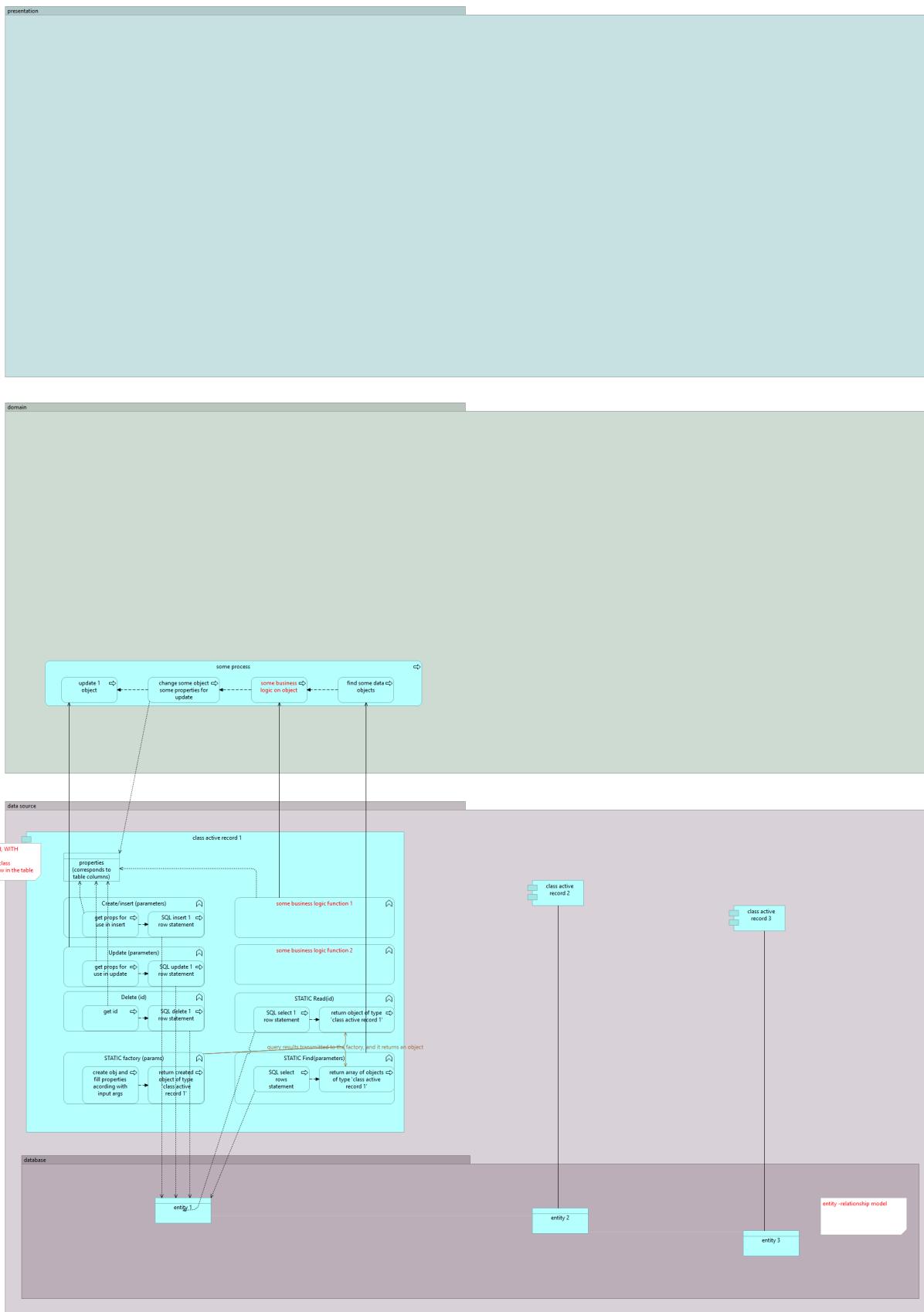


TABLE MODULE

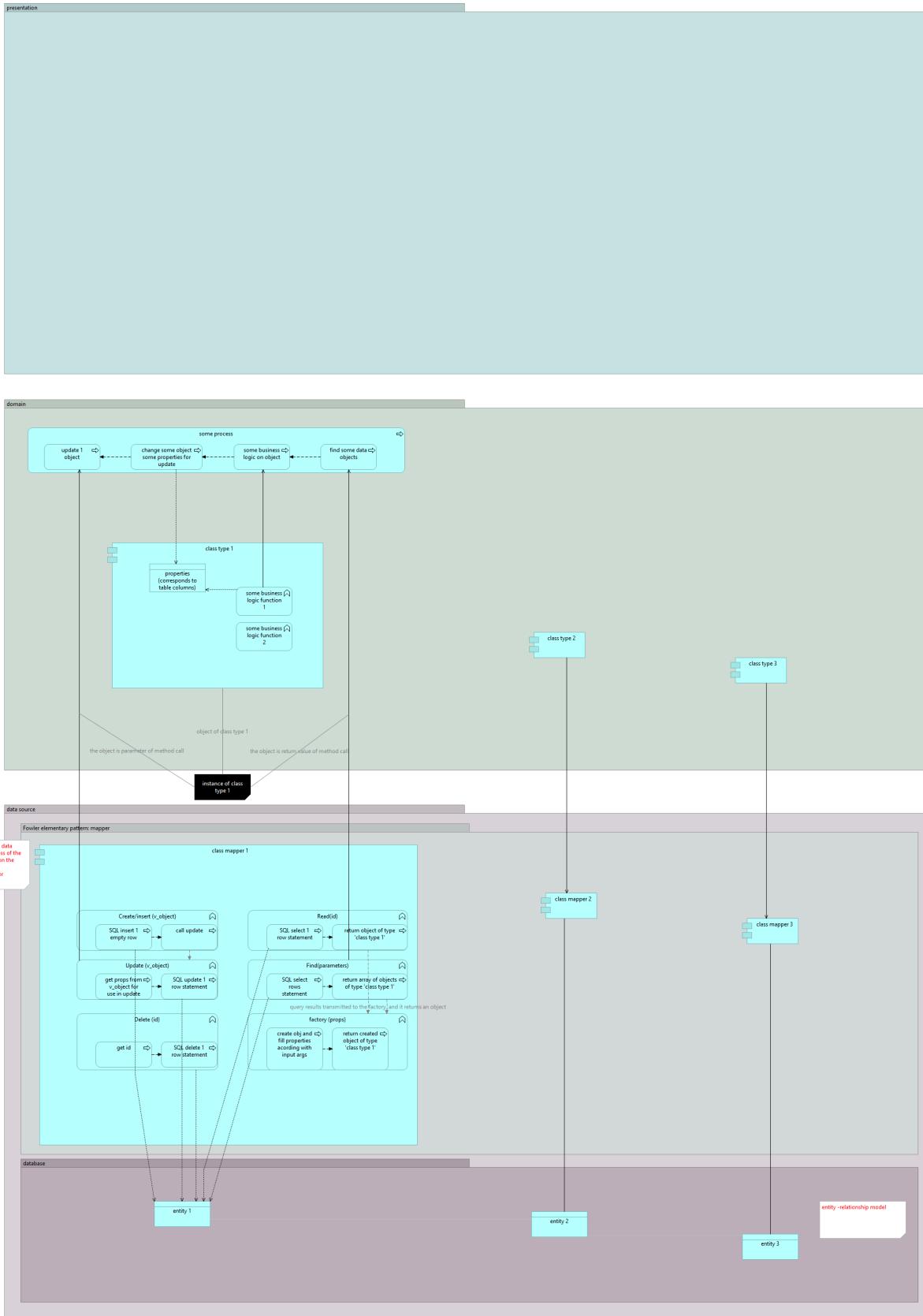


DATA SOURCES

ACTIVE RECORD



DATA MAPPER



ROW DATA GATEWAY

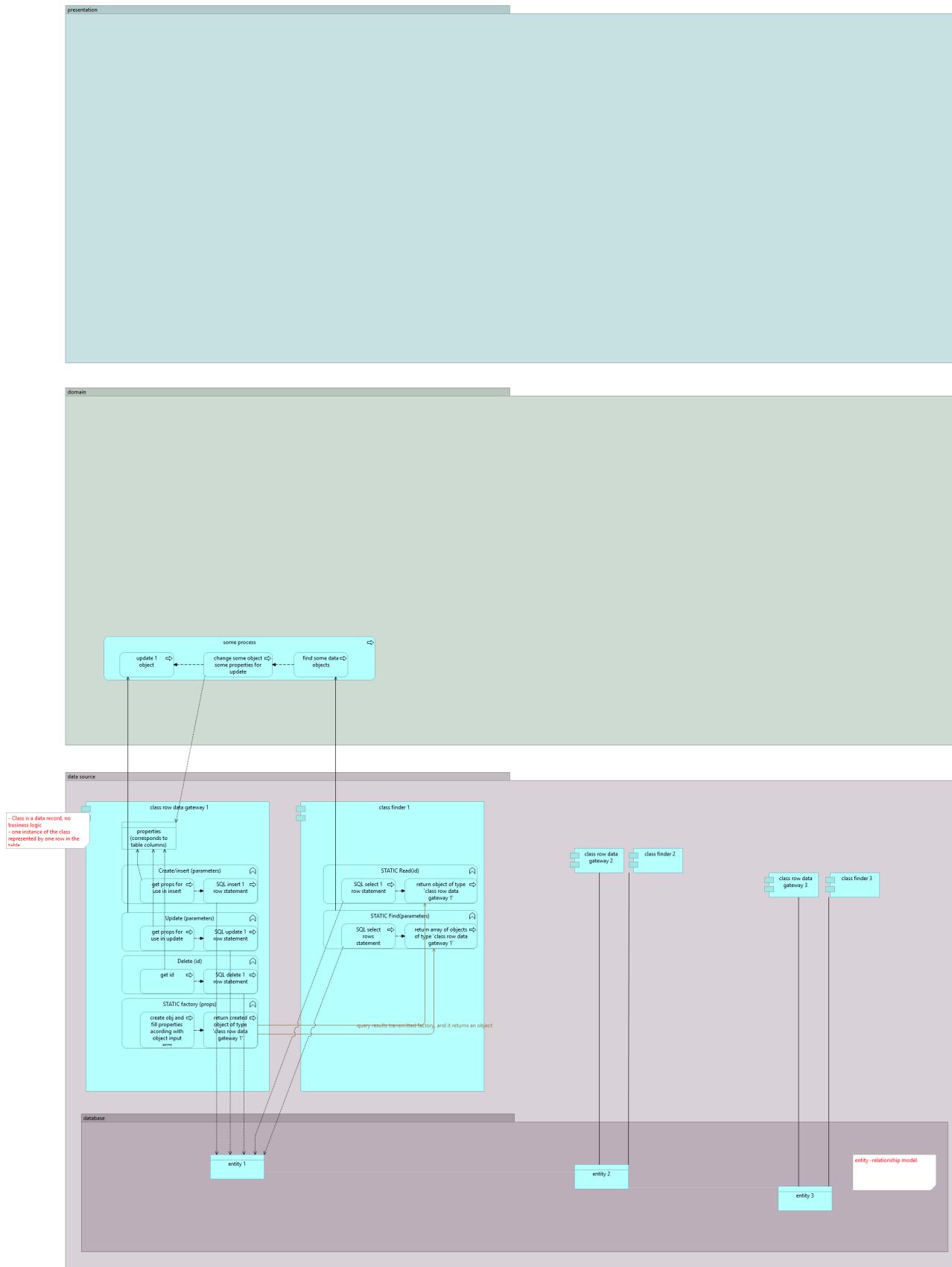
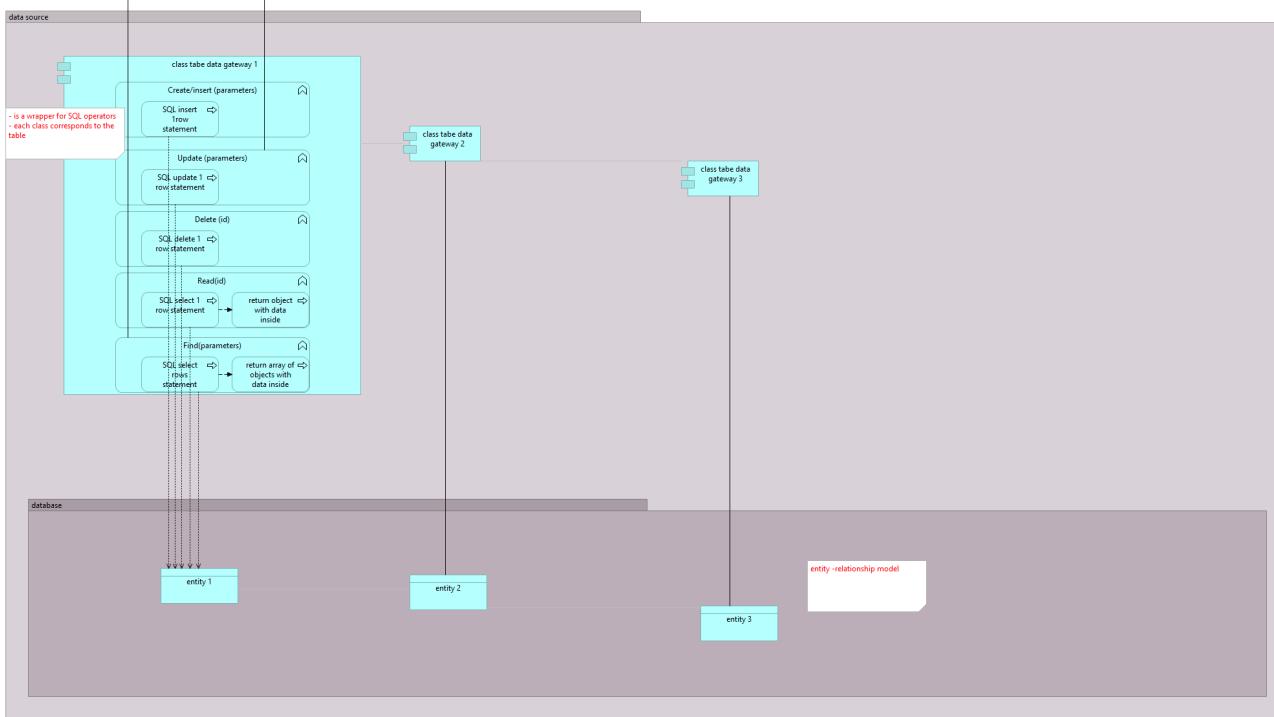
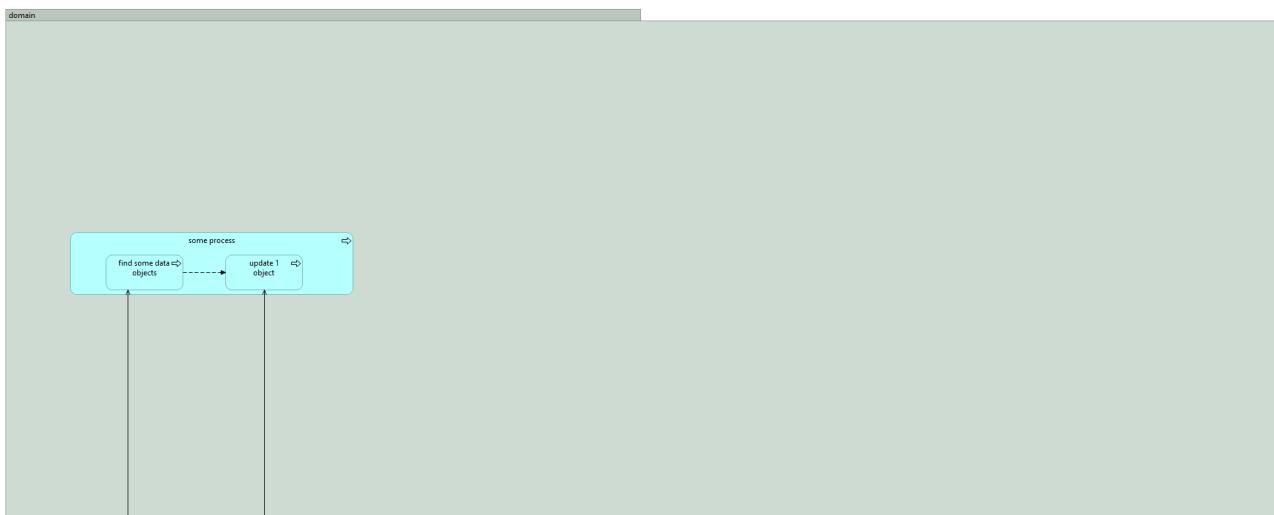
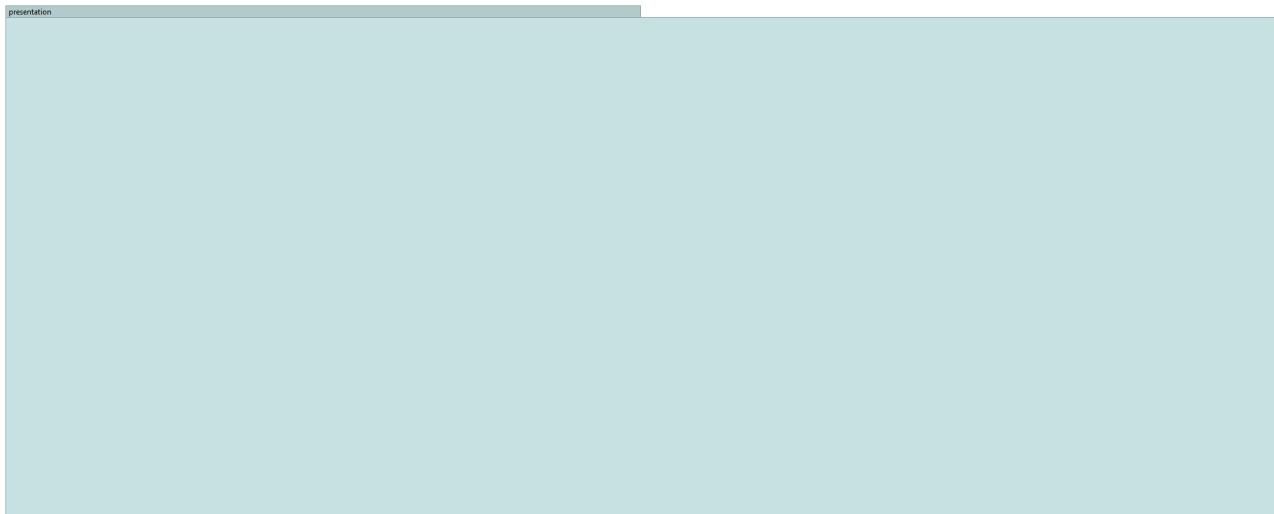
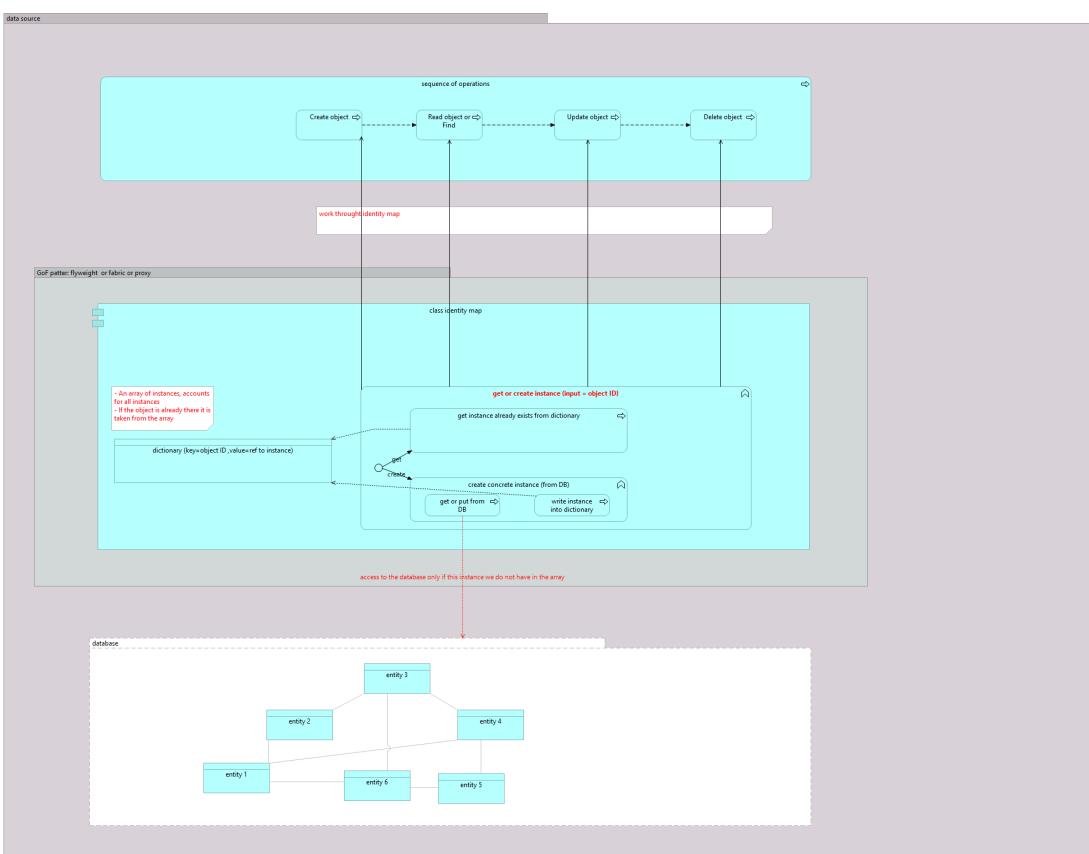
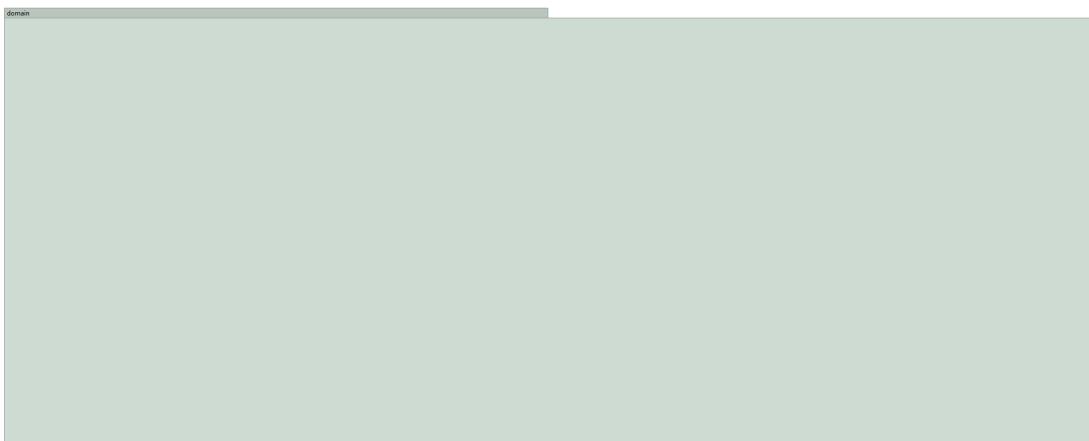
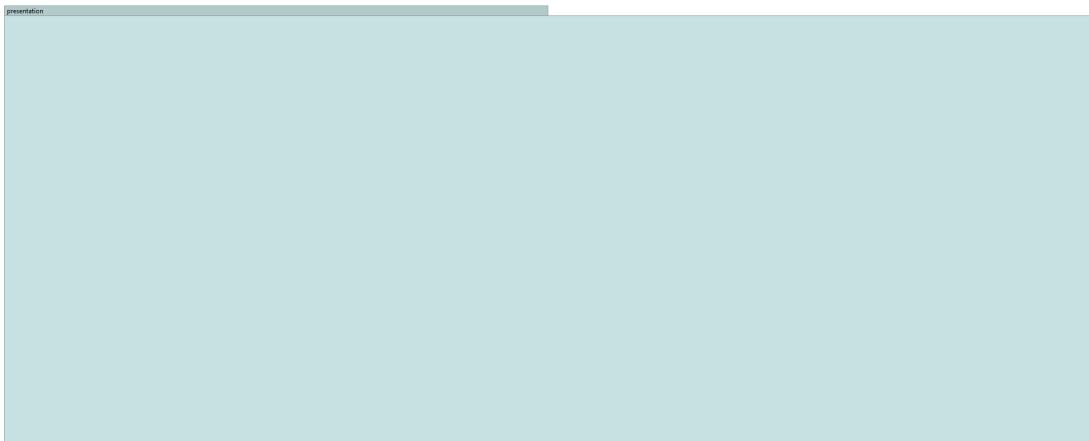


TABLE DATA GATEWAY

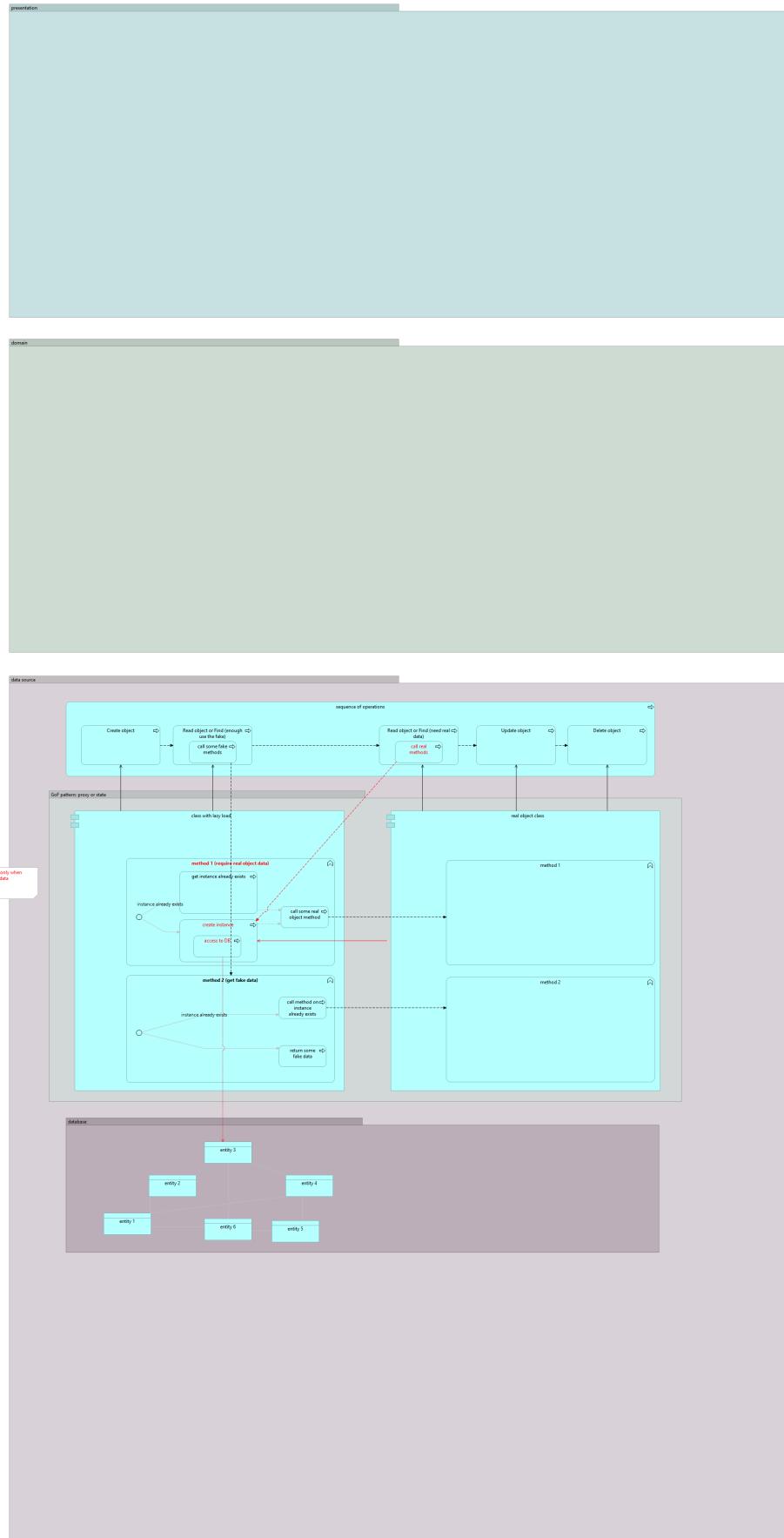


MODELLING BEHAVIOUR

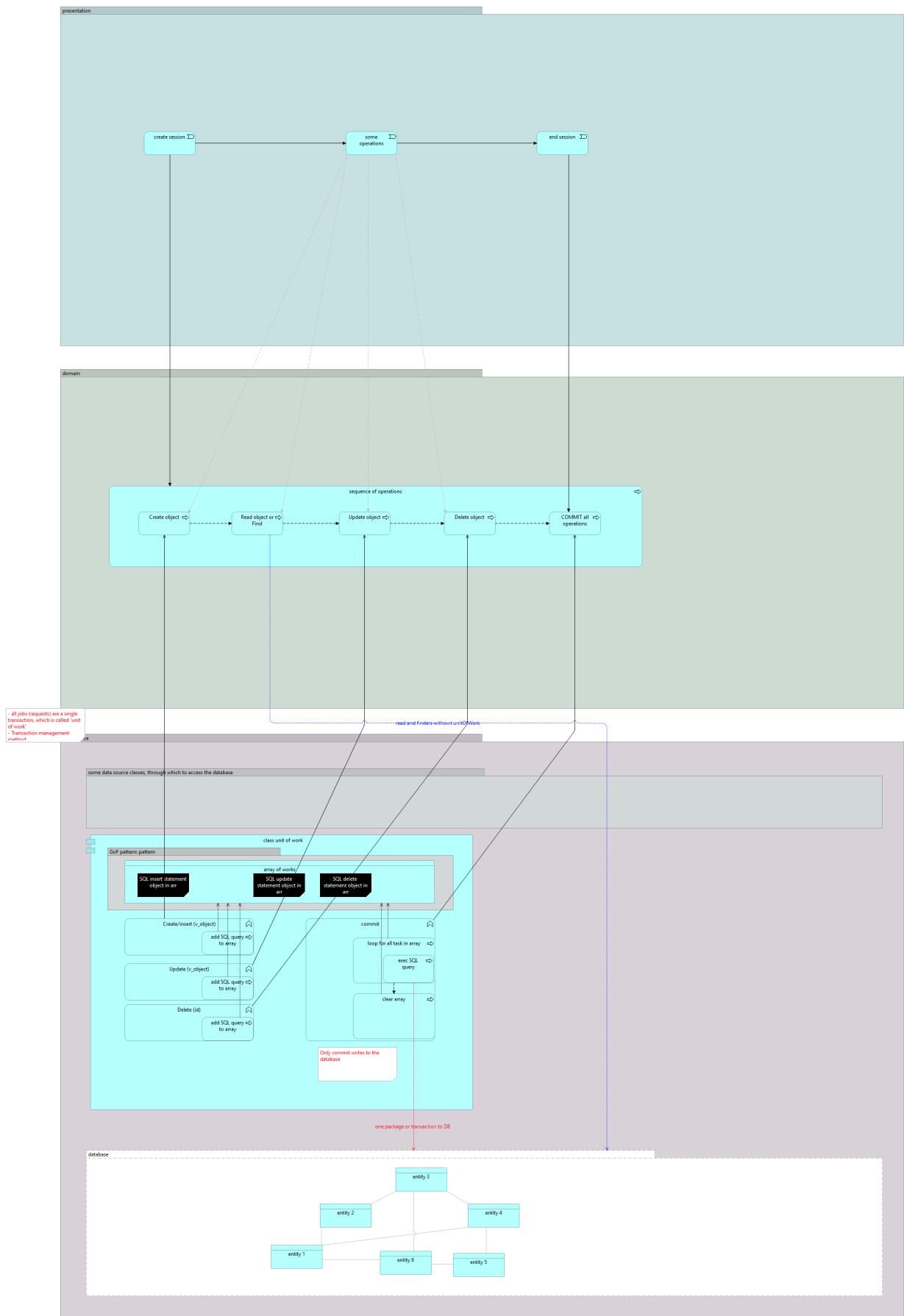
IDENTITY MAP



LAZY LOAD

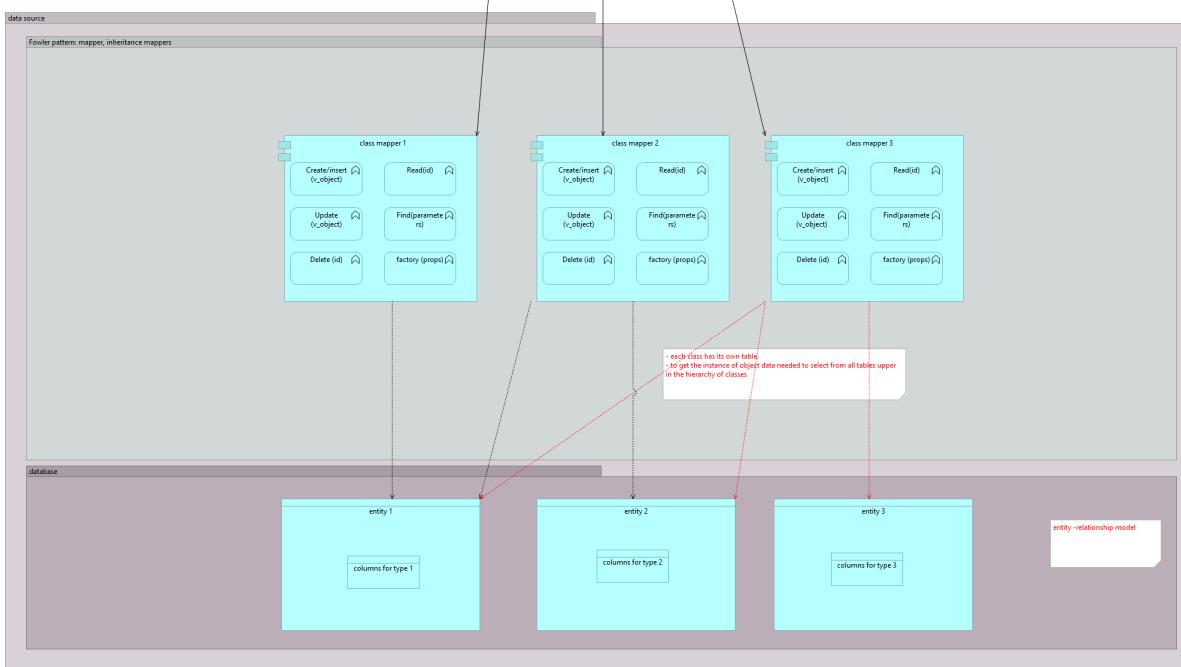
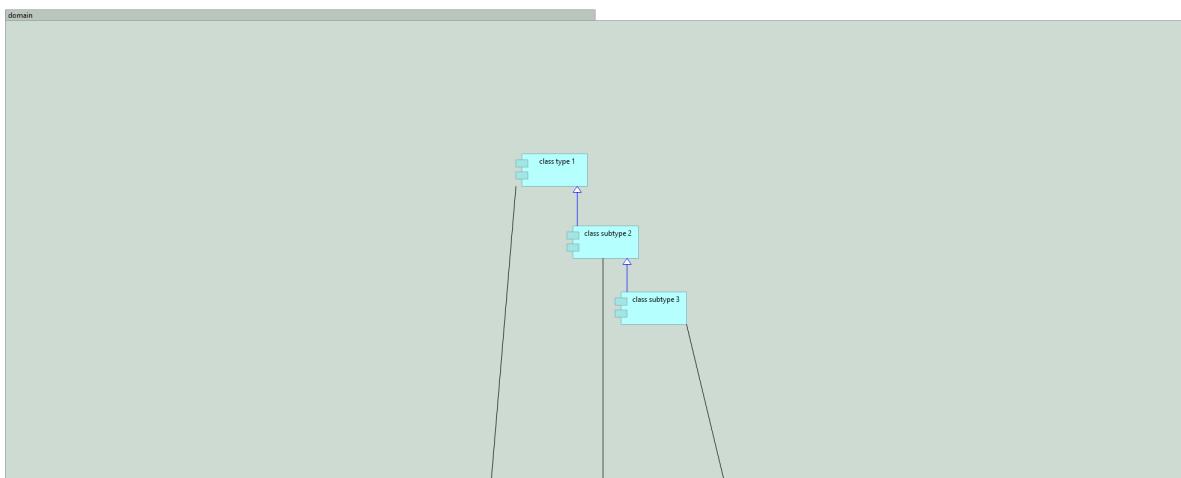
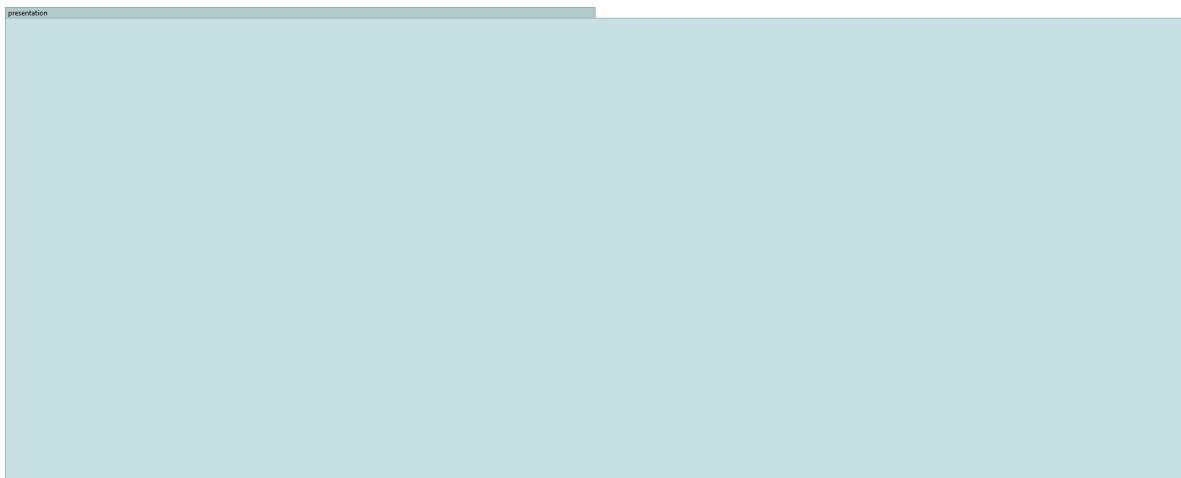


UNIT OF WORK.

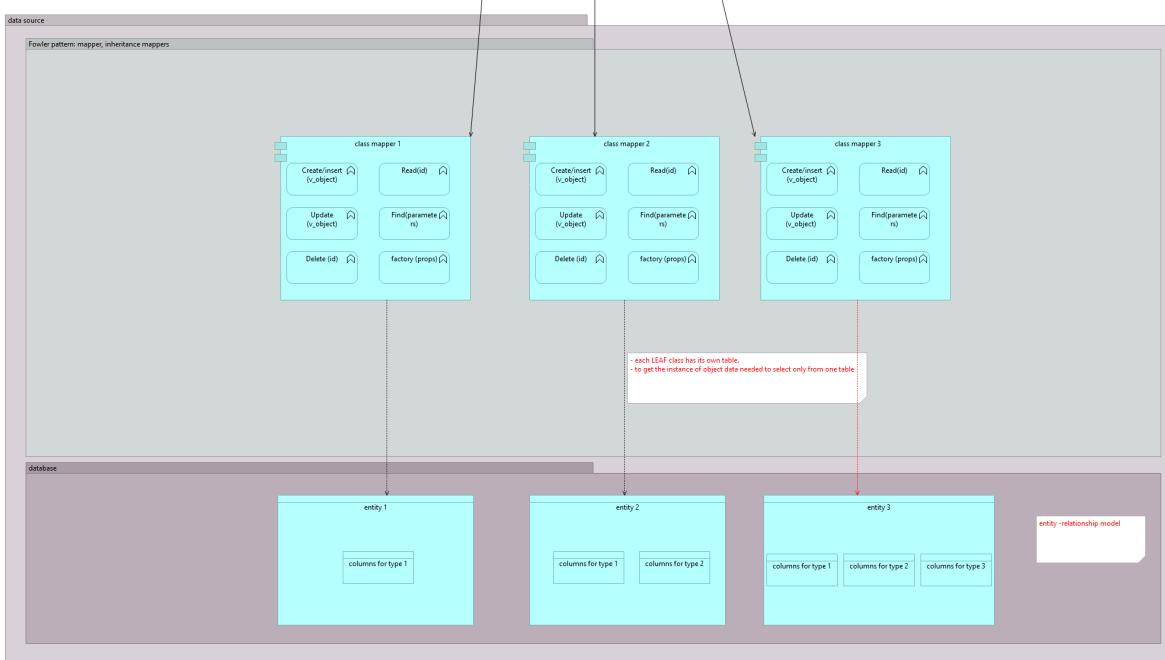
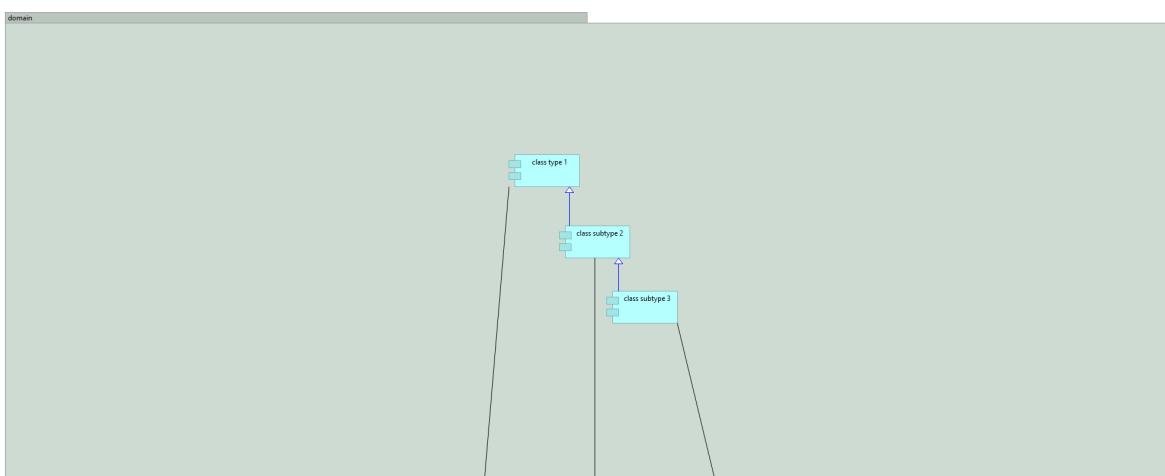
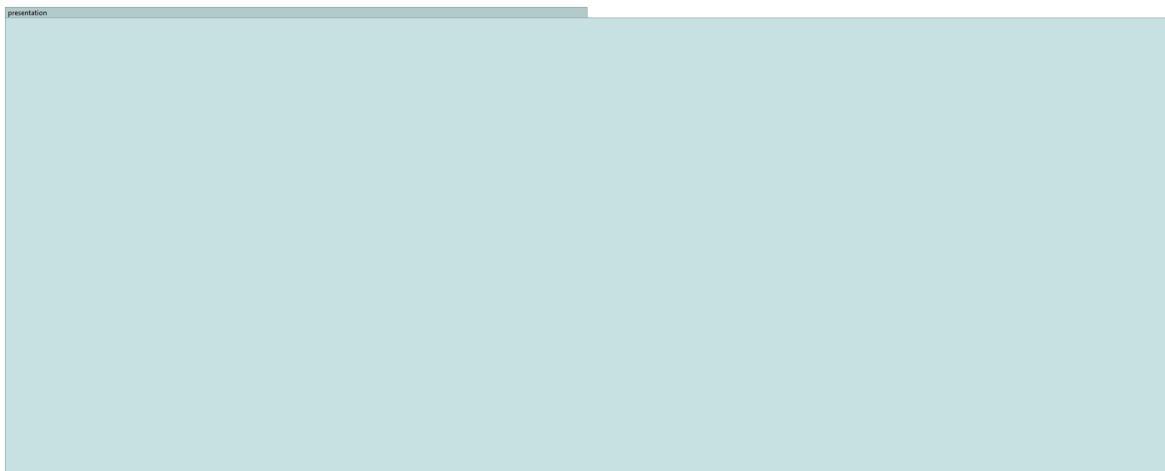


MODELLING STRUCTURE HIERARCHY

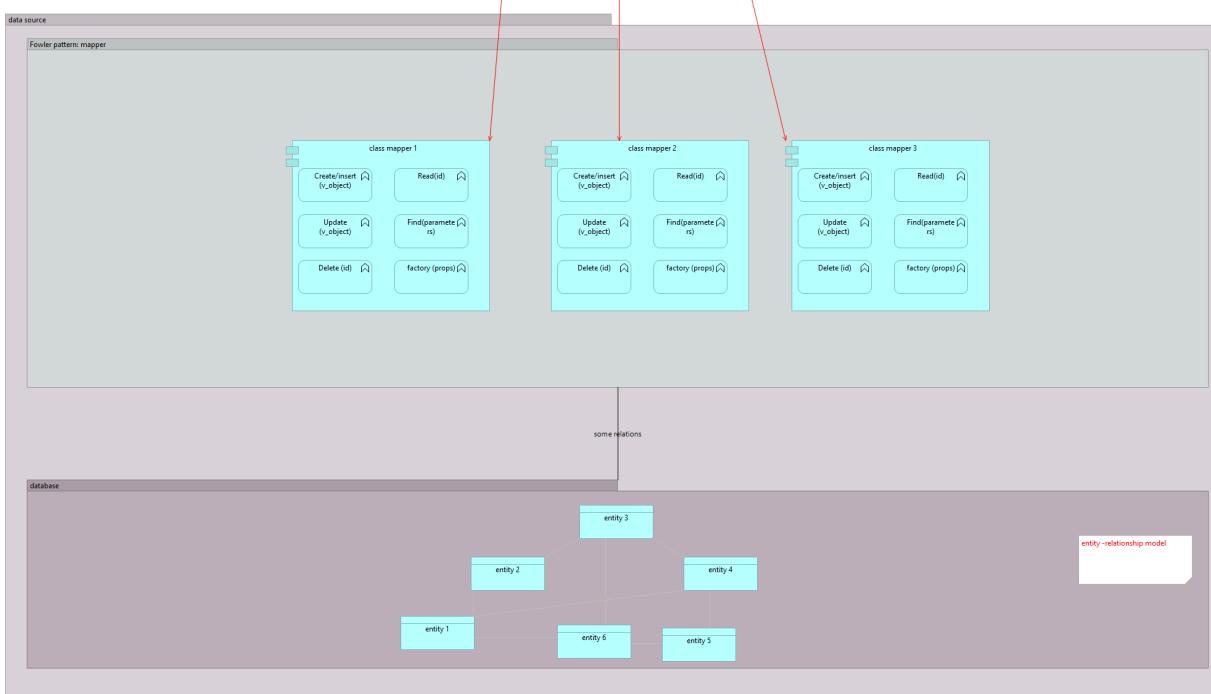
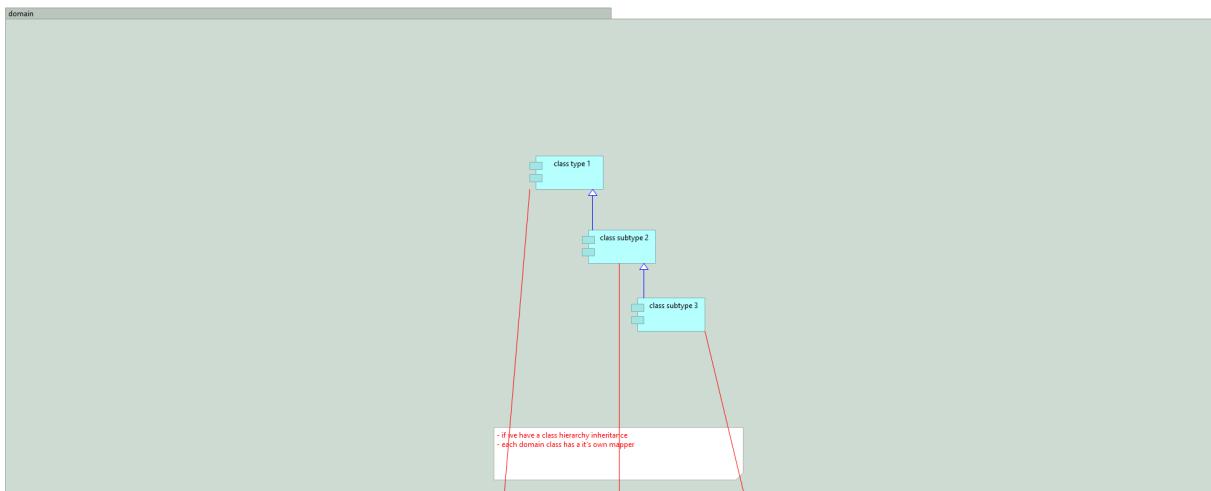
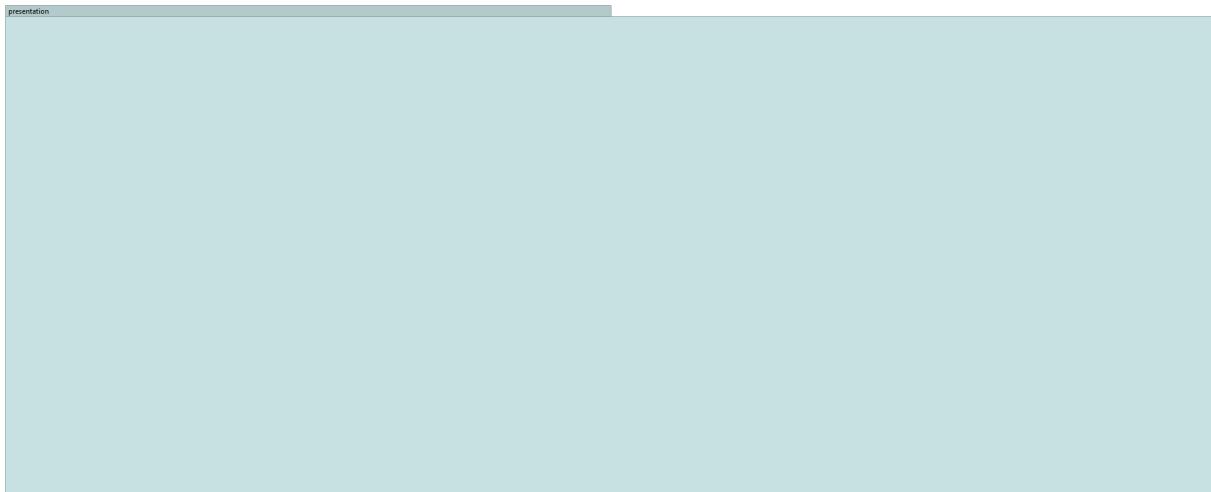
CLASS TABLE INHERITANCE



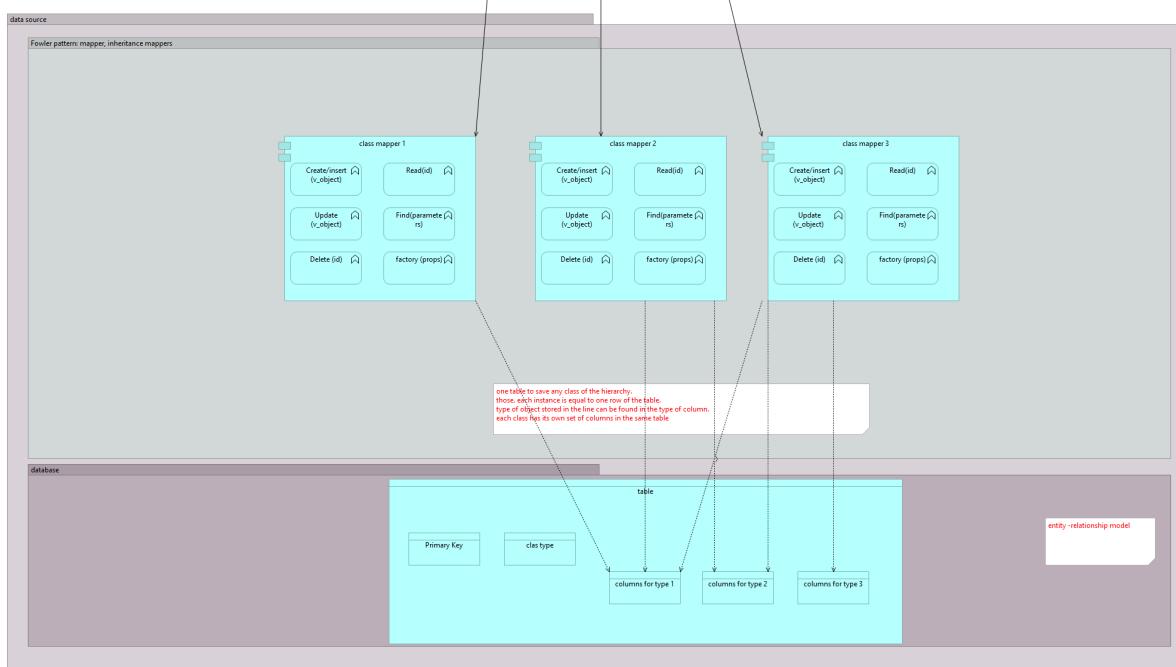
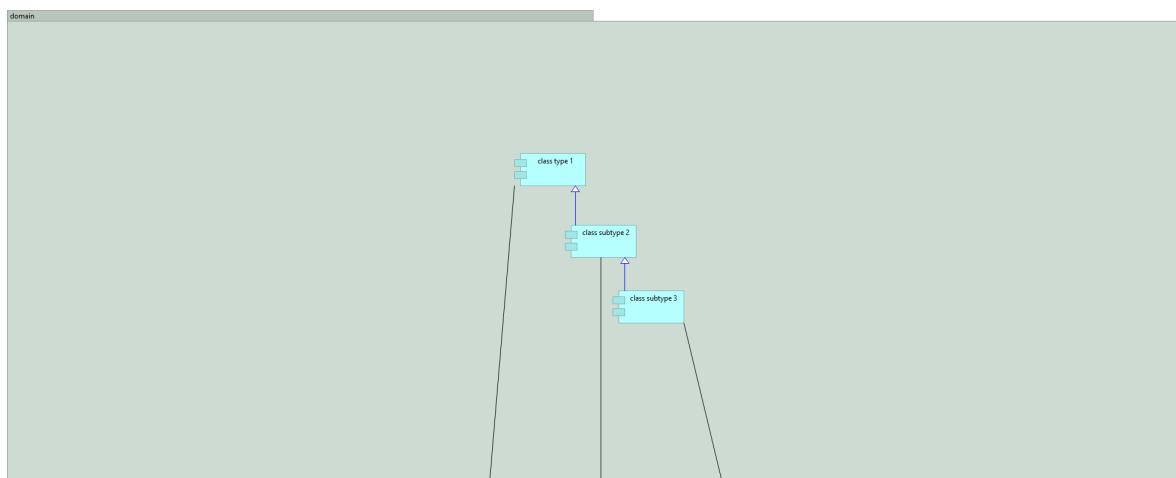
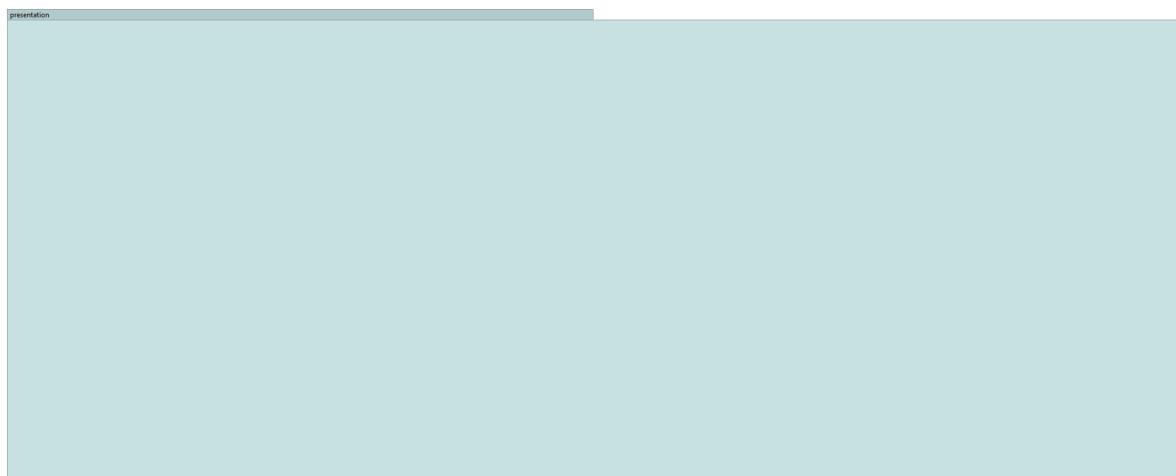
CONCRETE TABLE INHERITANCE



INHERITANCE MAPPERS

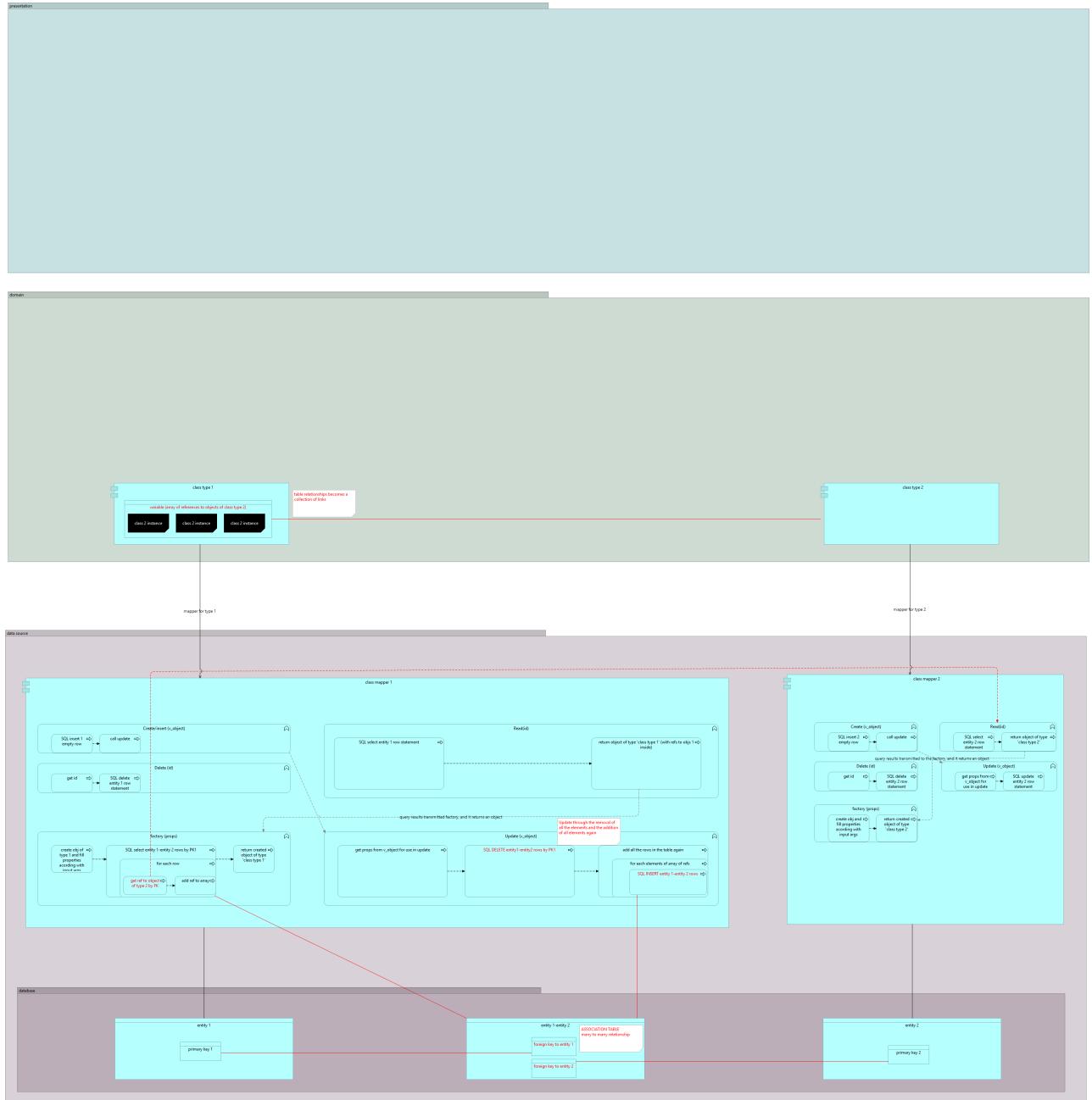


SINGLE TABLE INHERITANCE

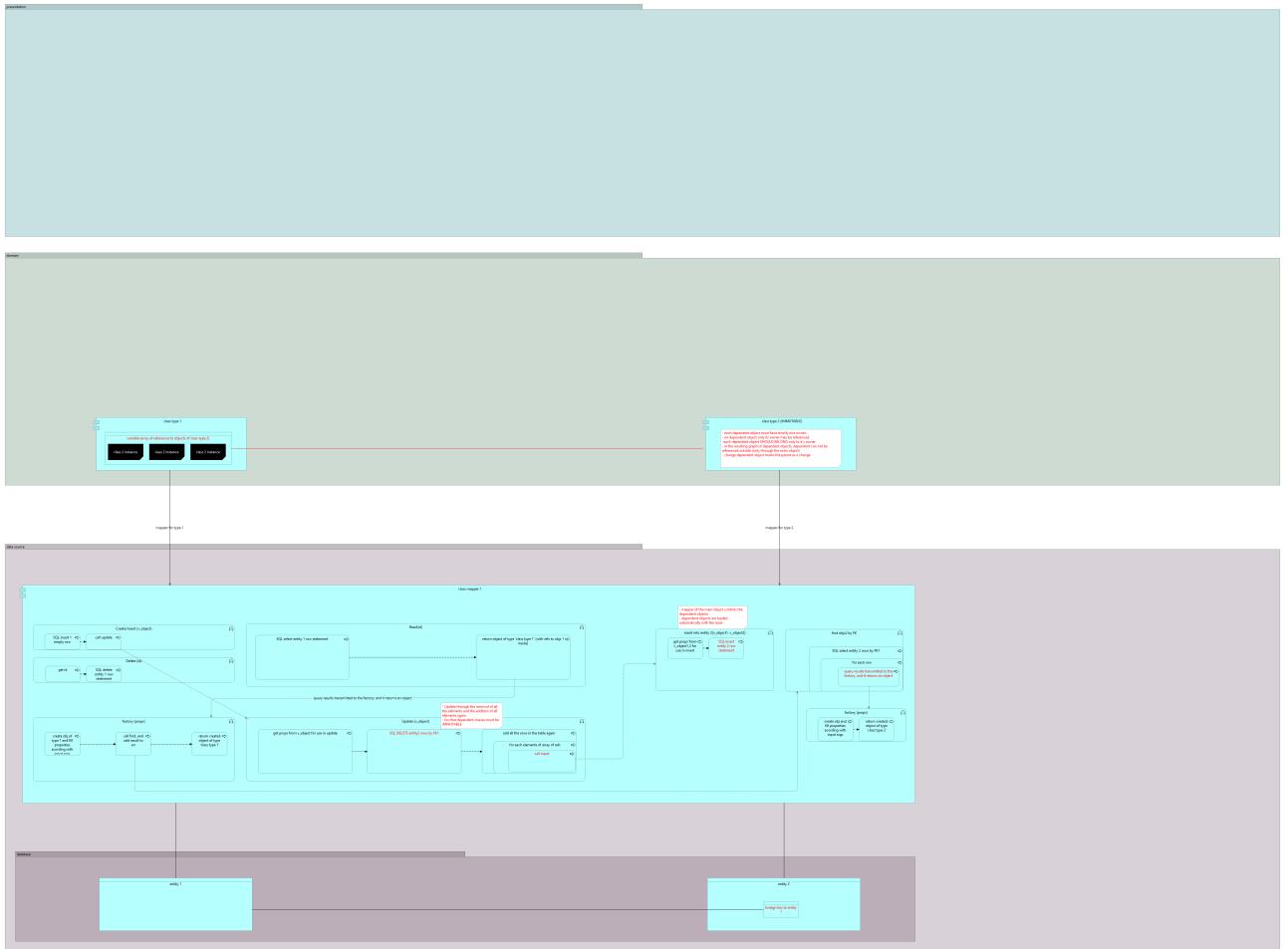


MODELLING STRUCTURE RELATIONS

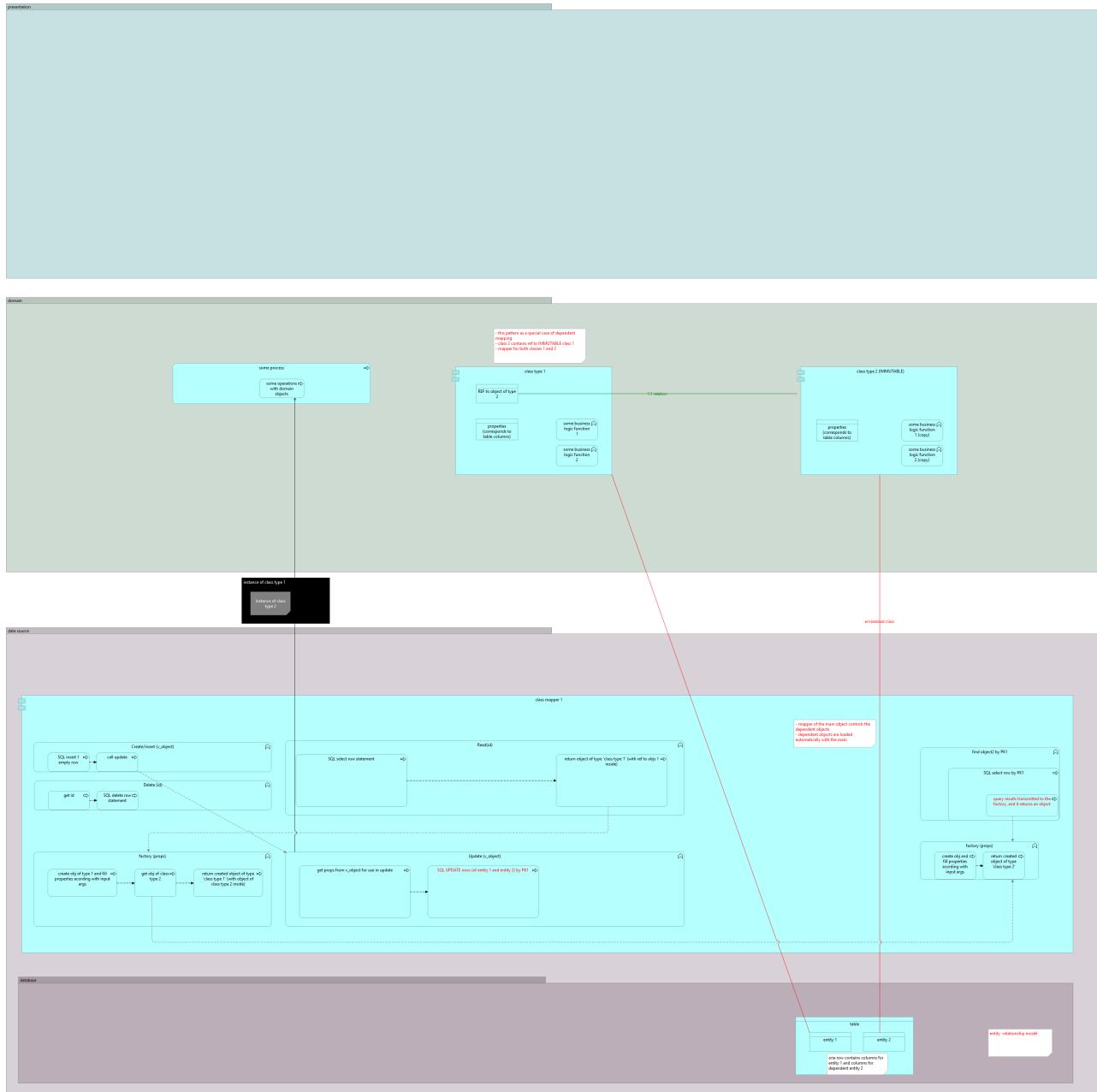
ASSOCIATION TABLE MAPPING



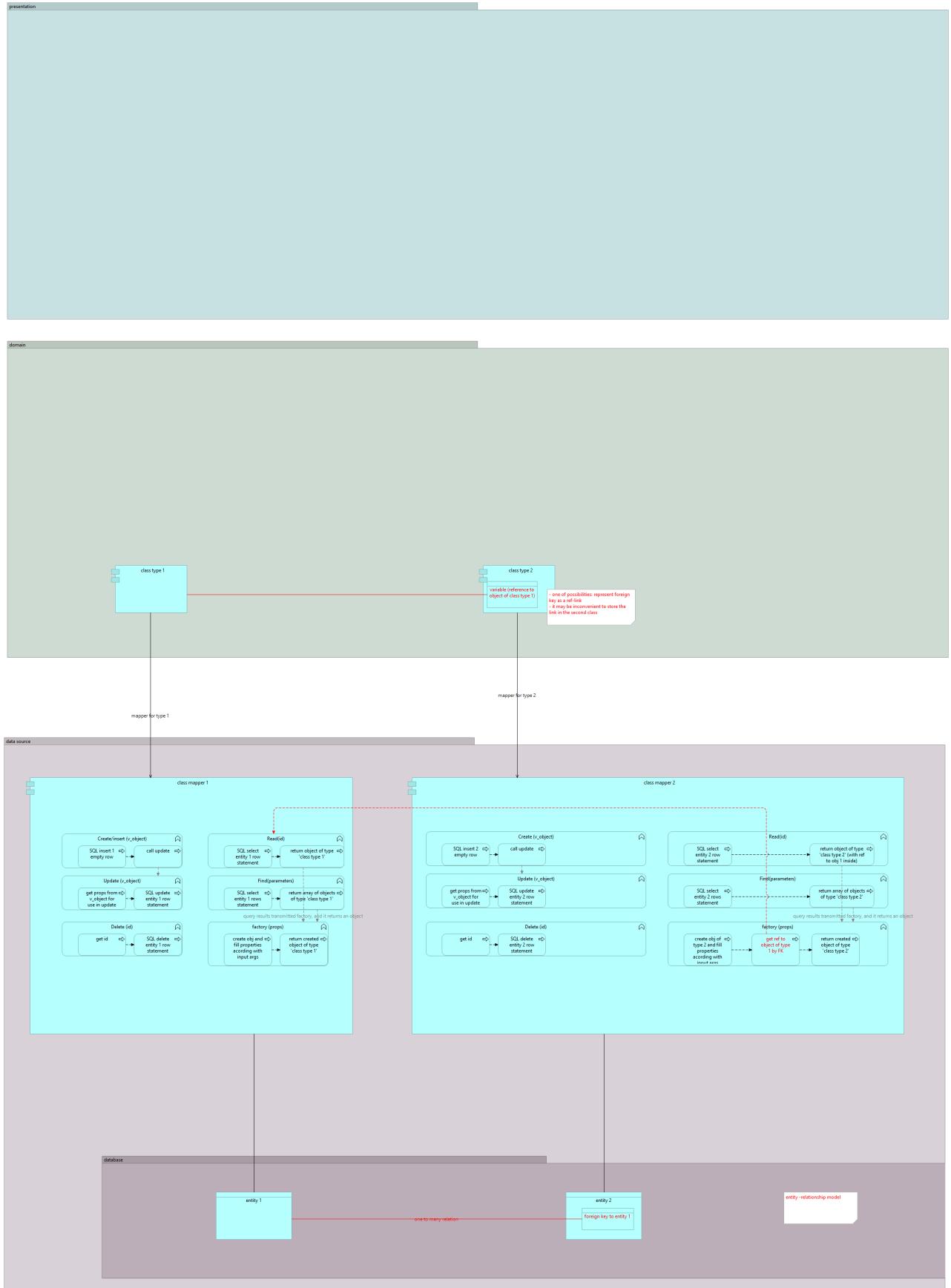
DEPENDENT MAPPING



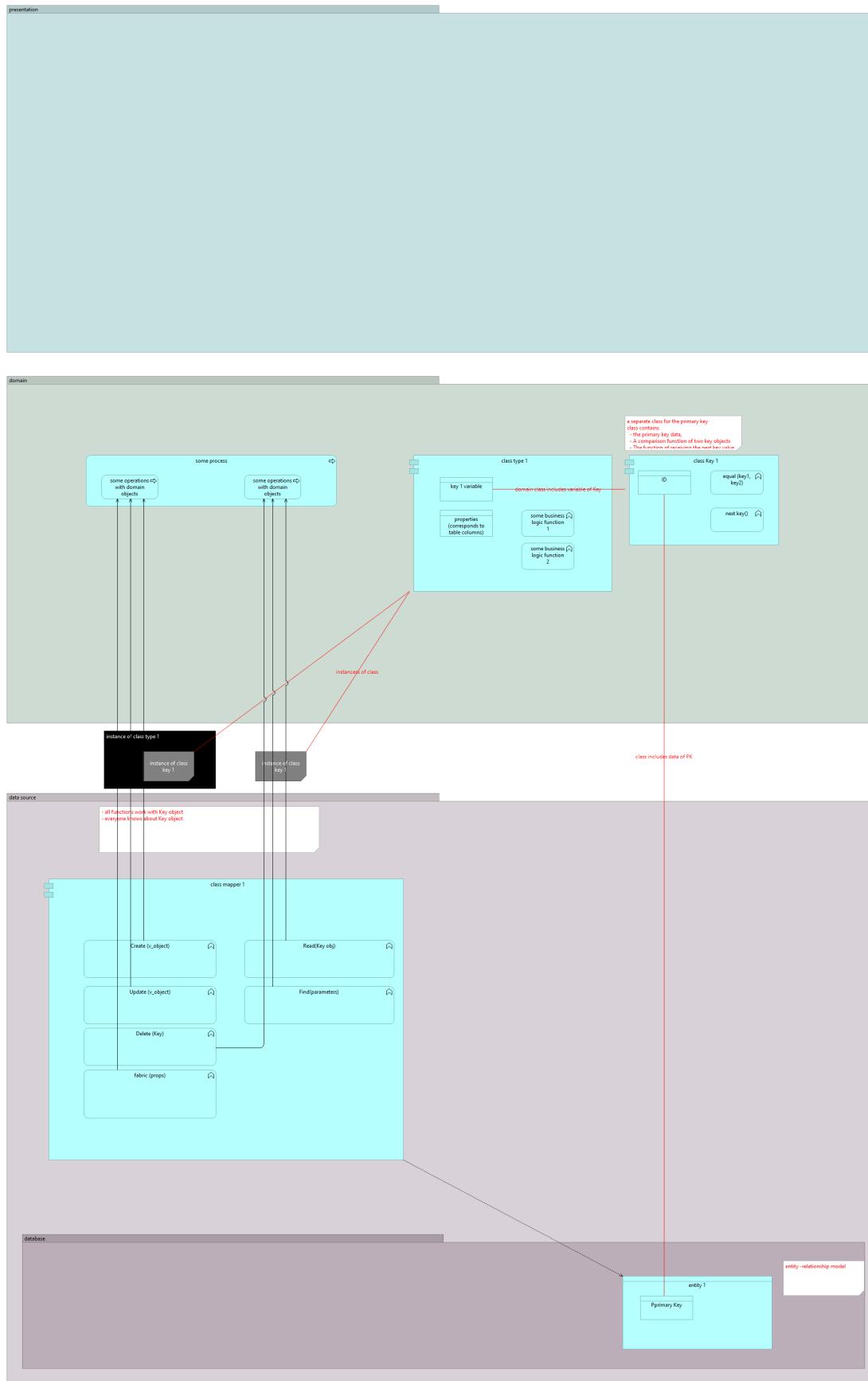
EMBEDDED VALUE



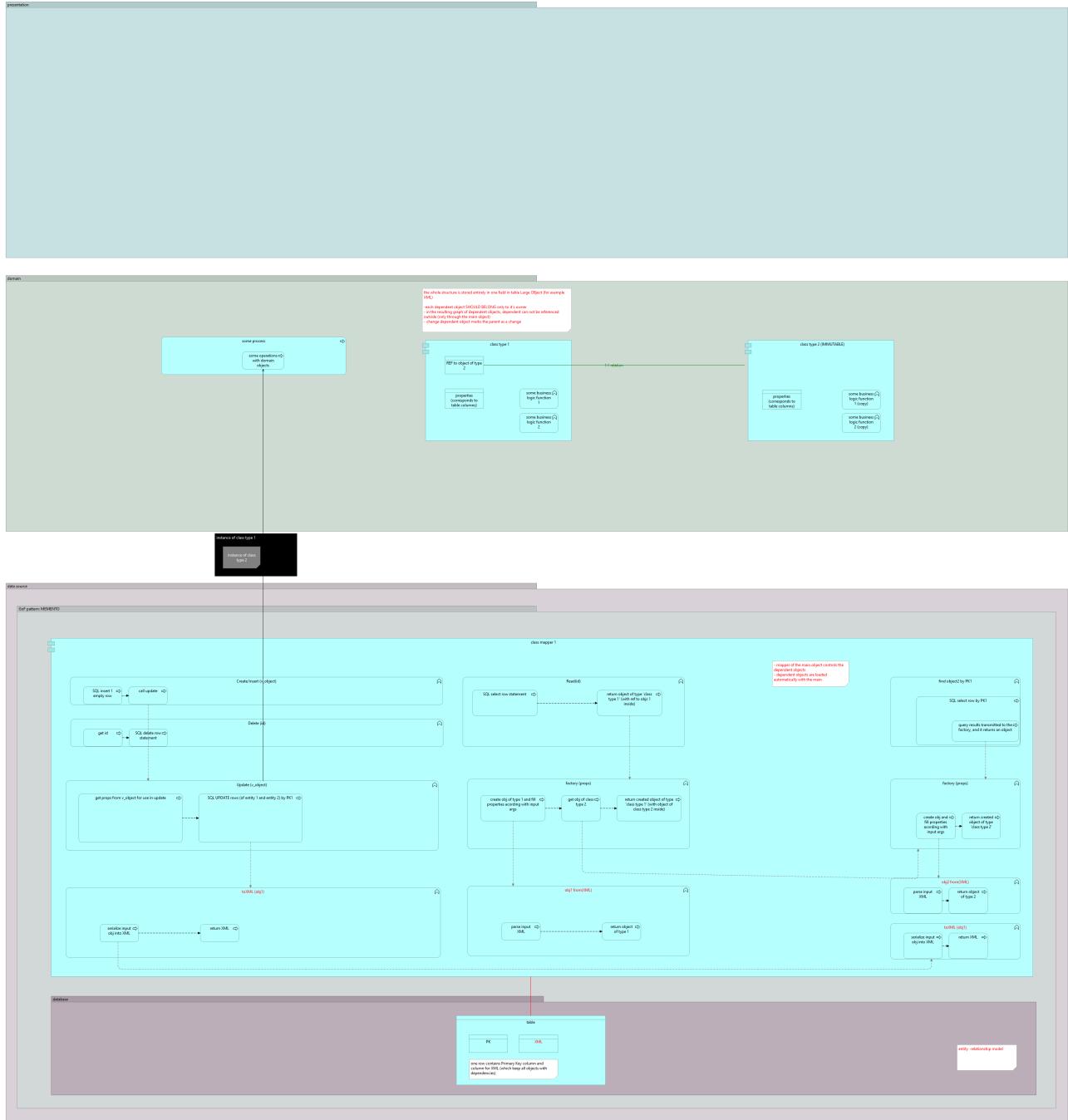
FOREIGN KEY MAPPING



IDENTITY FIELD

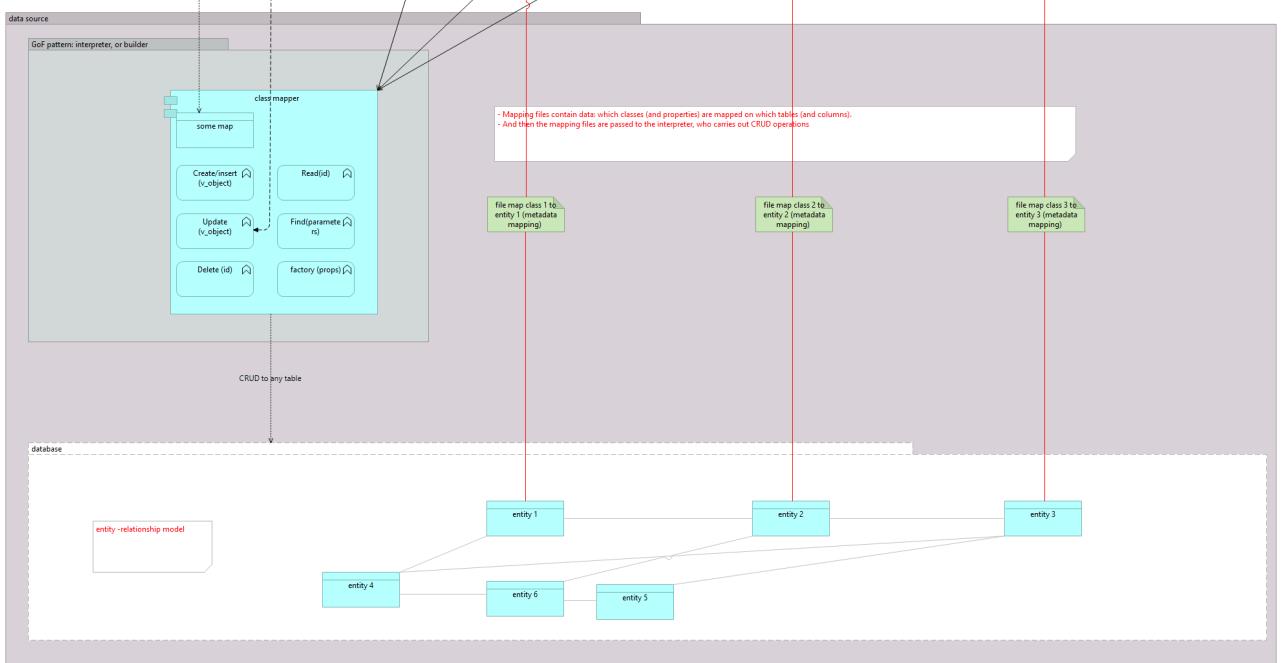
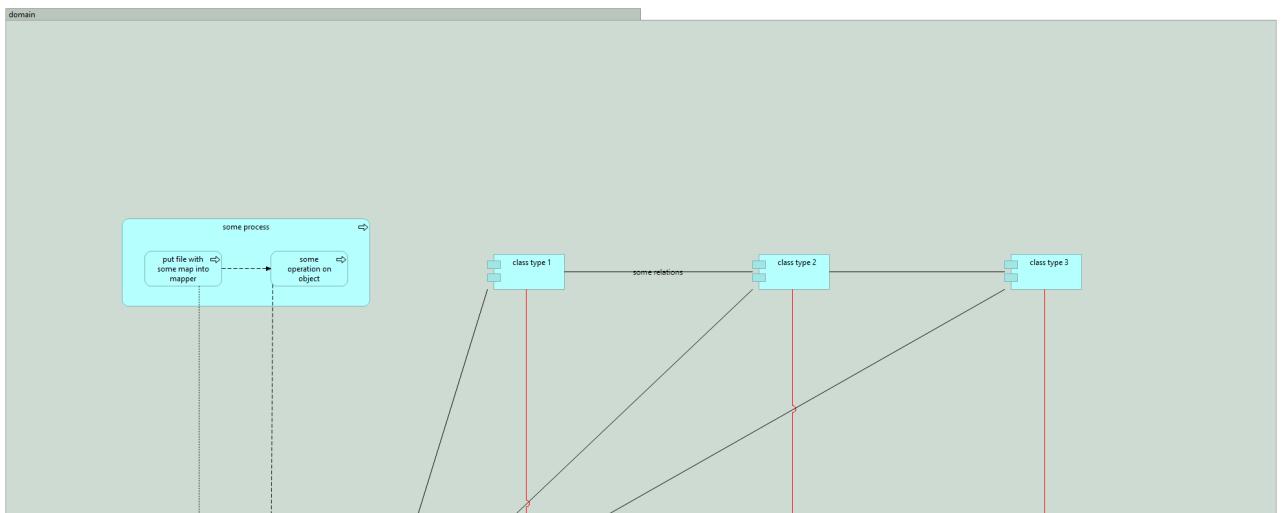
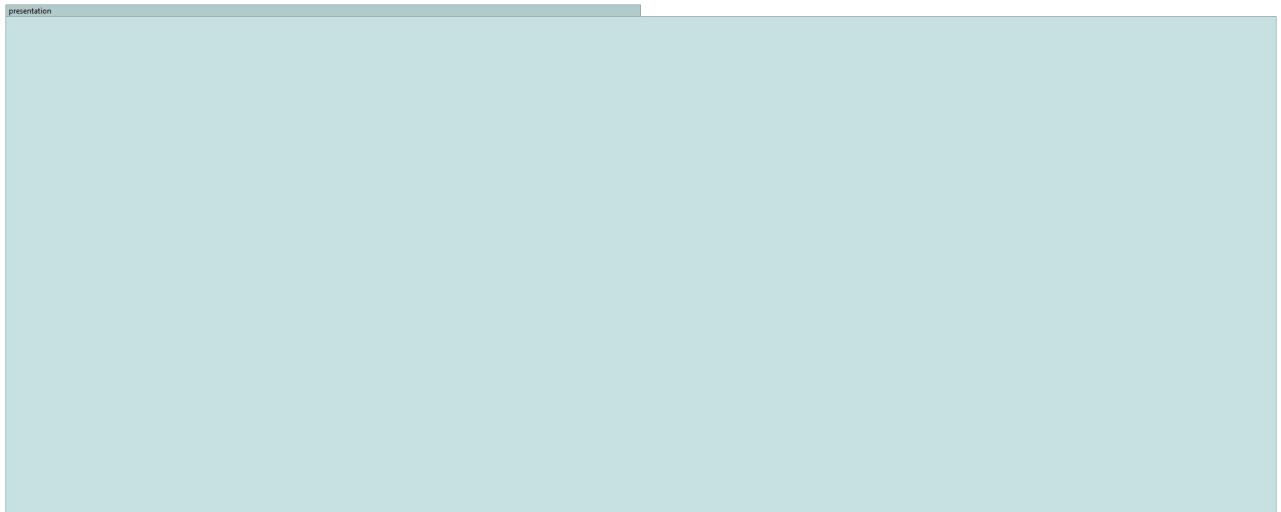


SERIALIZED LOB

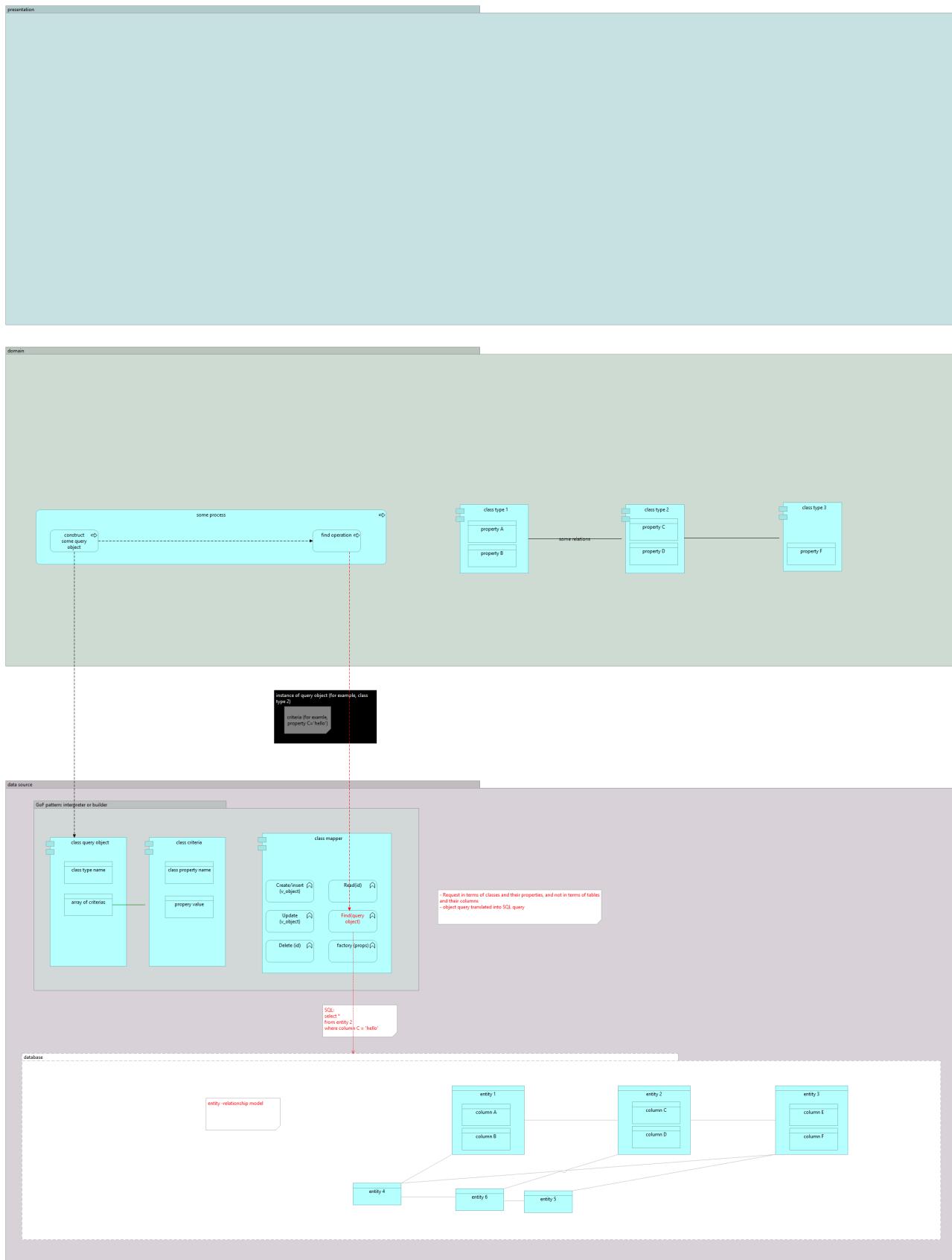


METADATA

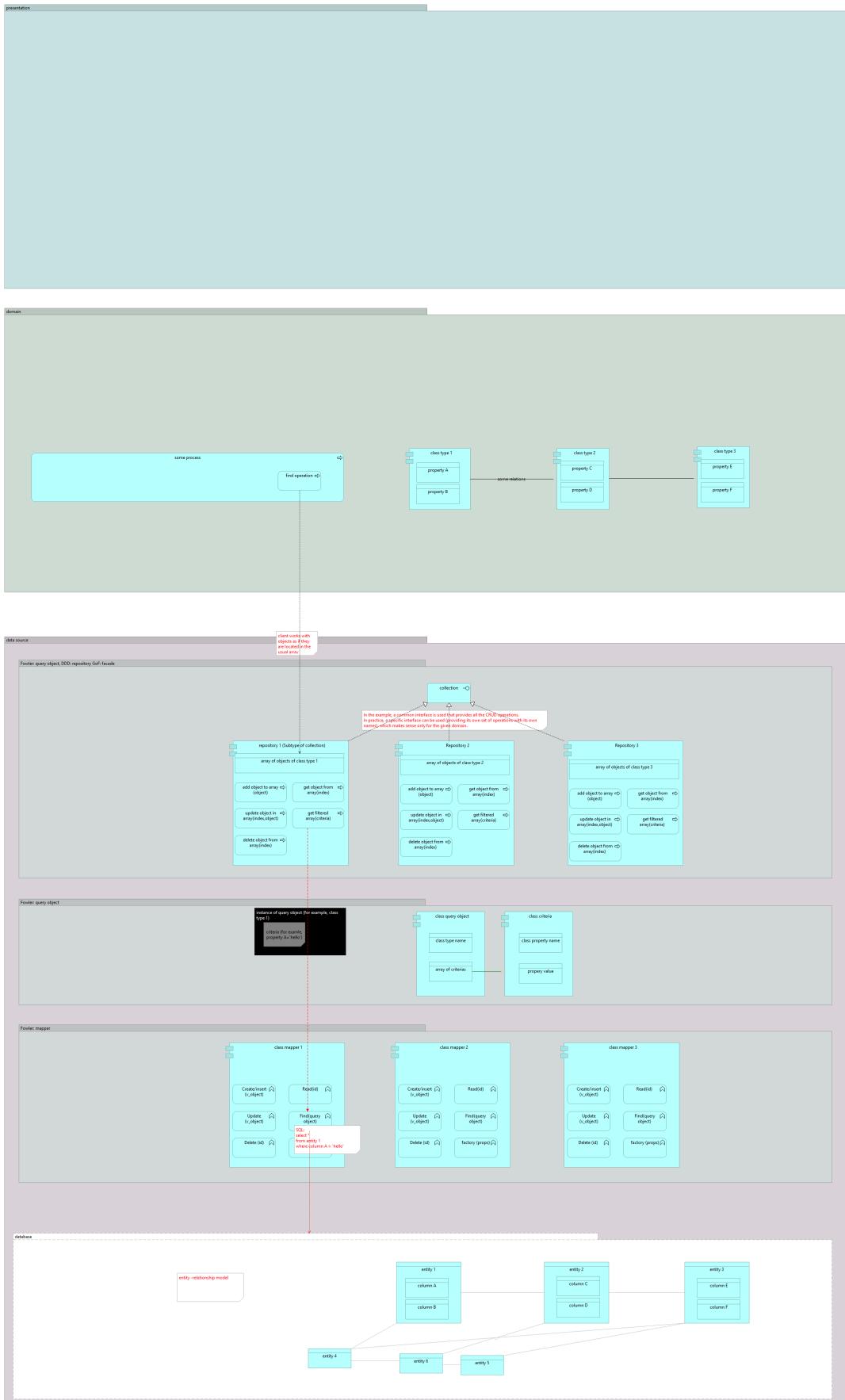
METADATA MAPPING



QUERY OBJECT

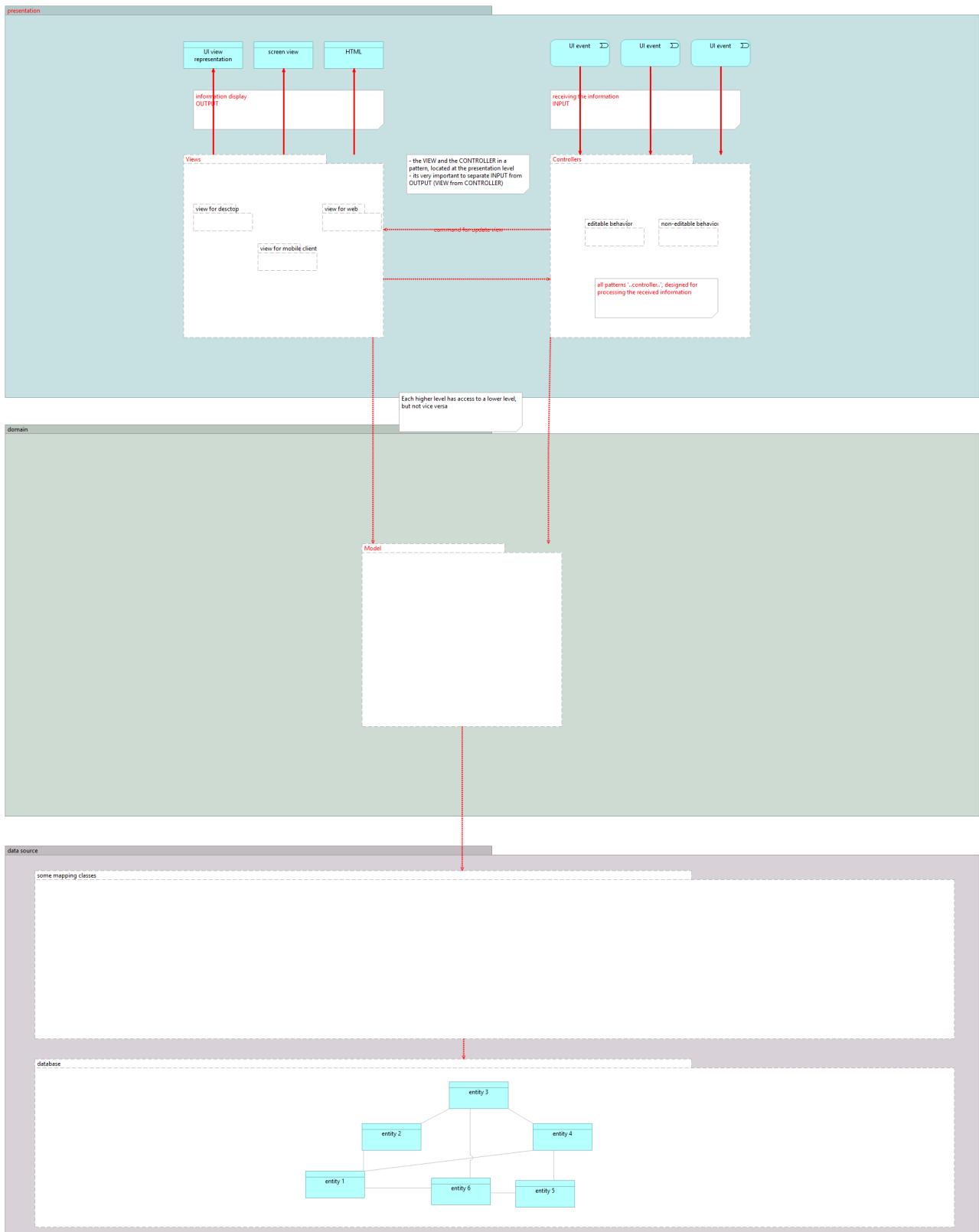


REPOSITORY

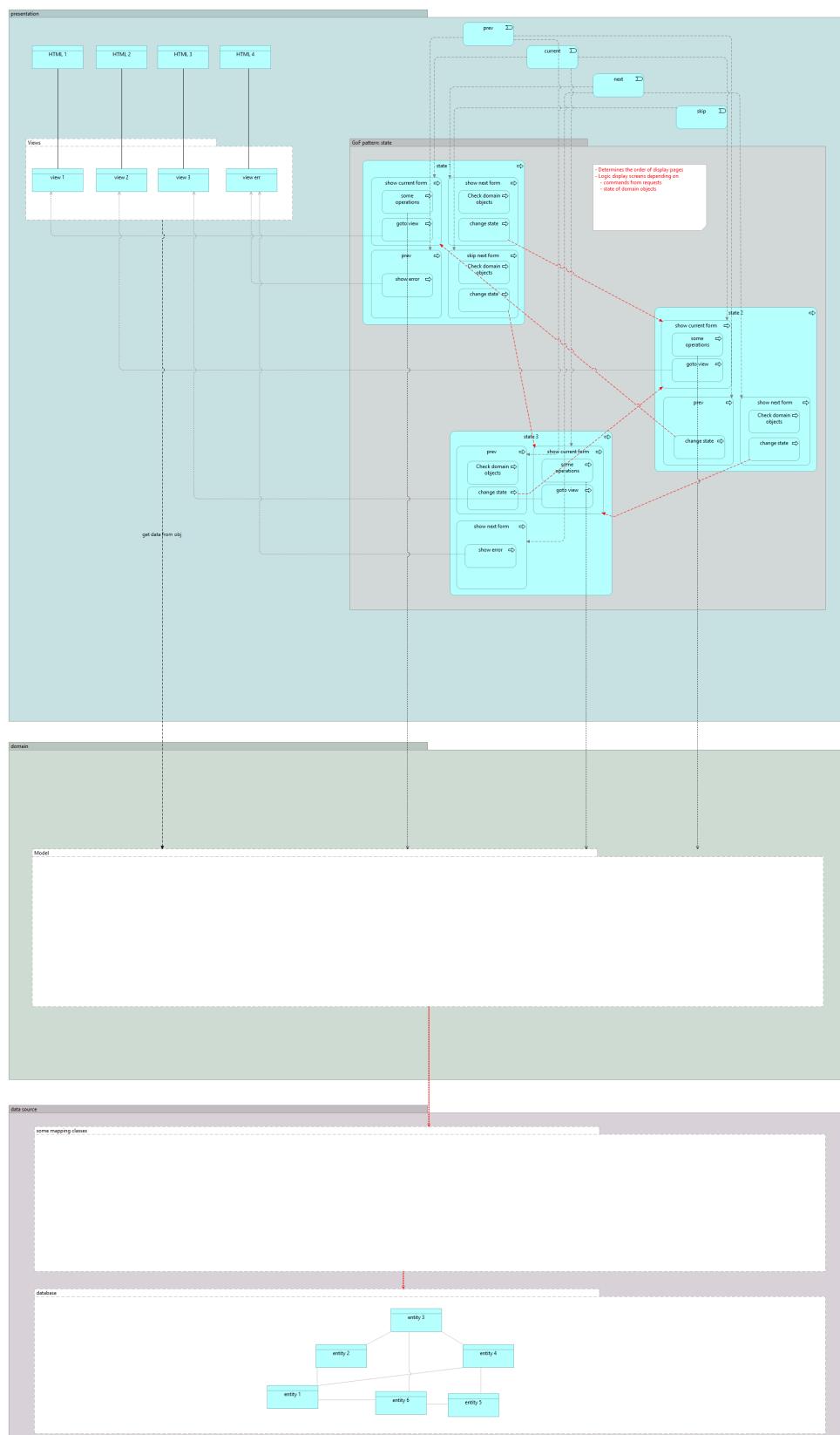


WEB REPRESENTATION CONTROLLER

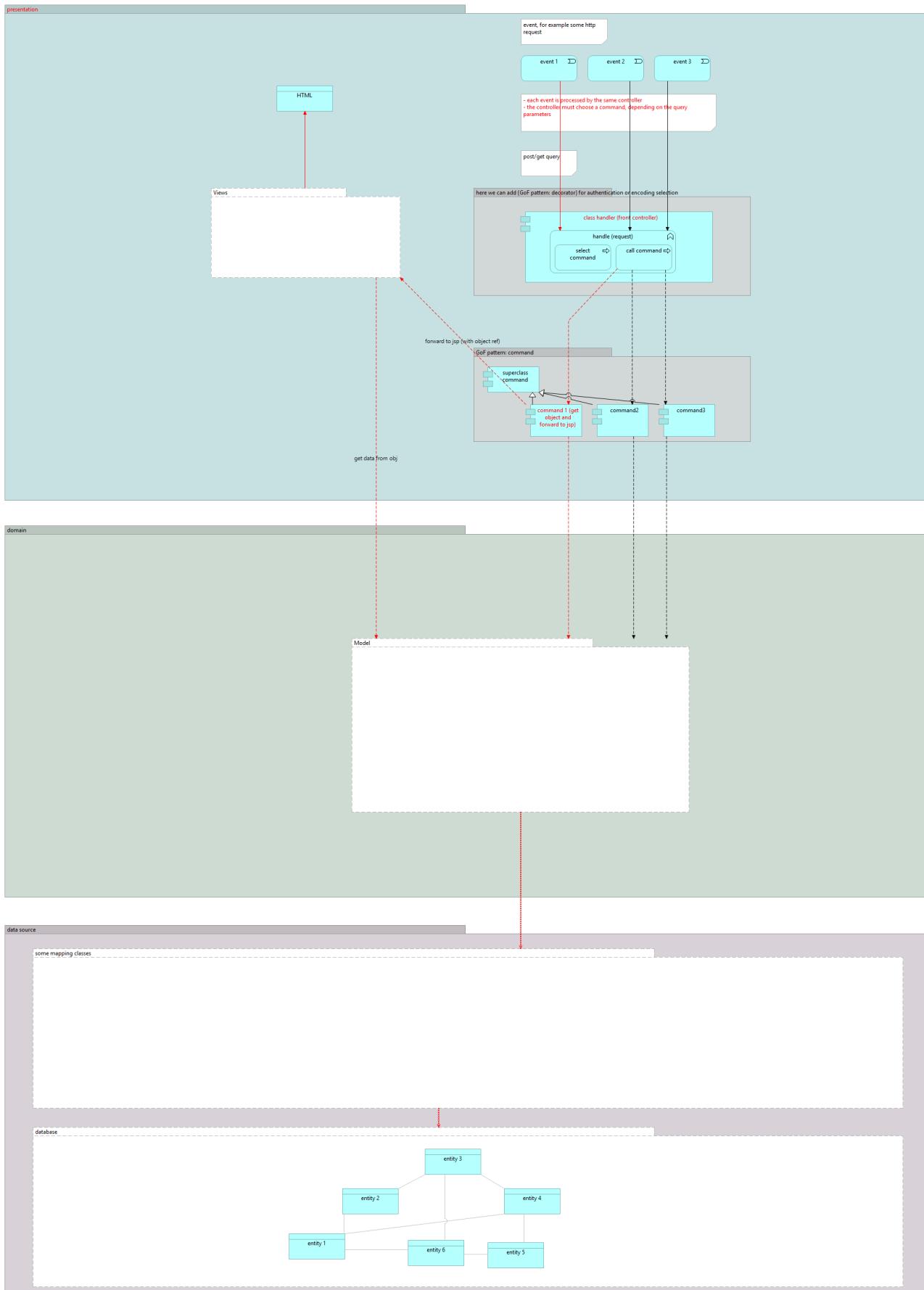
MODEL VIEW CONTROLLER



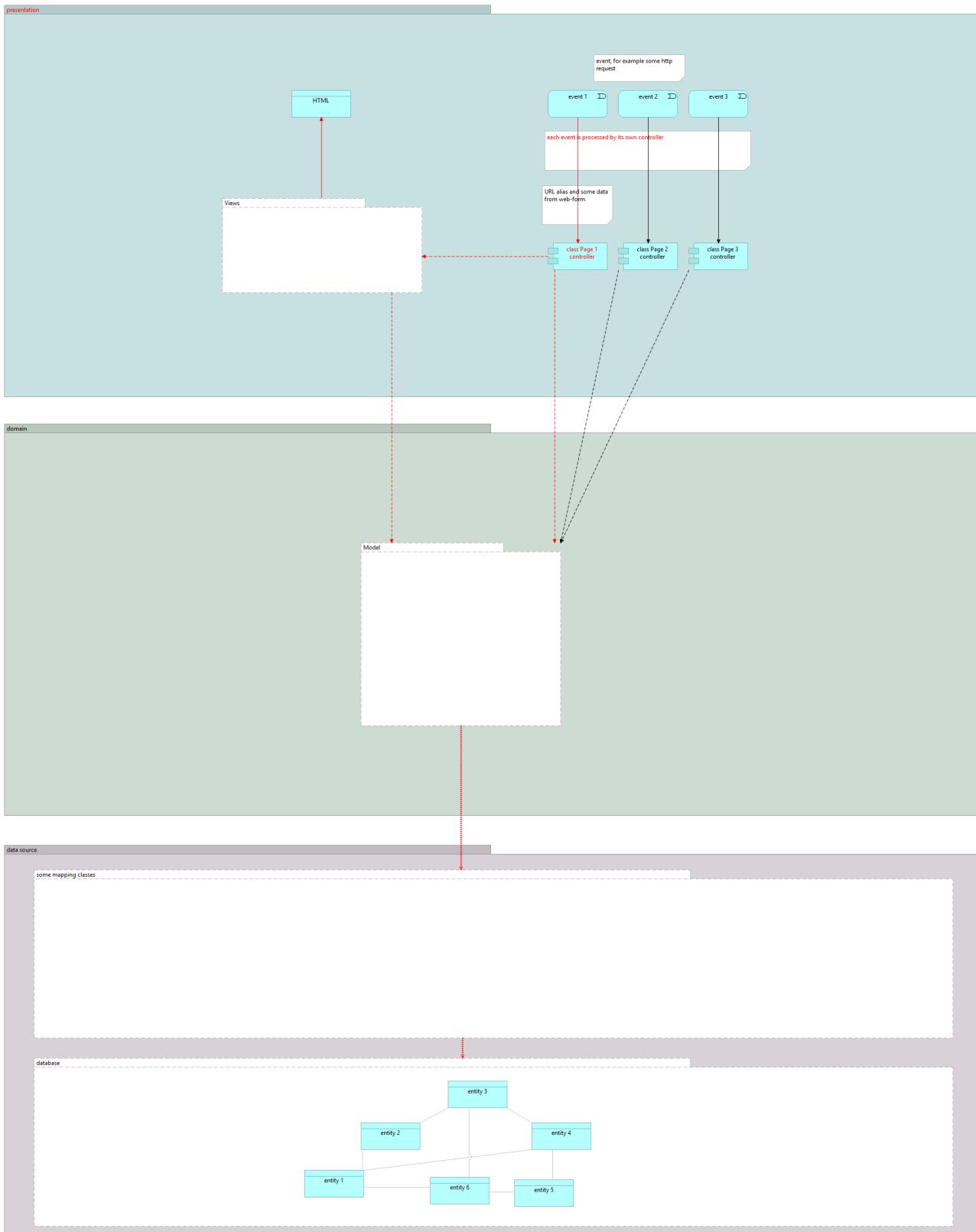
APPLICATION CONTROLLER



FRONT CONTROLLER

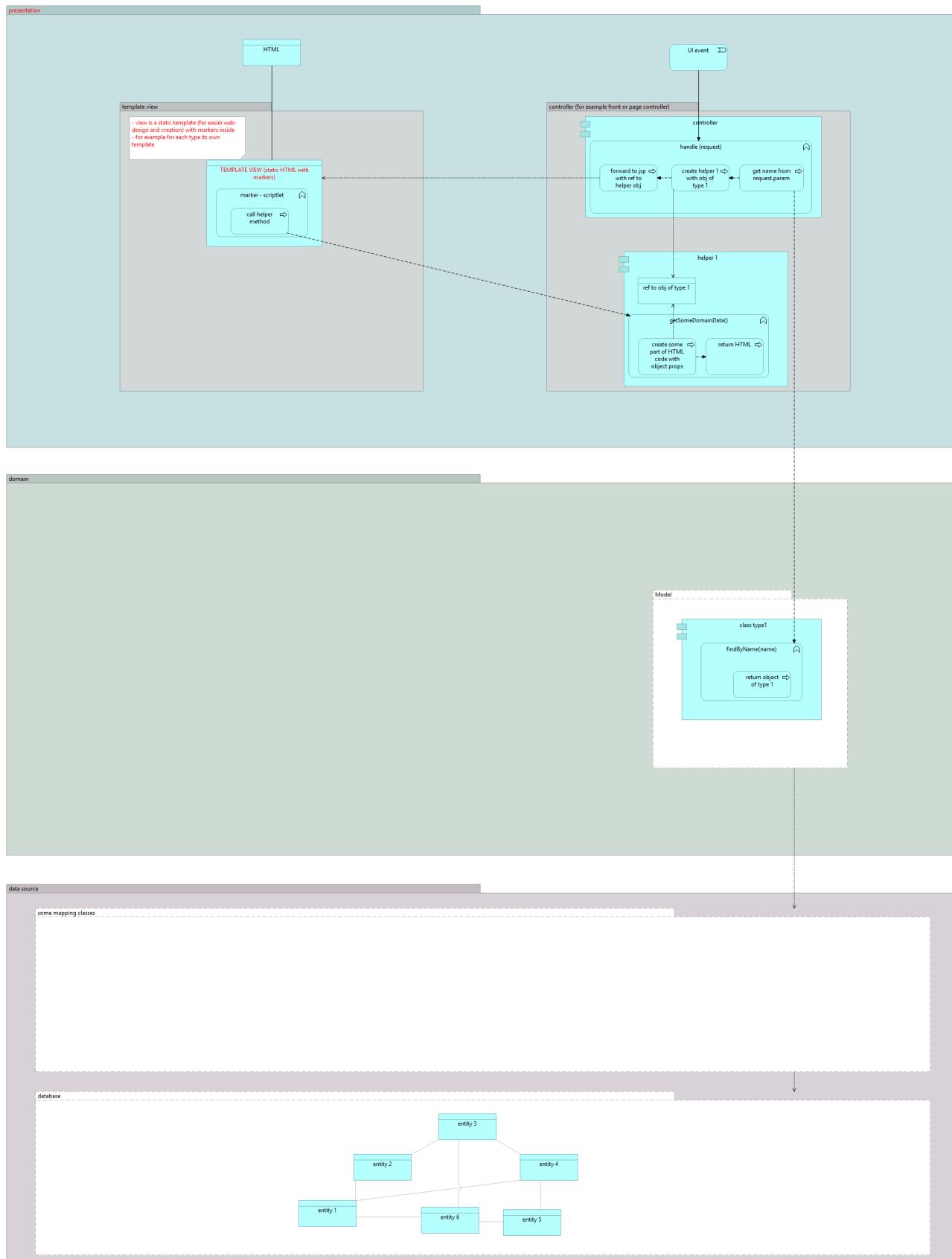


PAGE CONTROLLER

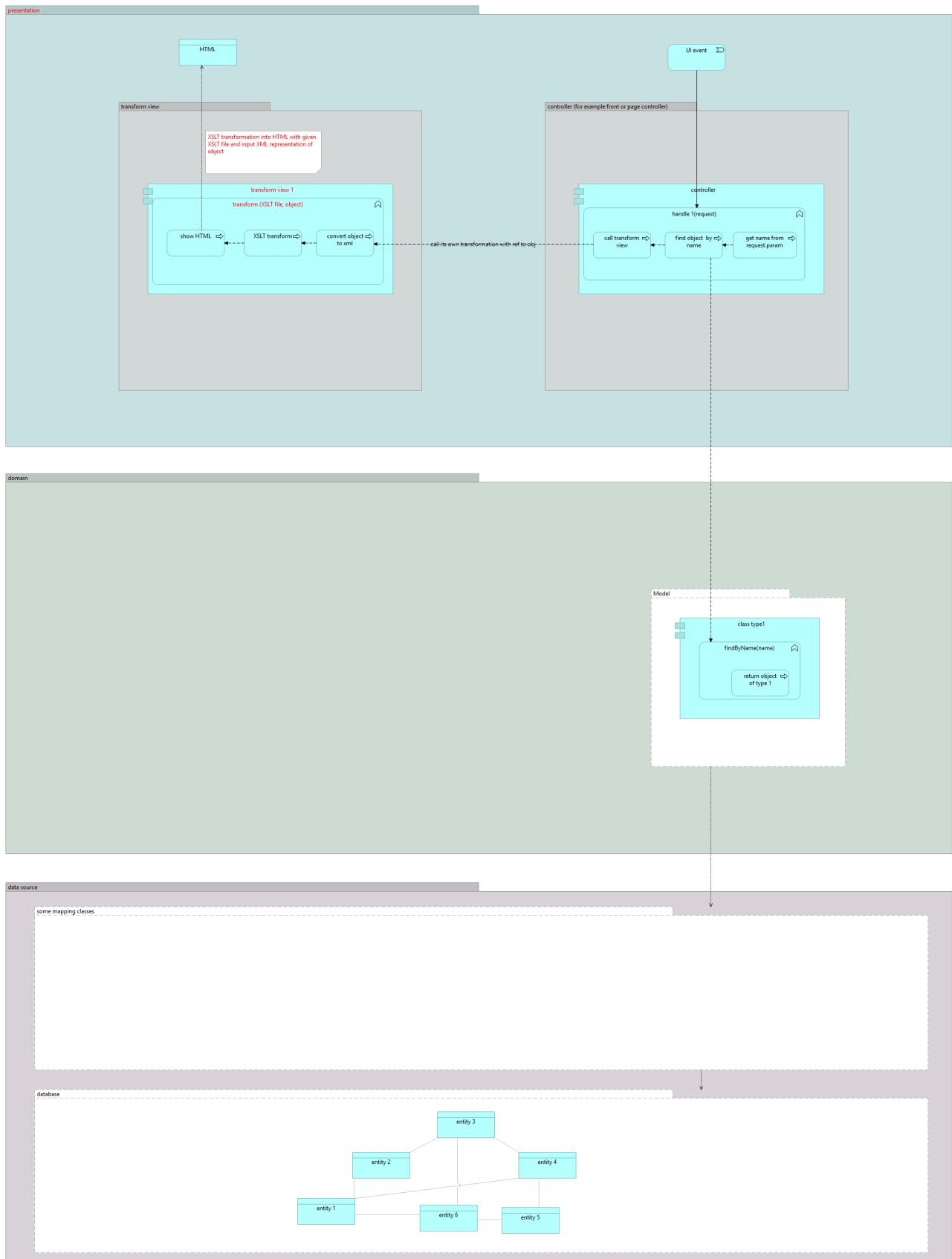


WEB REPRESENTATION VIEW

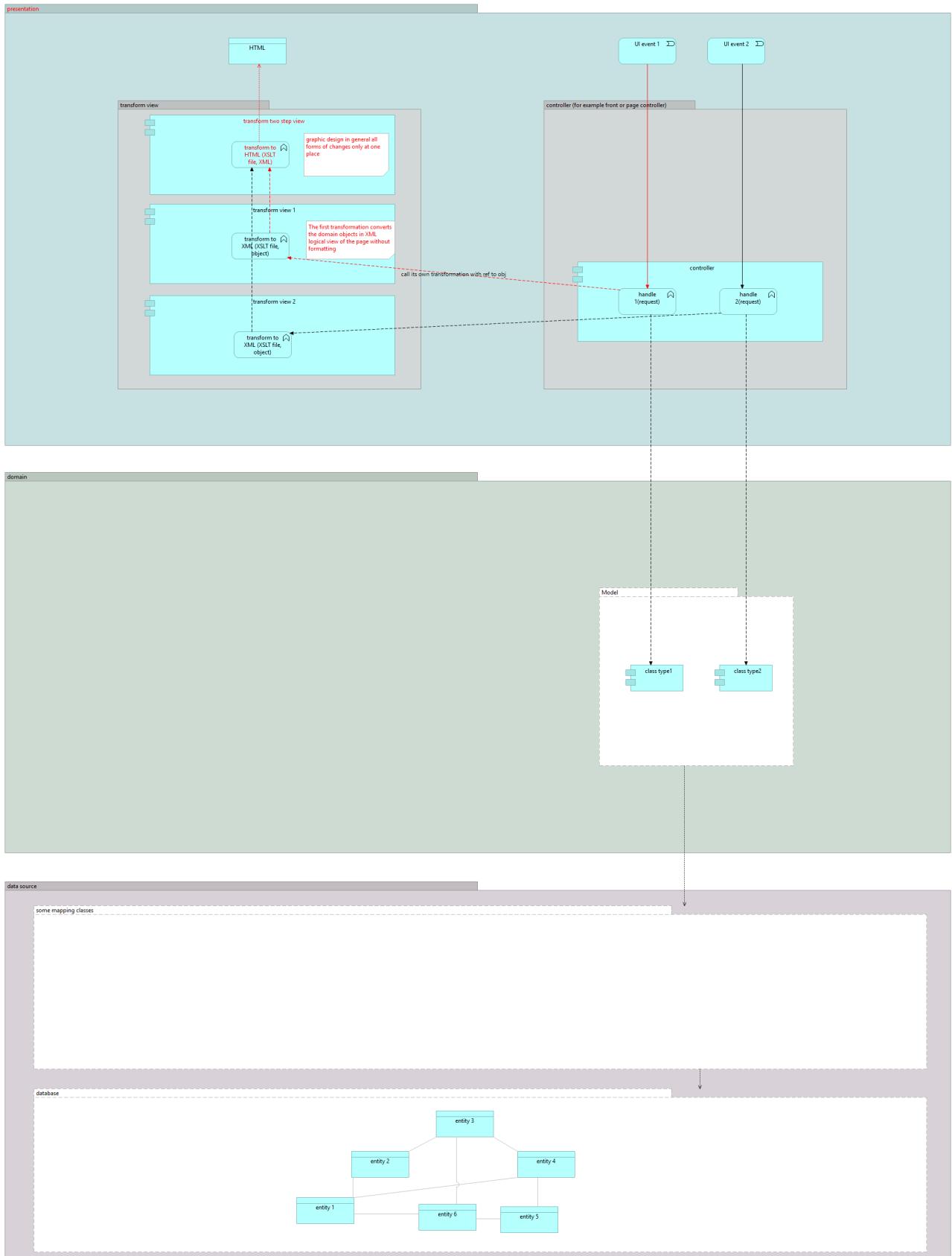
TEMPLATE VIEW



TRANSFORM VIEW



TWO STEP VIEW



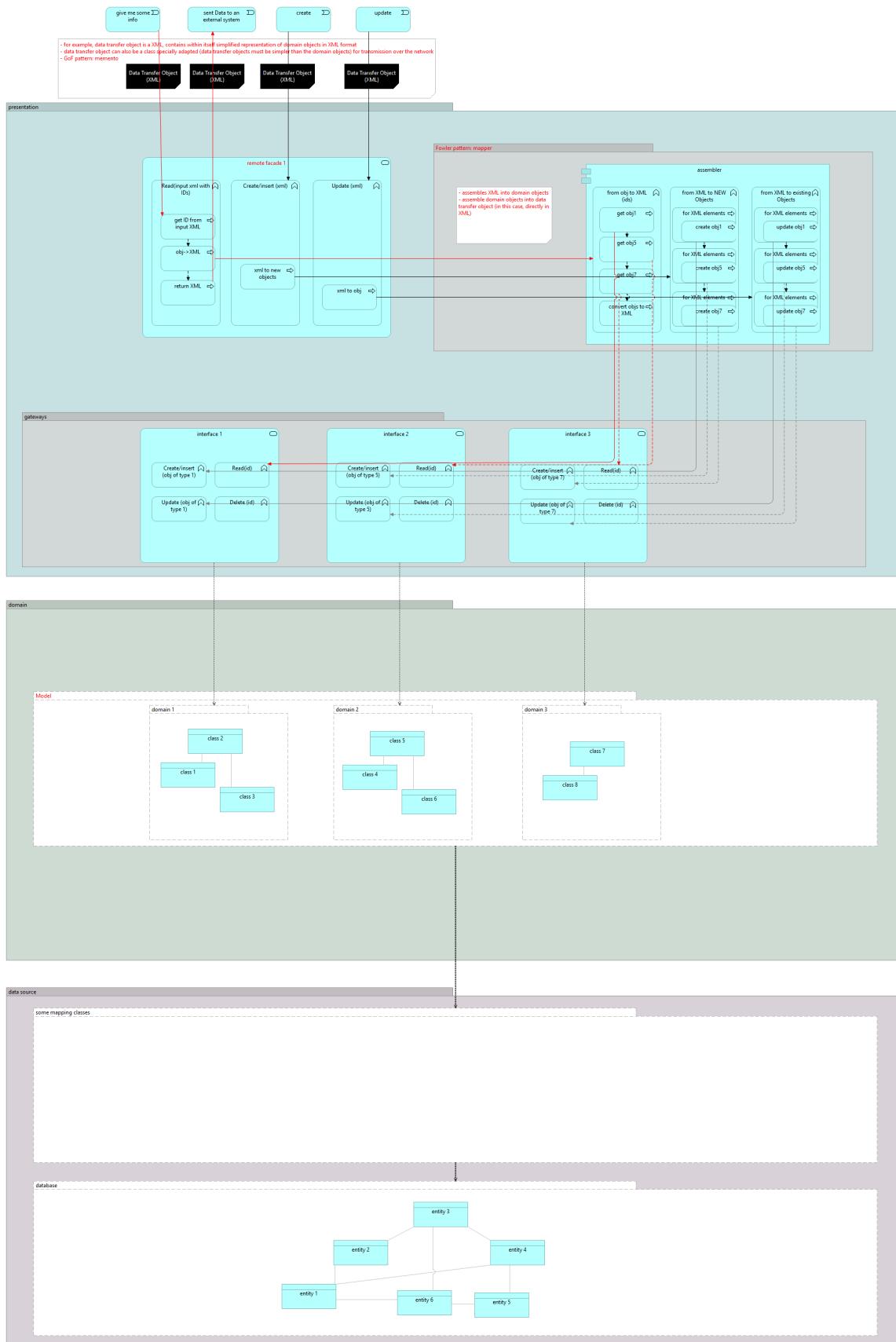
DISTRIBUTED PROCESSING

ENTERPRISE PATTERNS

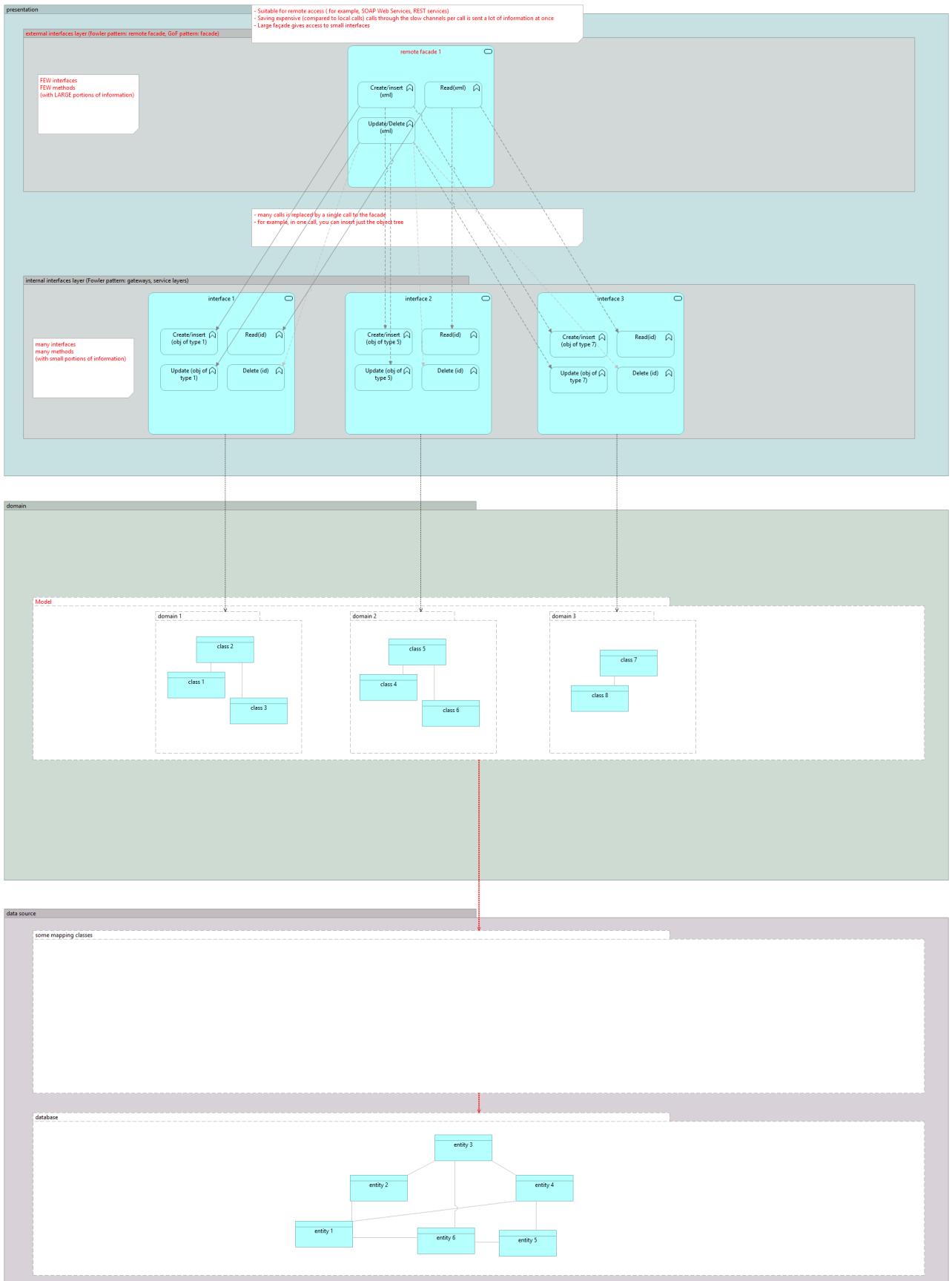
Martin Fowler



DATA TRANSFER OBJECT

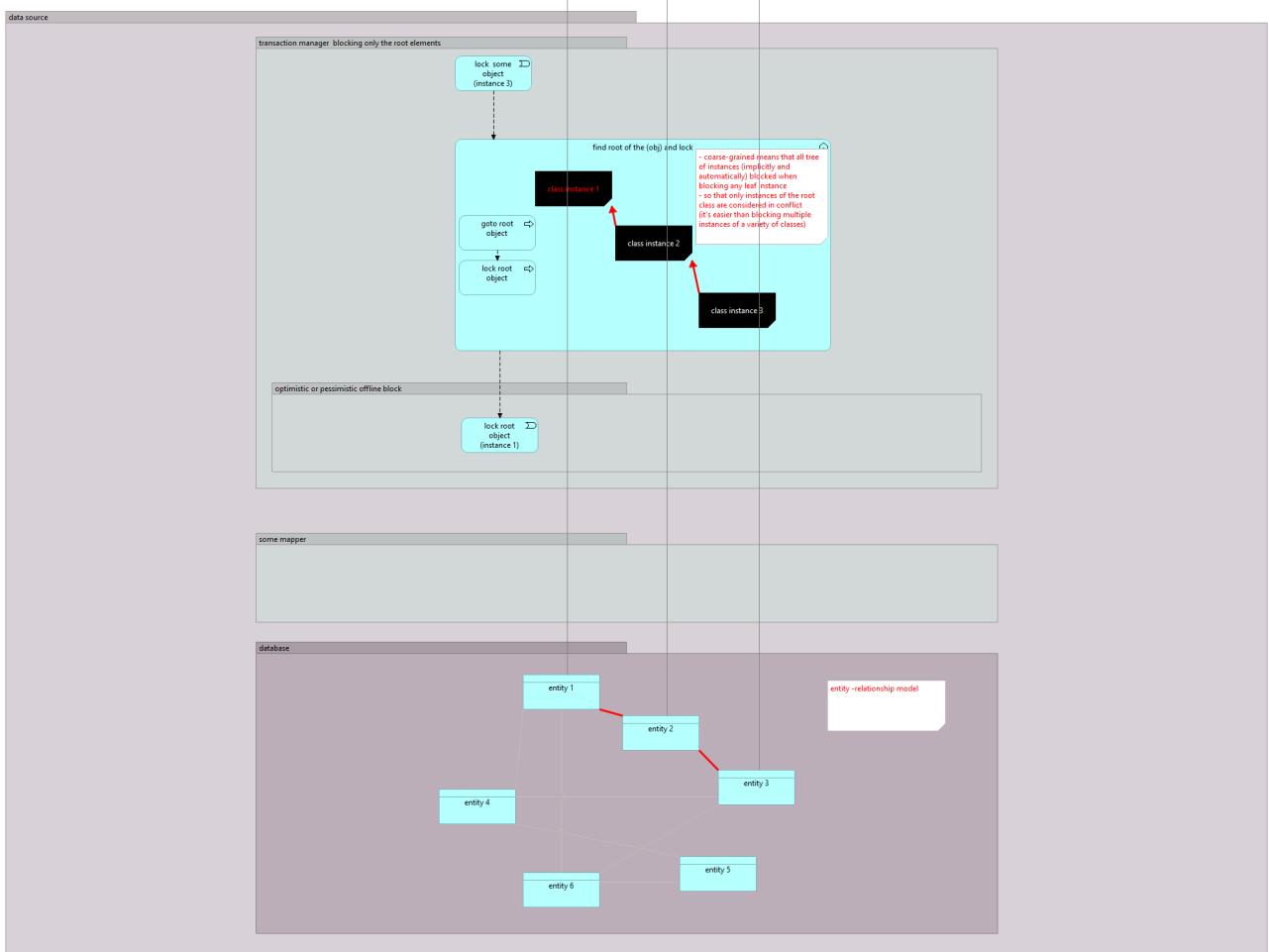
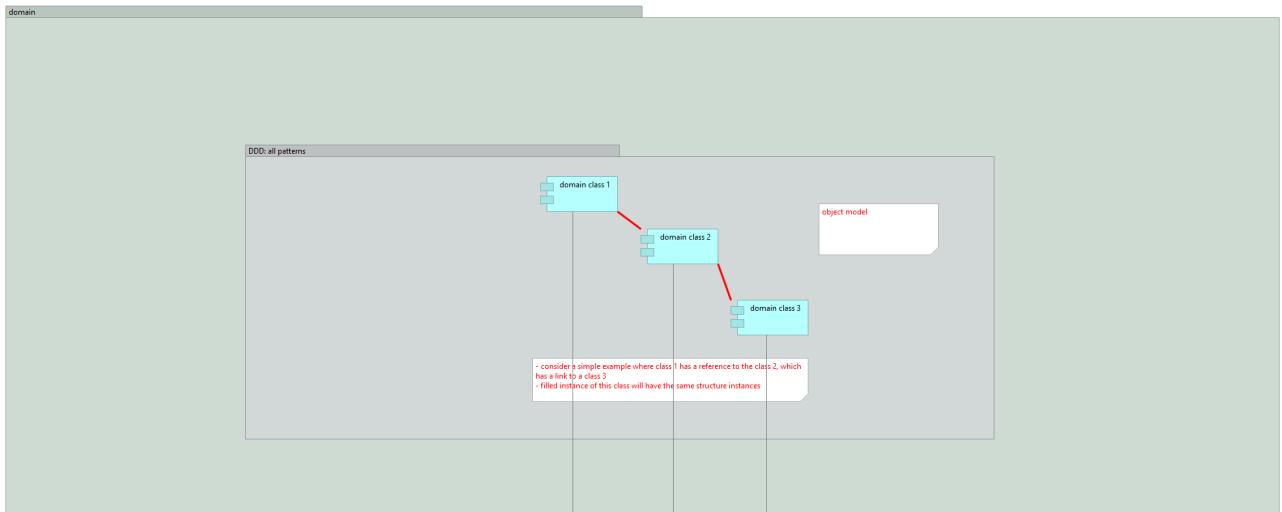


REMOTE FAÇADE

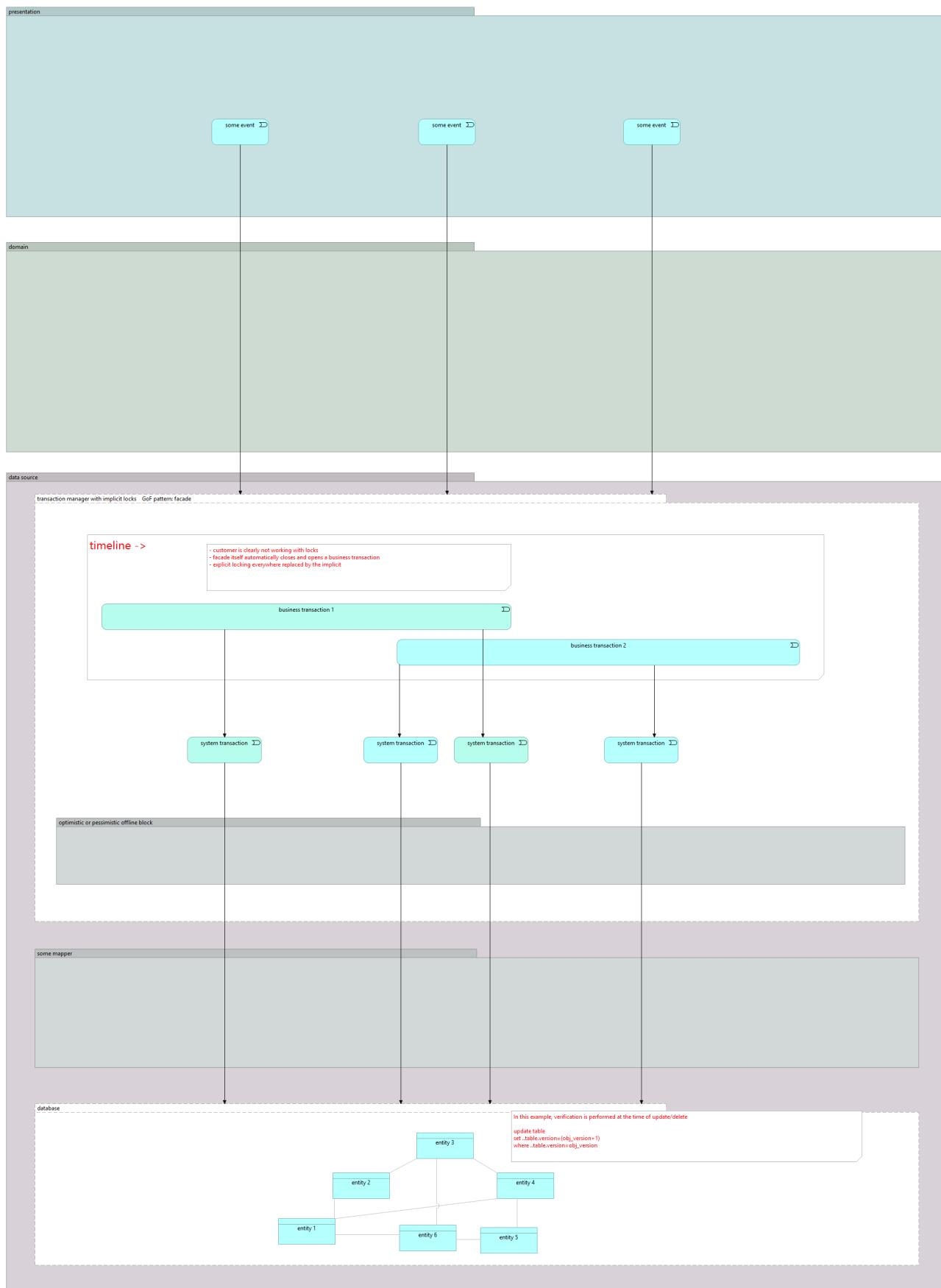


PARALLEL PROCESSING

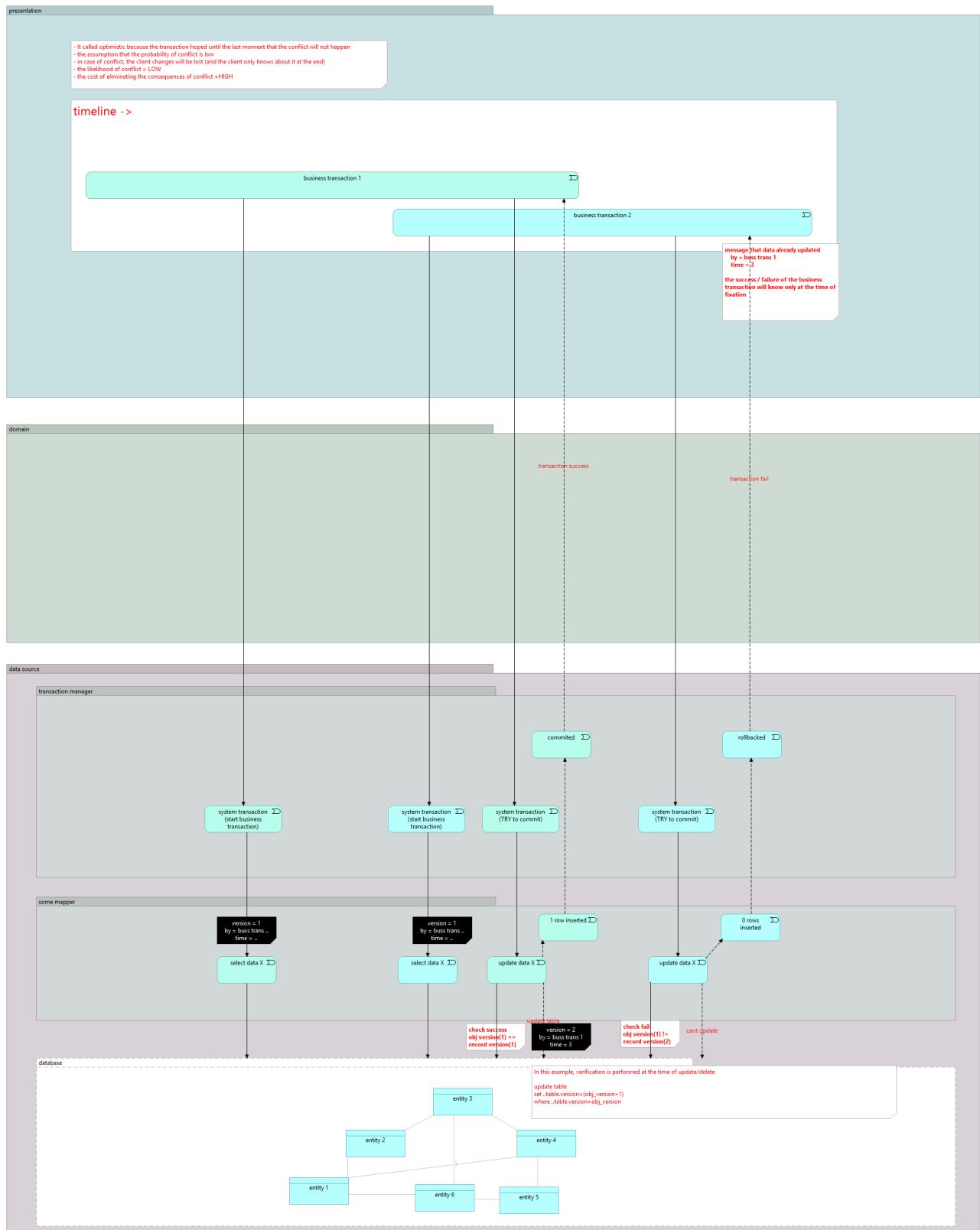
COARSE-GRAINED LOCK



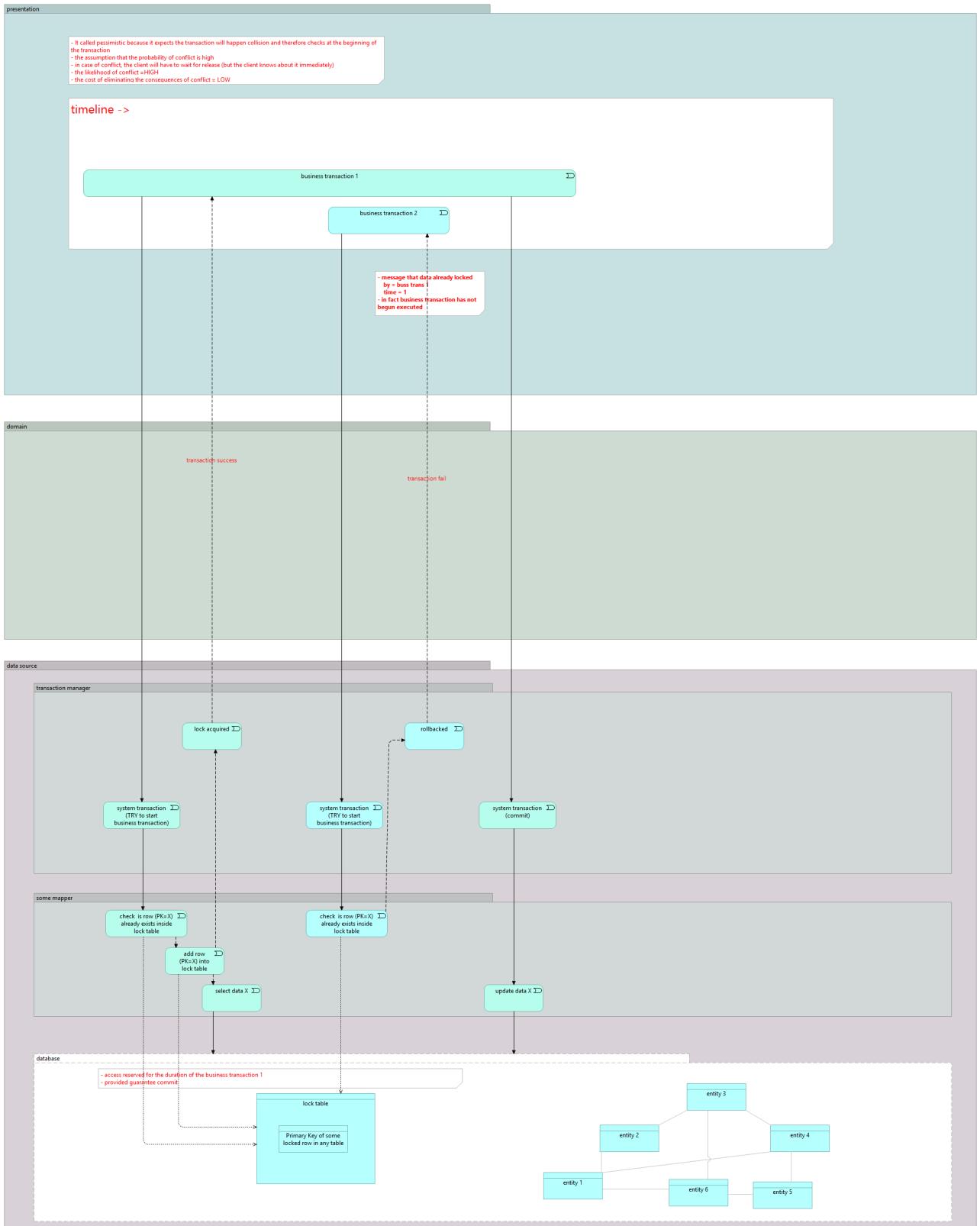
IMPLICIT LOCK



OPTIMISTIC OFFLINE LOCK

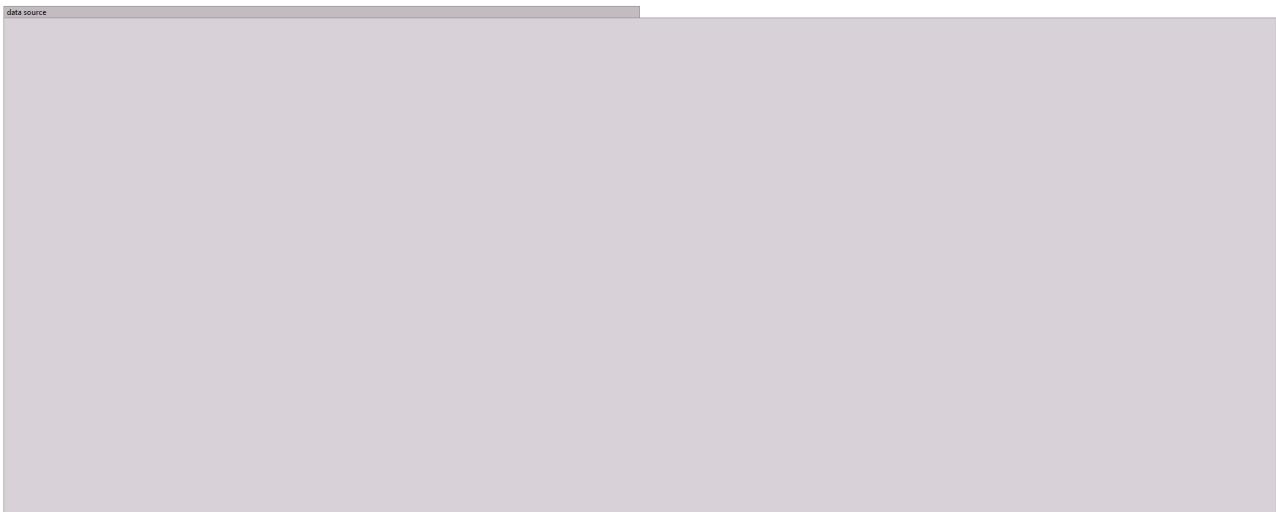
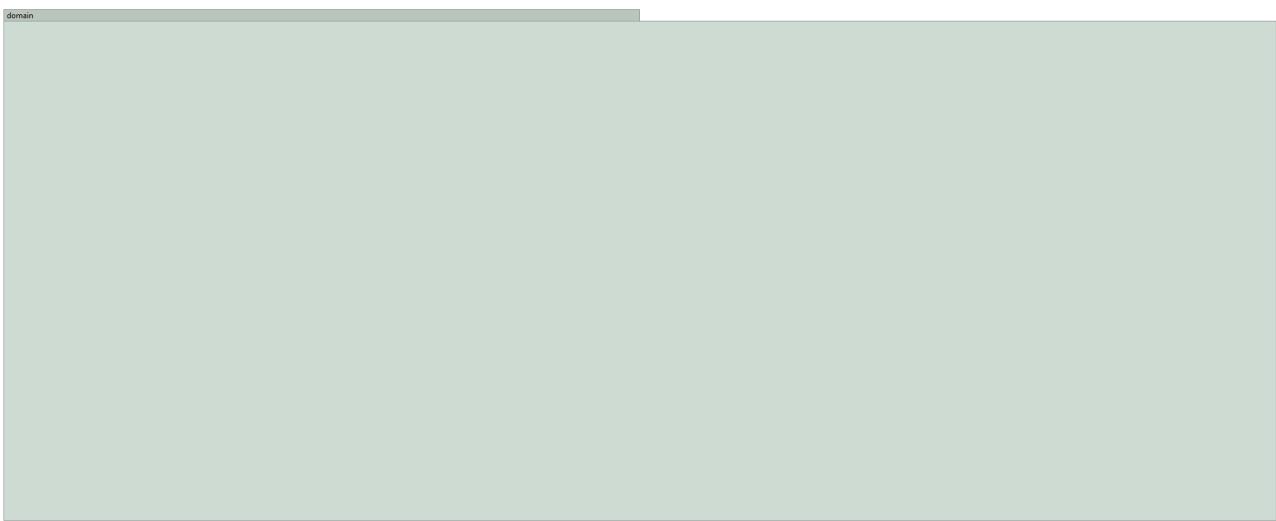
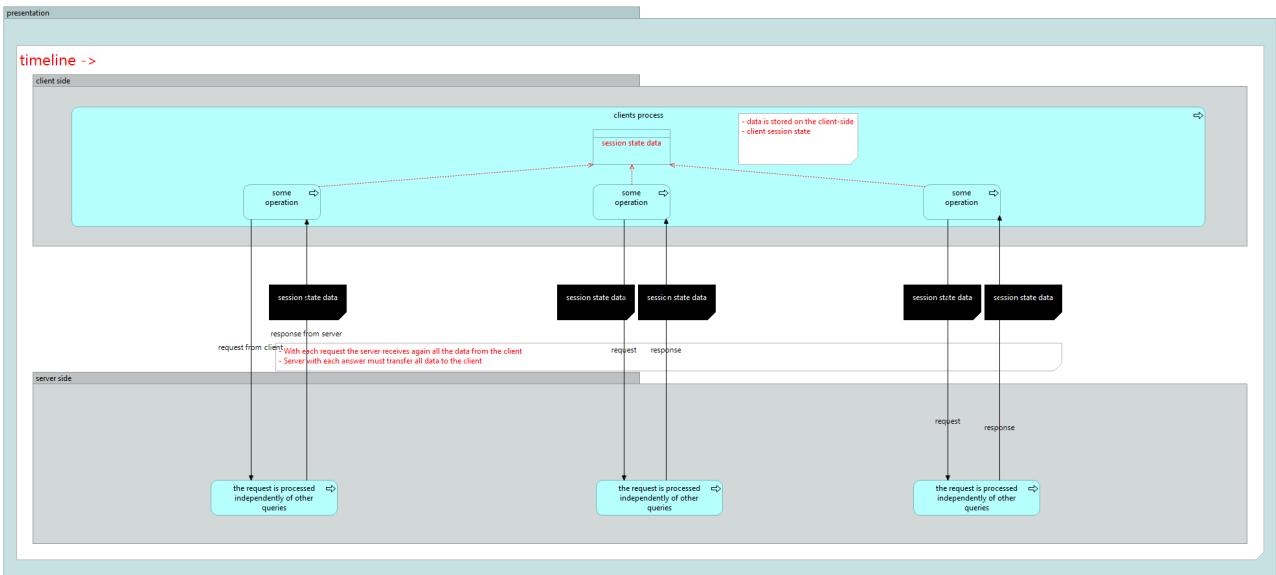


PESSIMISTIC OFFLINE LOCK

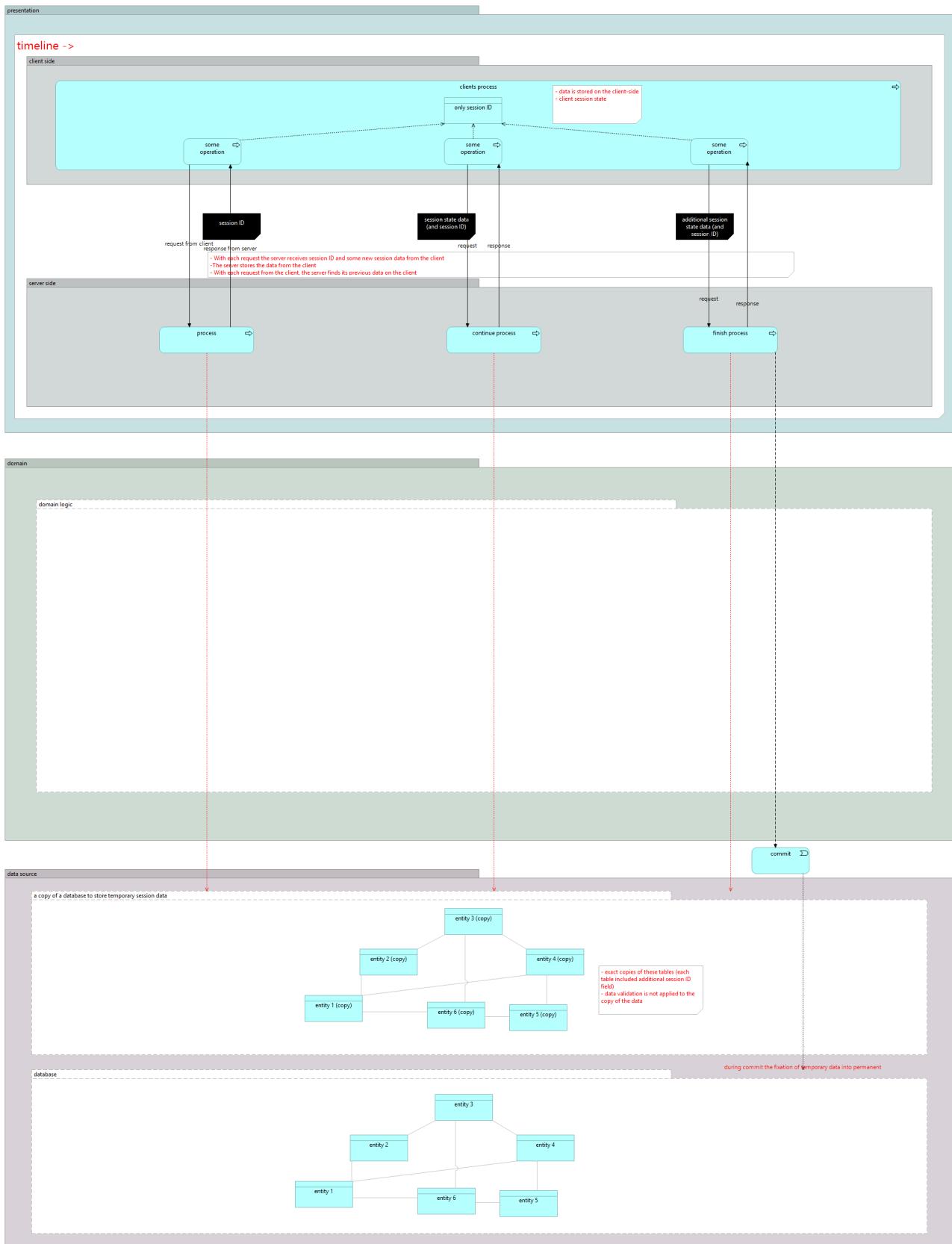


SESSION STATE

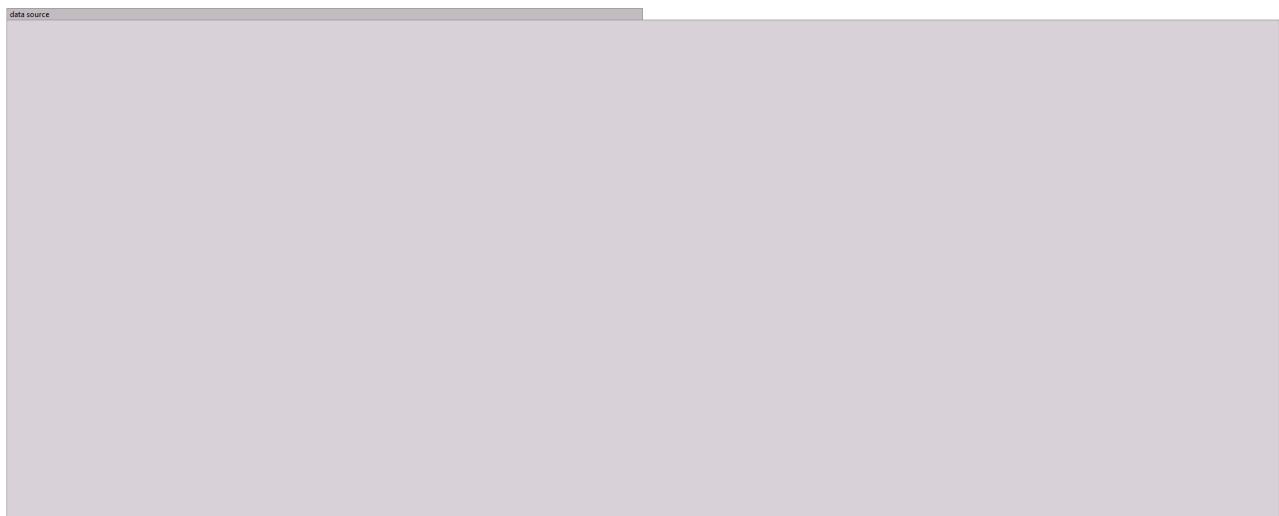
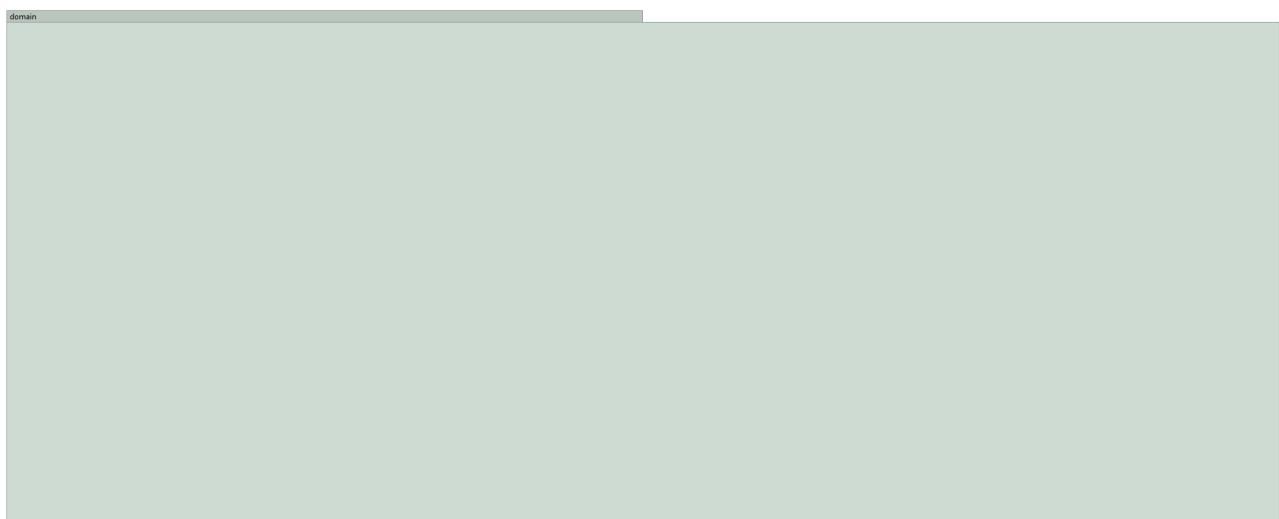
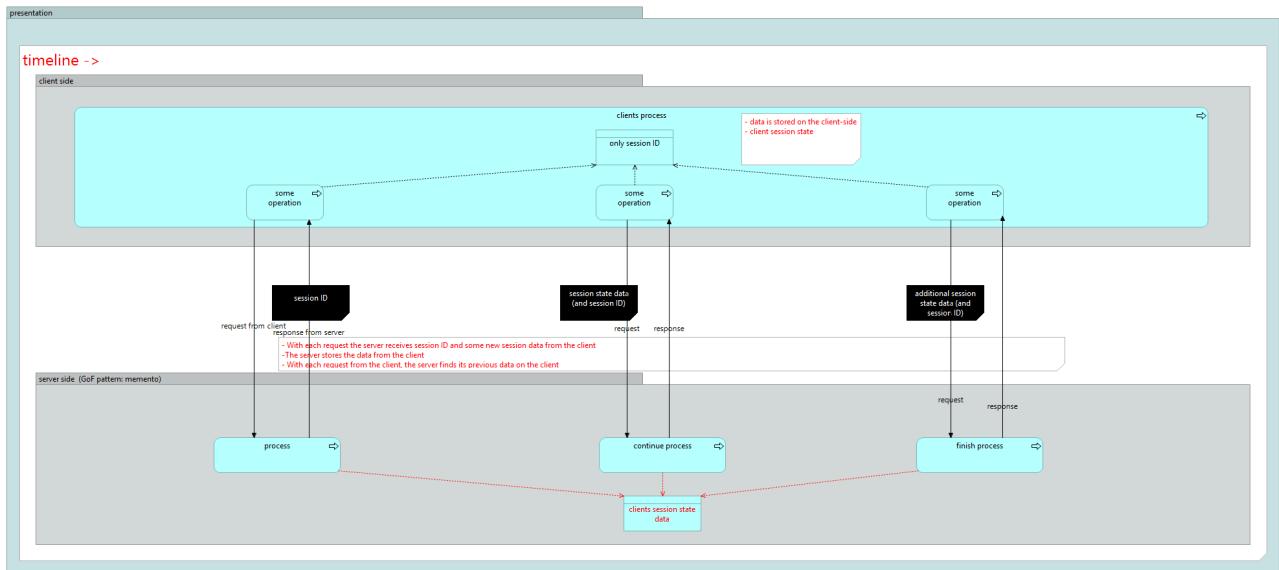
CLIENT SESSION STATE



DATABASE SESSION STATE



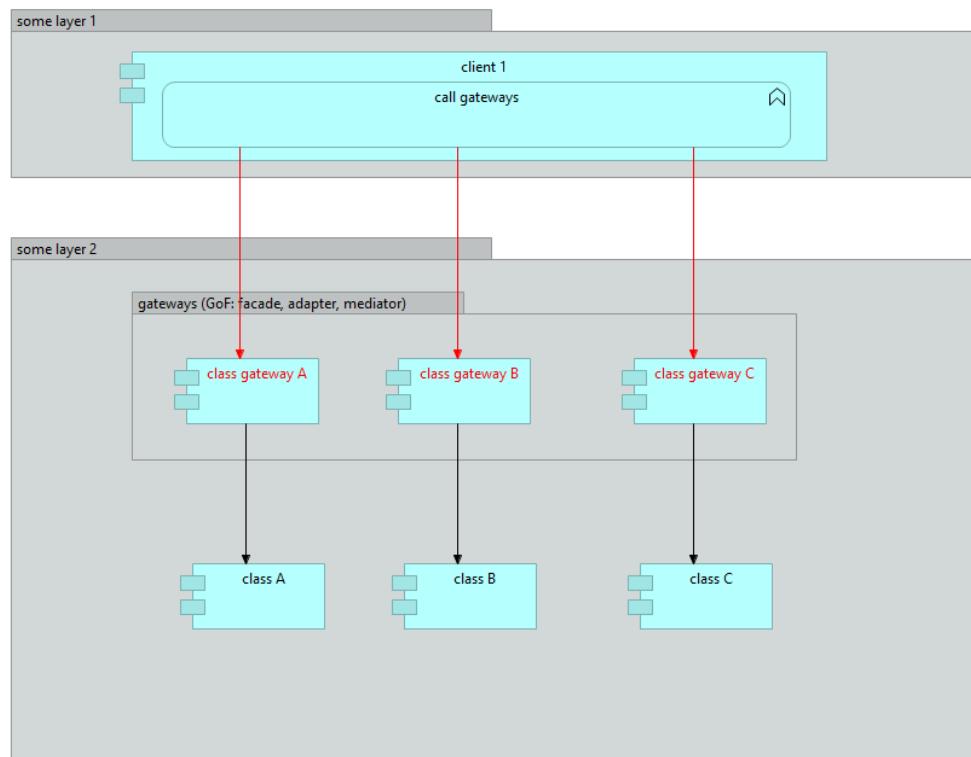
SERVER SESSION STATE



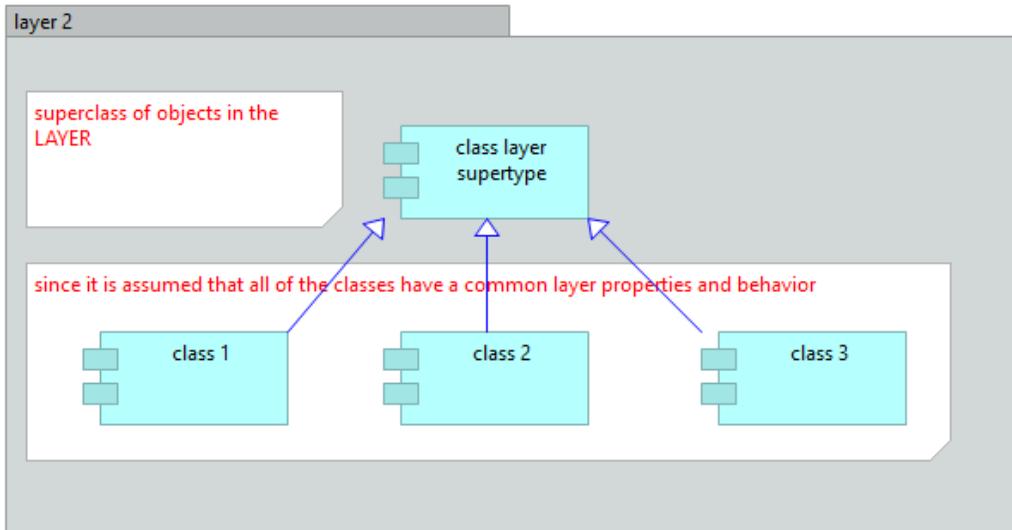
COMMON PATTERNS

GATEWAY

- a wrapper of classes that simplifies access it from the CLIENT 1
- not too simplifying to all possible clients (as in the facade pattern)
- And not adapting one class to another (as in the adapter) because both sides are developed at the same time



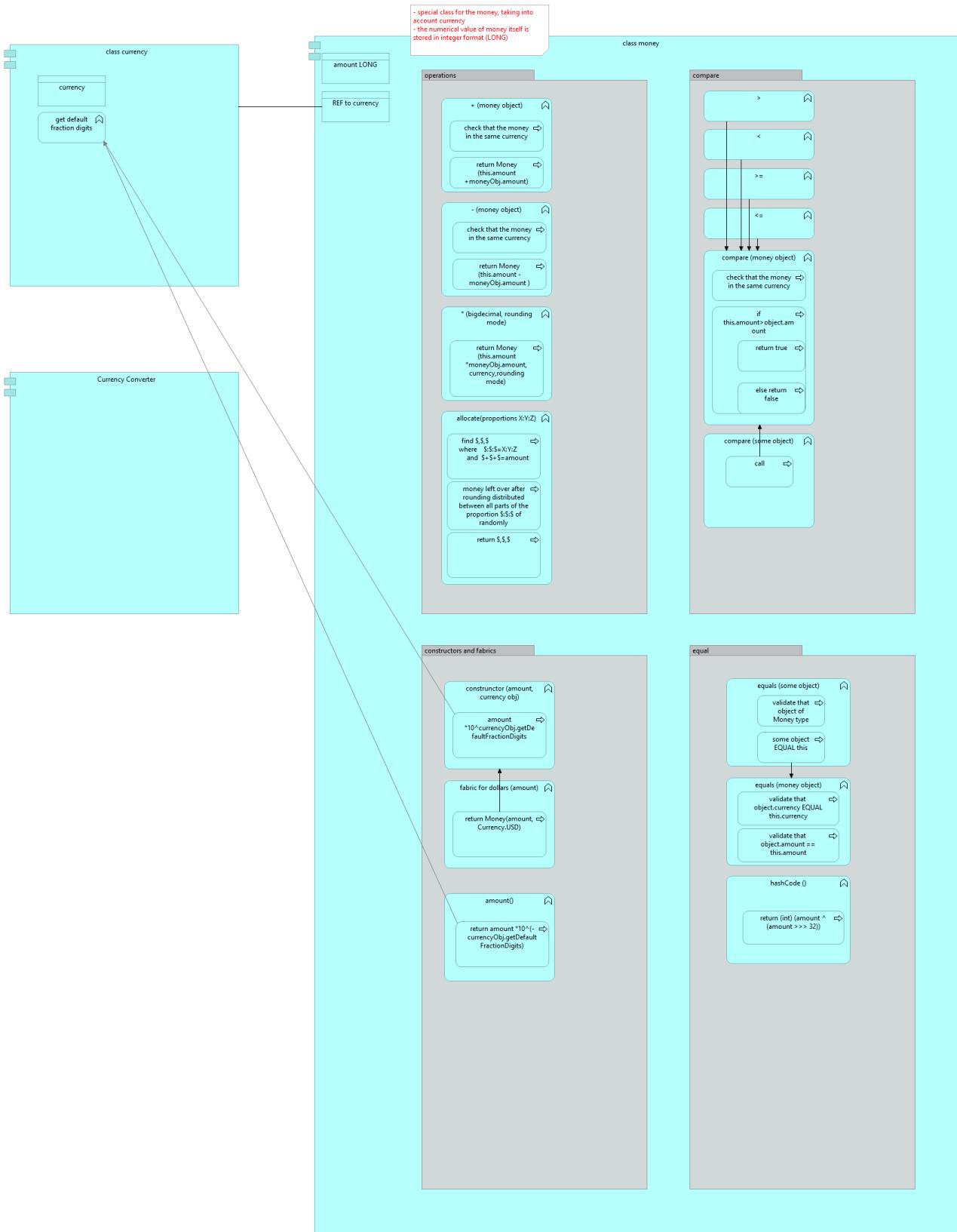
LAYER SUPERTYPE



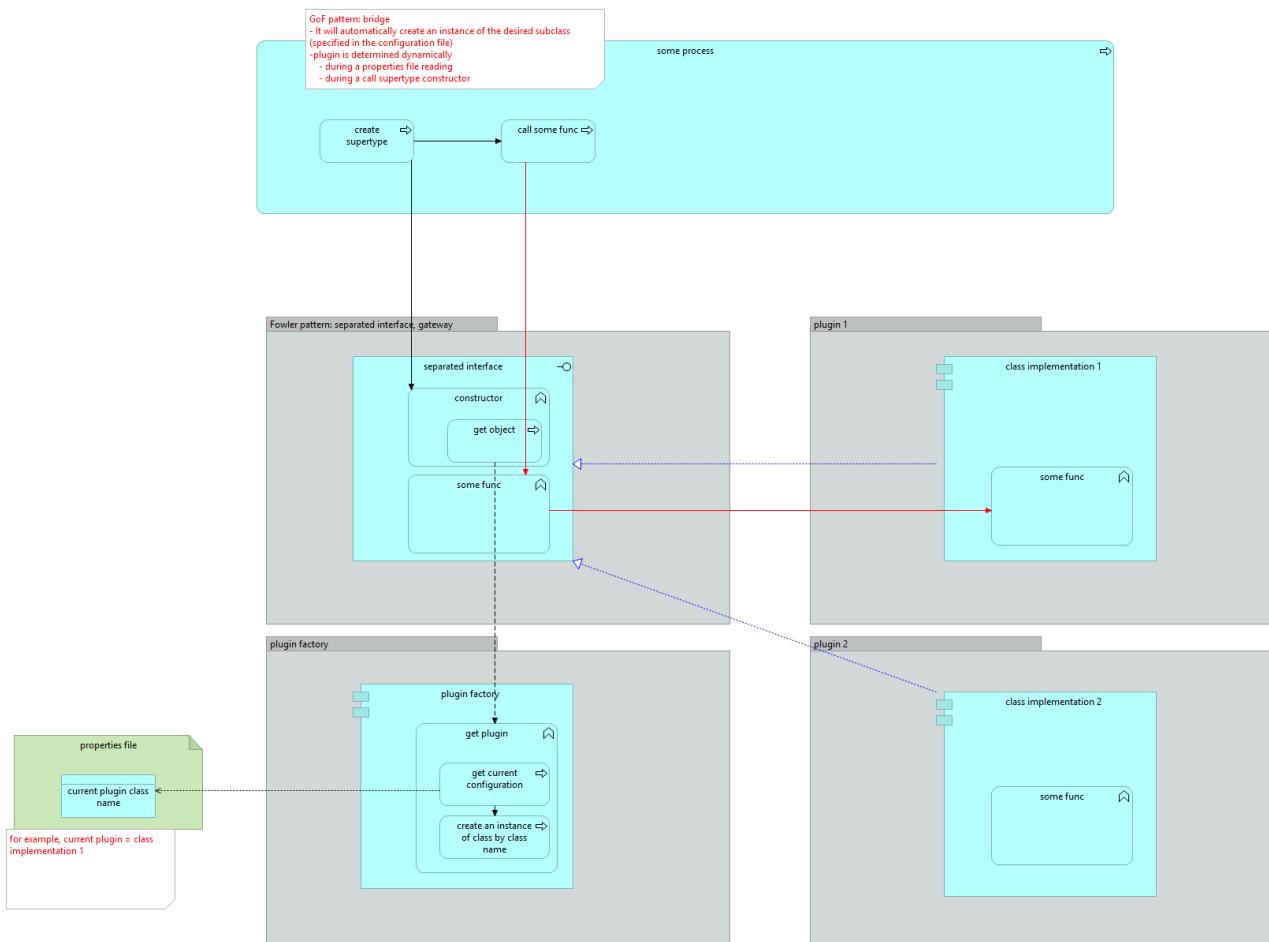
MAPPER



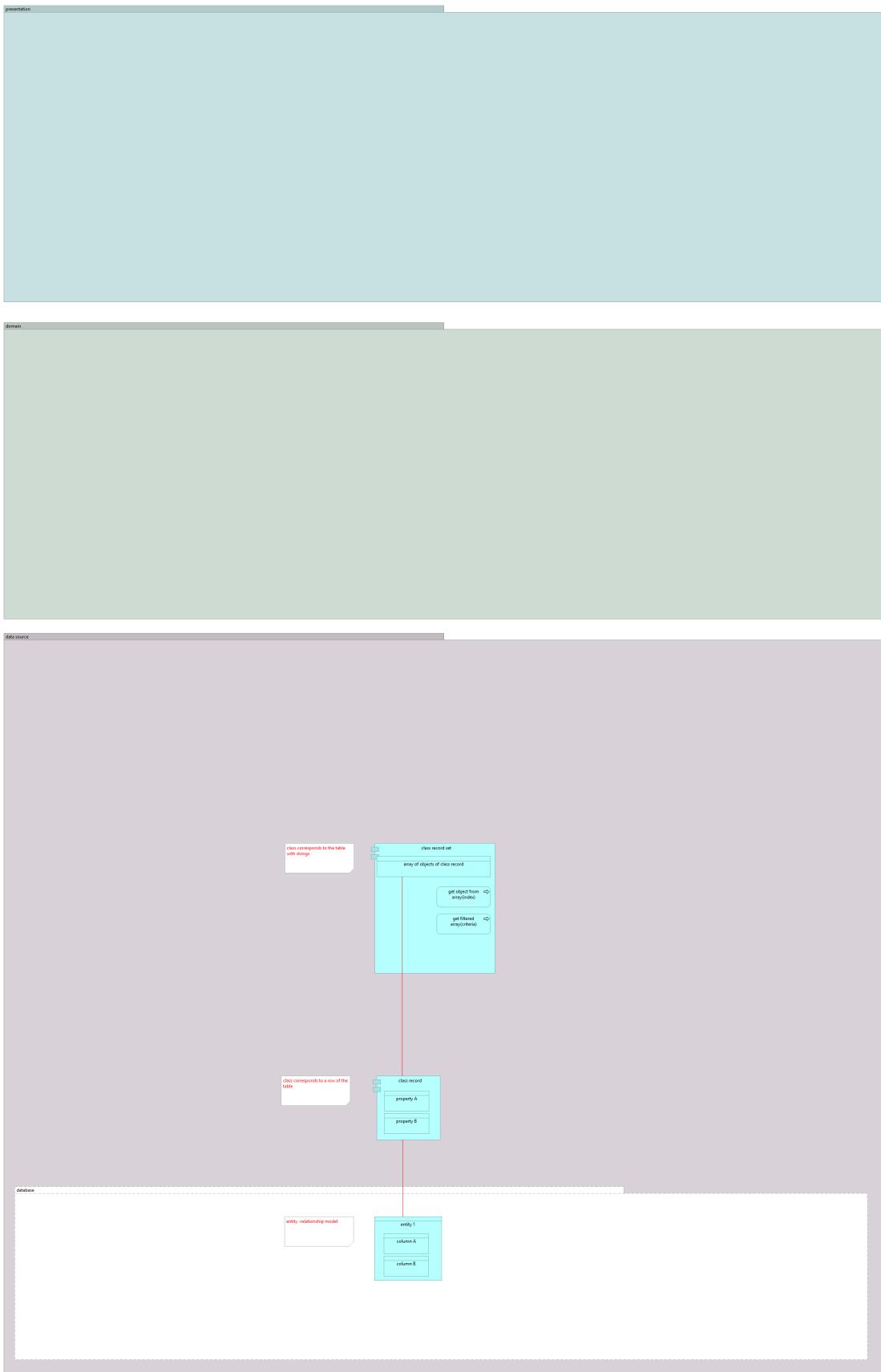
MONEY



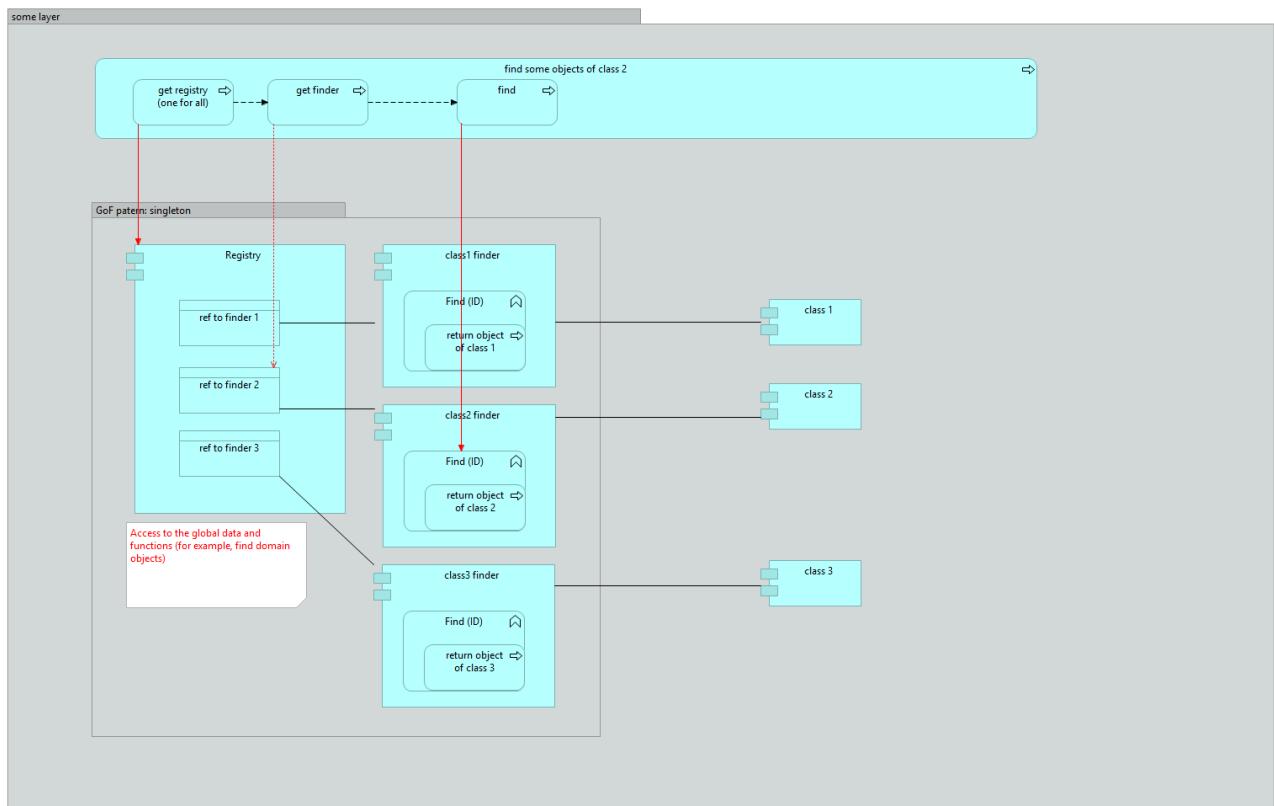
PLUGIN



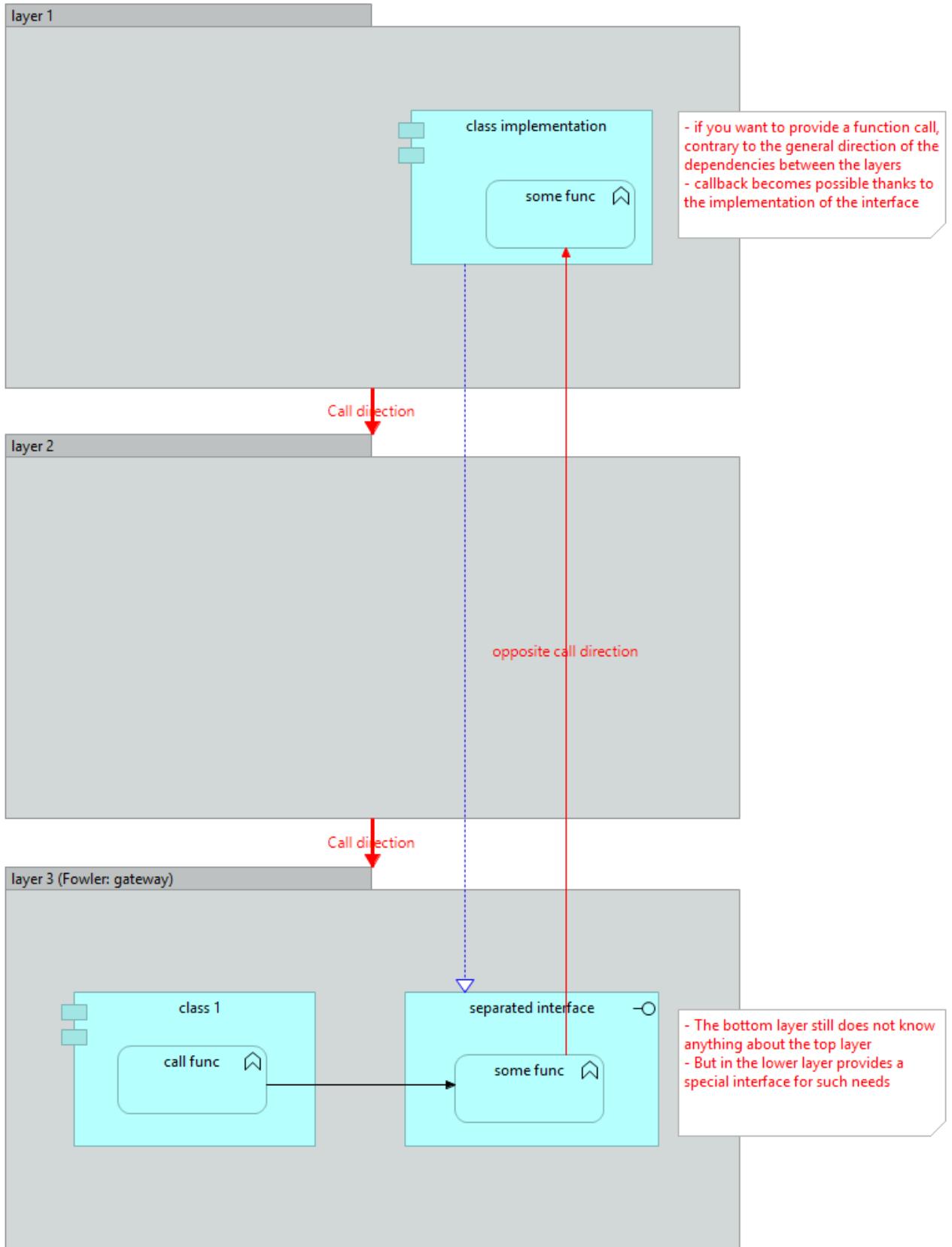
RECORD SET



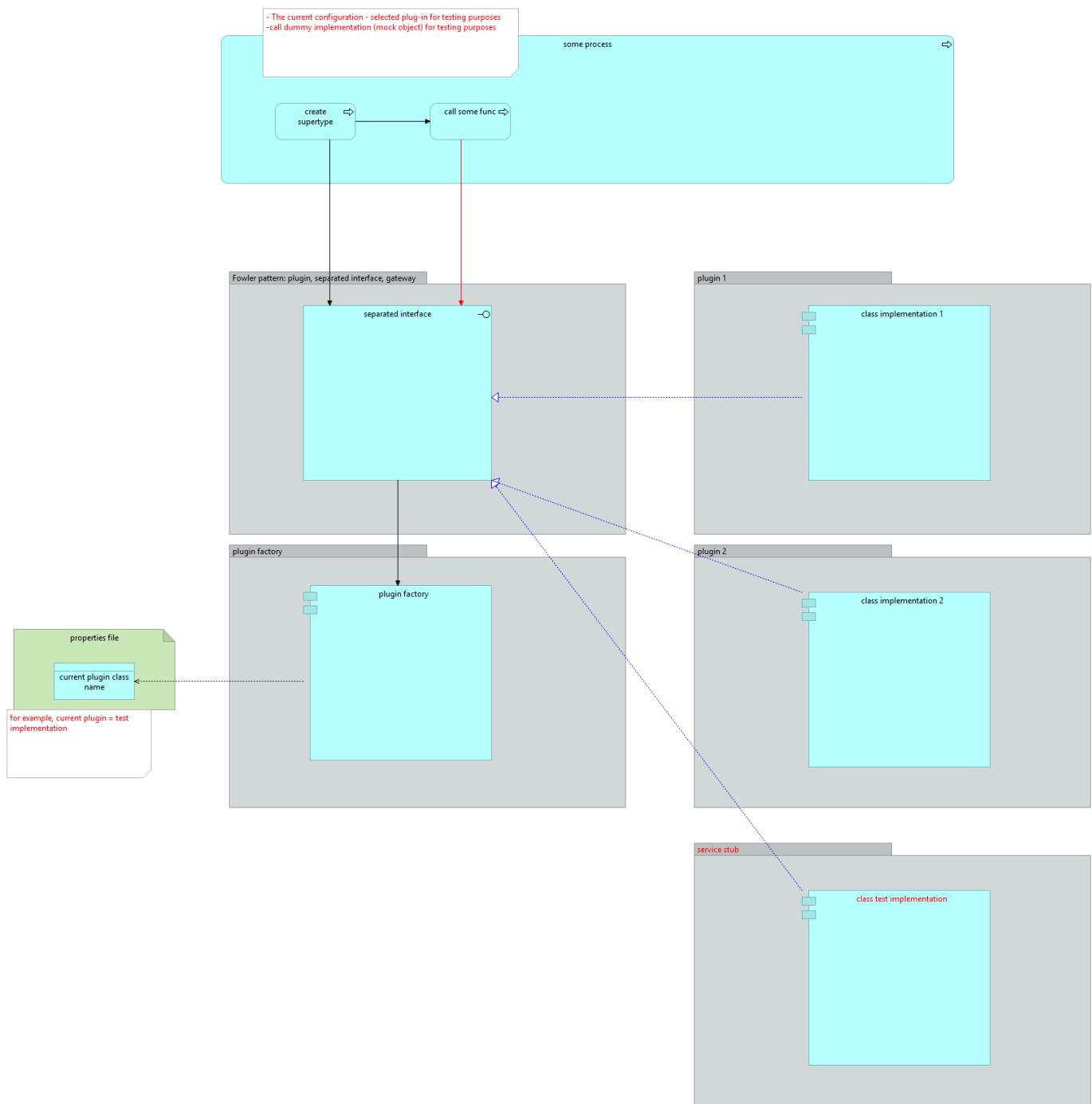
REGISTRY



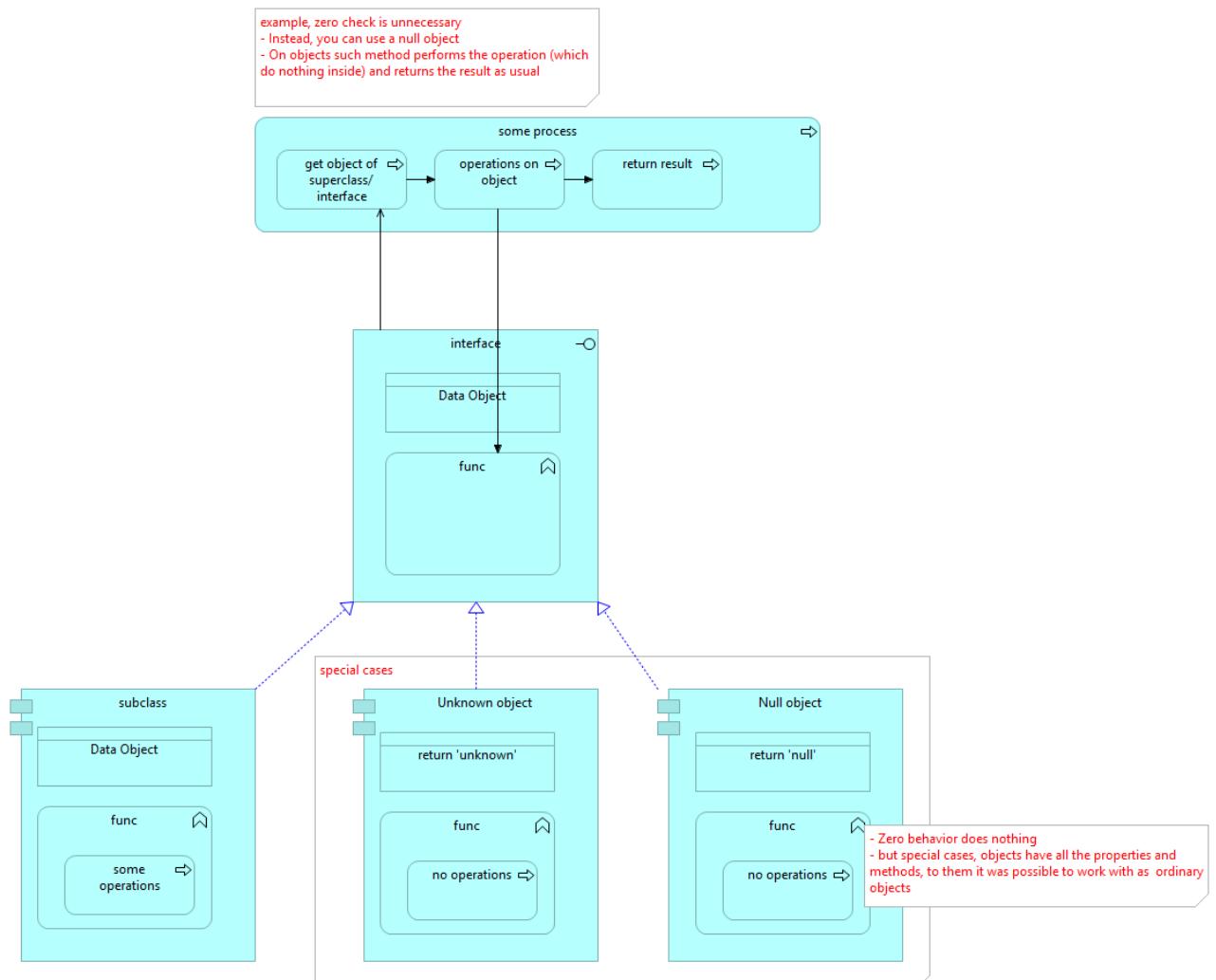
SEPARATED INTERFACE



SERVICE STUB

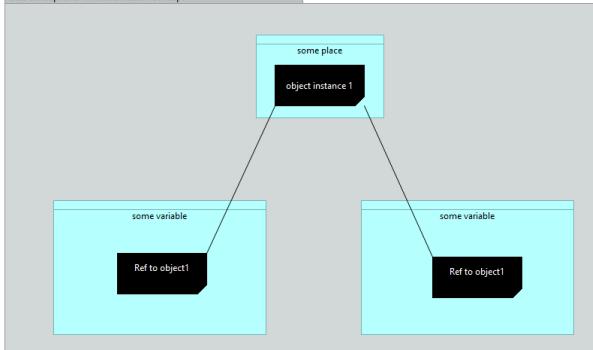


SPECIAL CASE

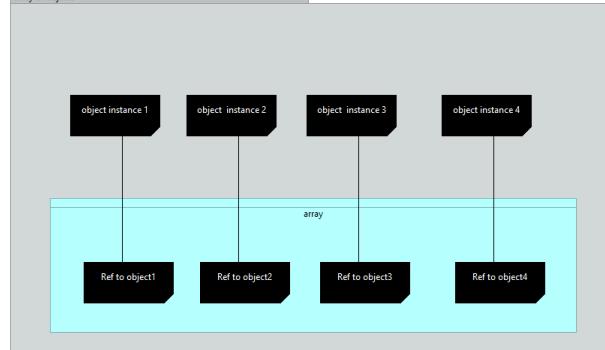


VALUE OBJECT

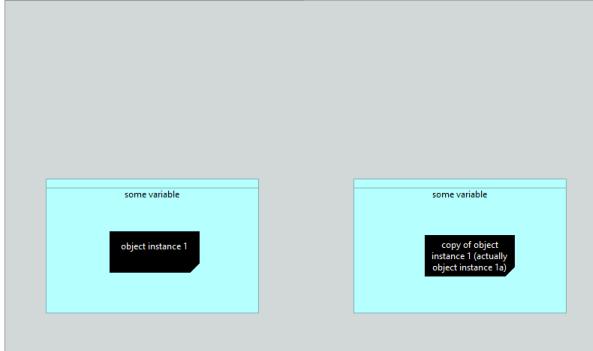
class concept and link transmission concept



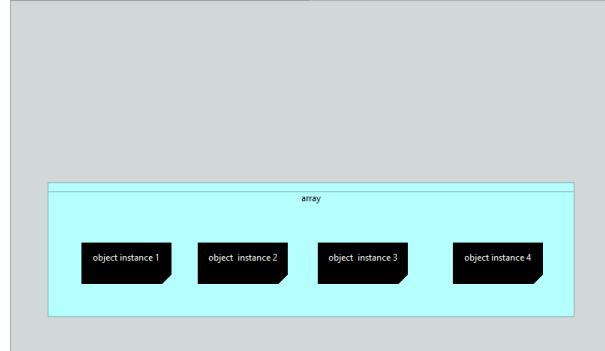
array of objects



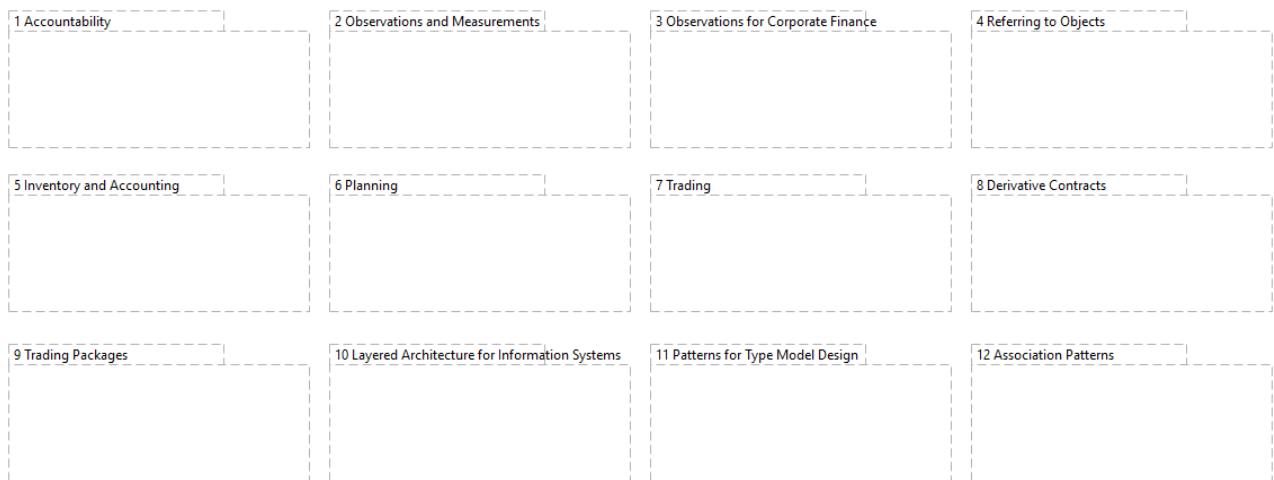
value object concept and by value transfer concept



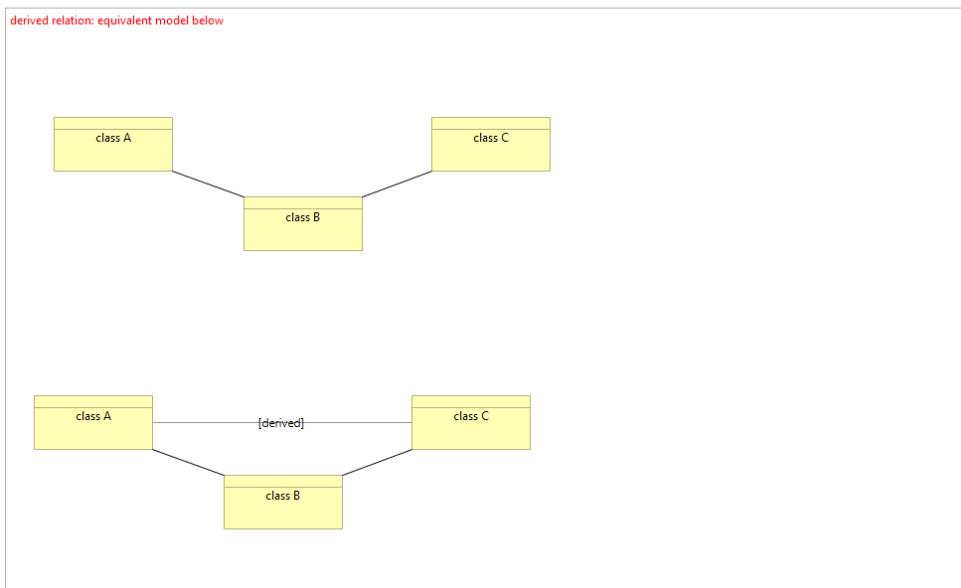
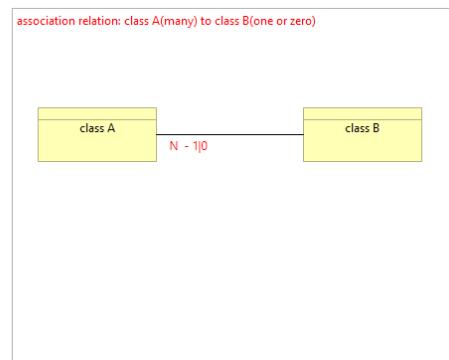
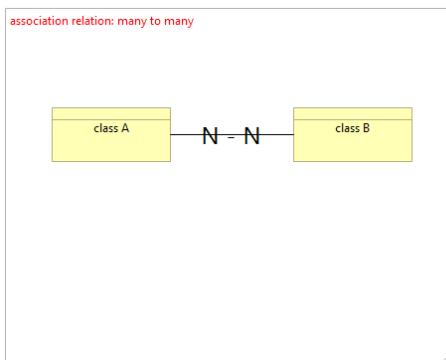
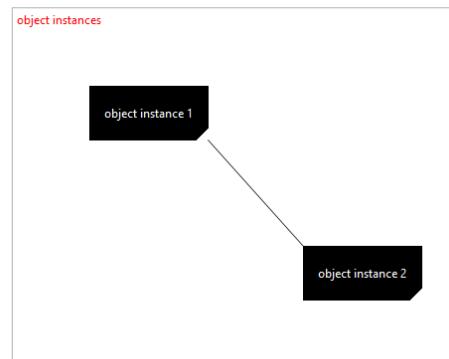
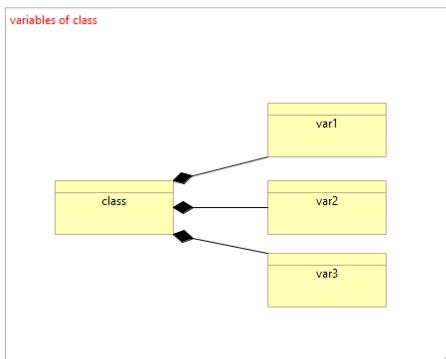
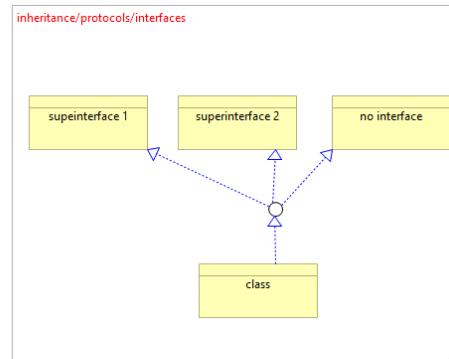
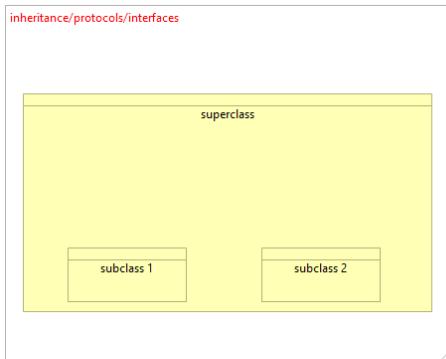
array of value objects



3 Analysis patterns



USED NOTATION

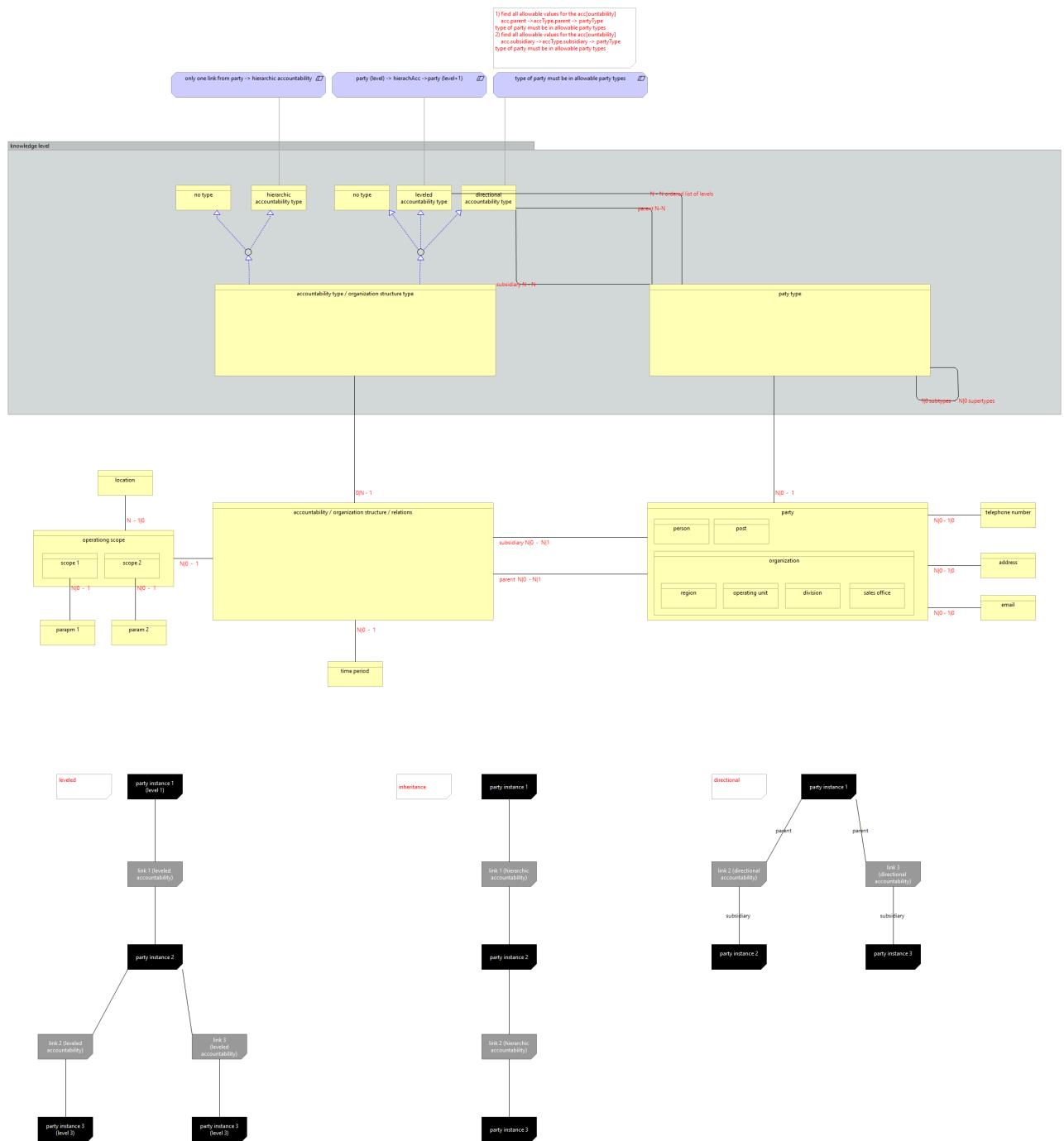


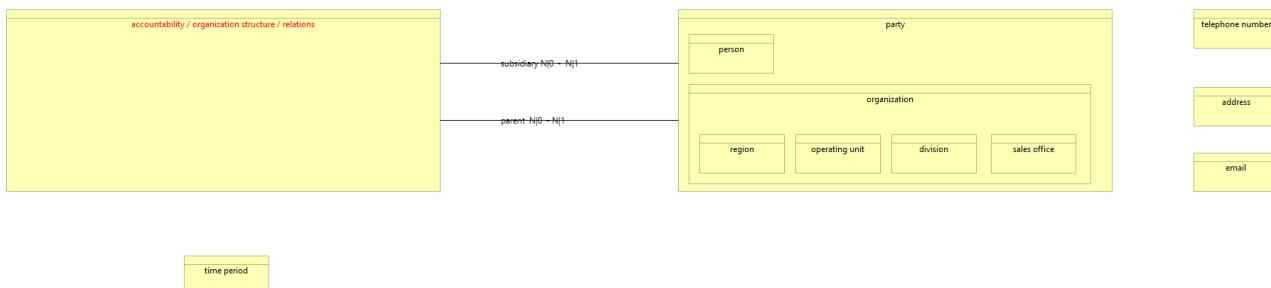
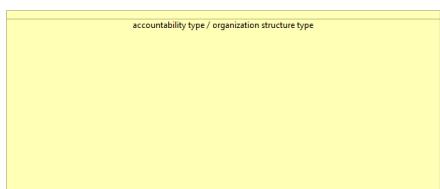
ACCOUNTABILITY

PARTY

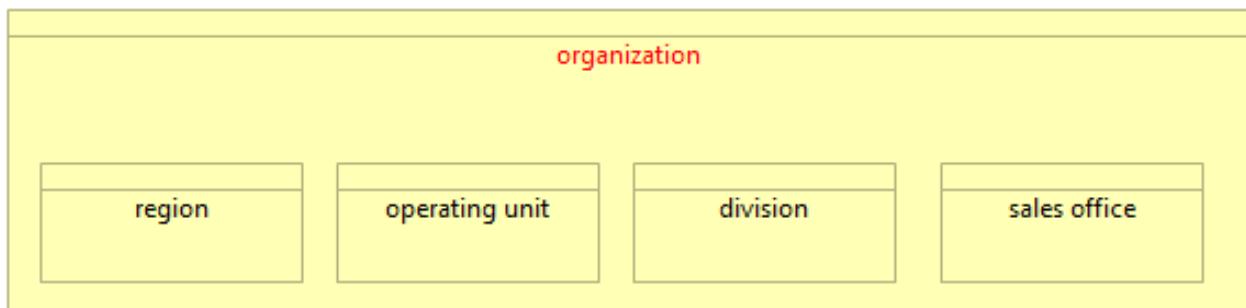


ACCOUNTABILITY

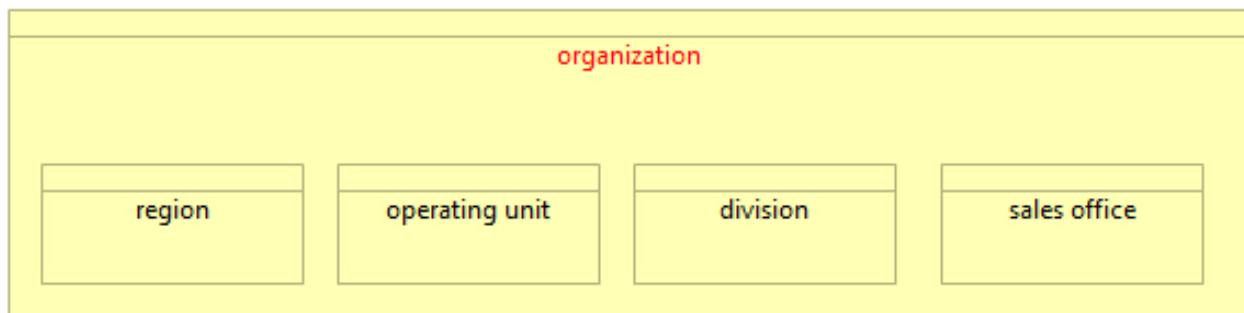




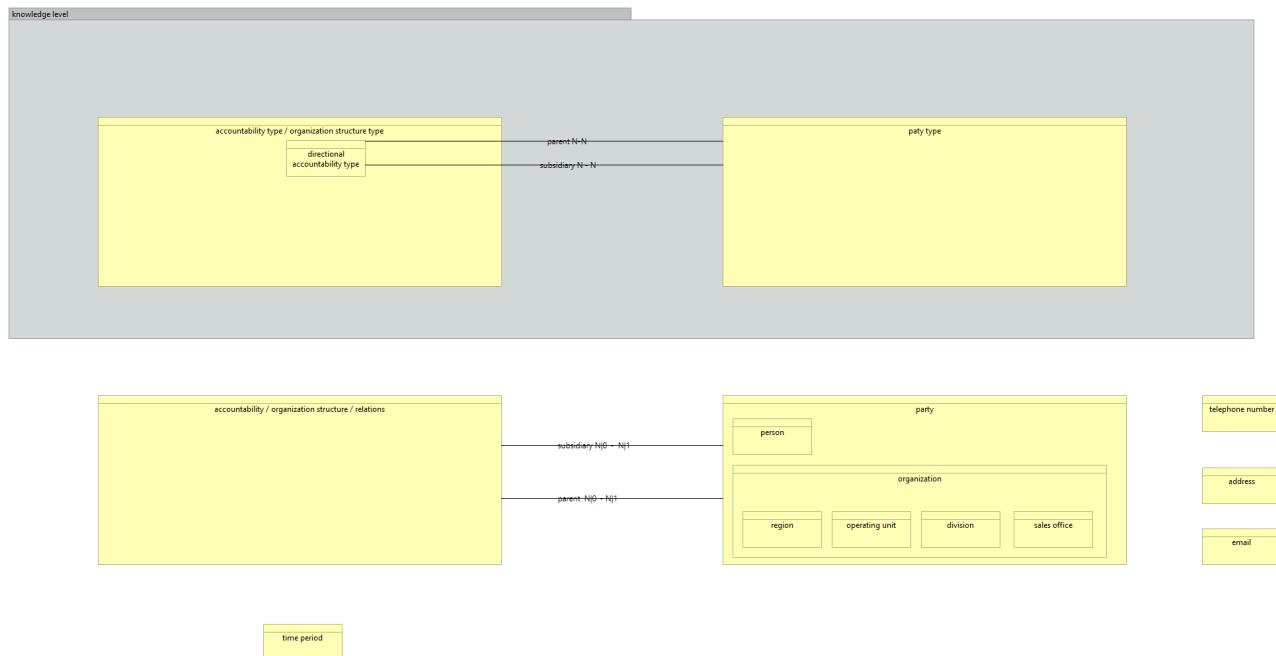
ORGANIZATION HIERARCHIES



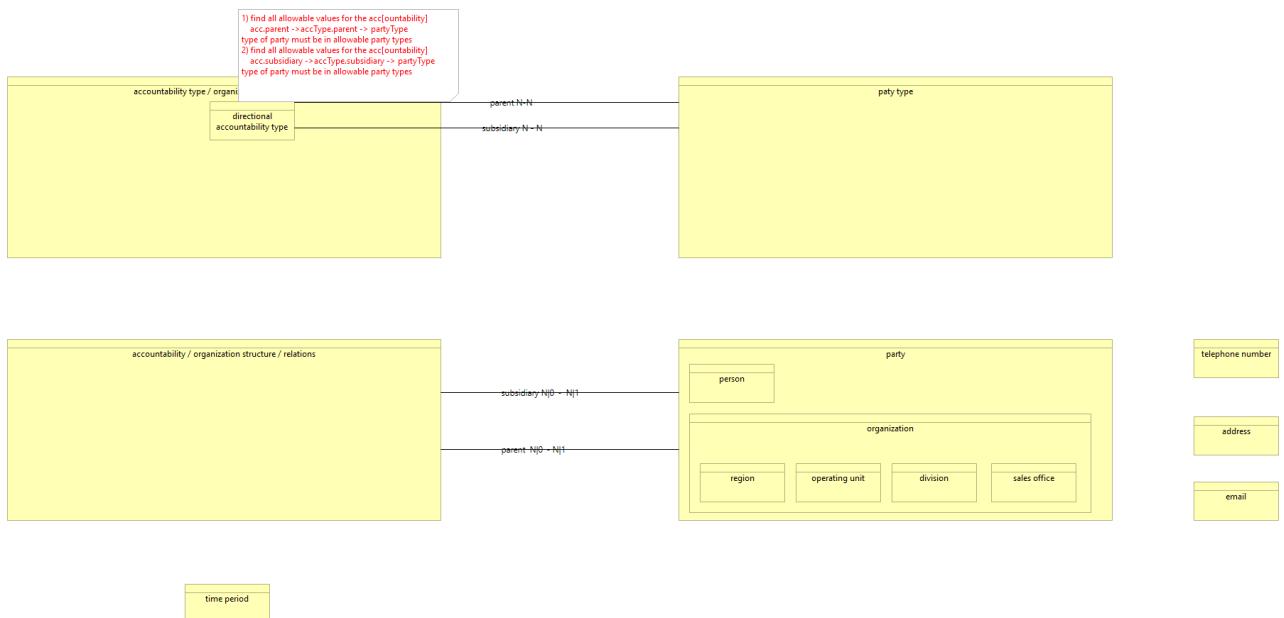
ORGANIZATION STRUCTURE



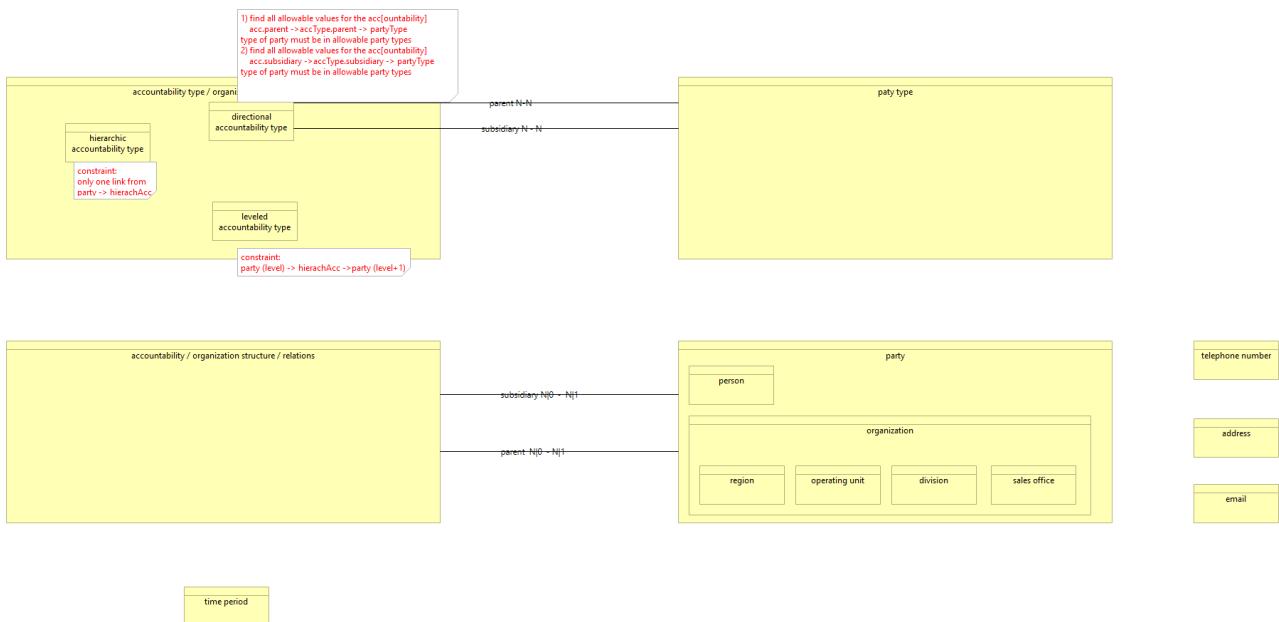
ACCOUNTABILITY KNOWLEDGE LEVEL



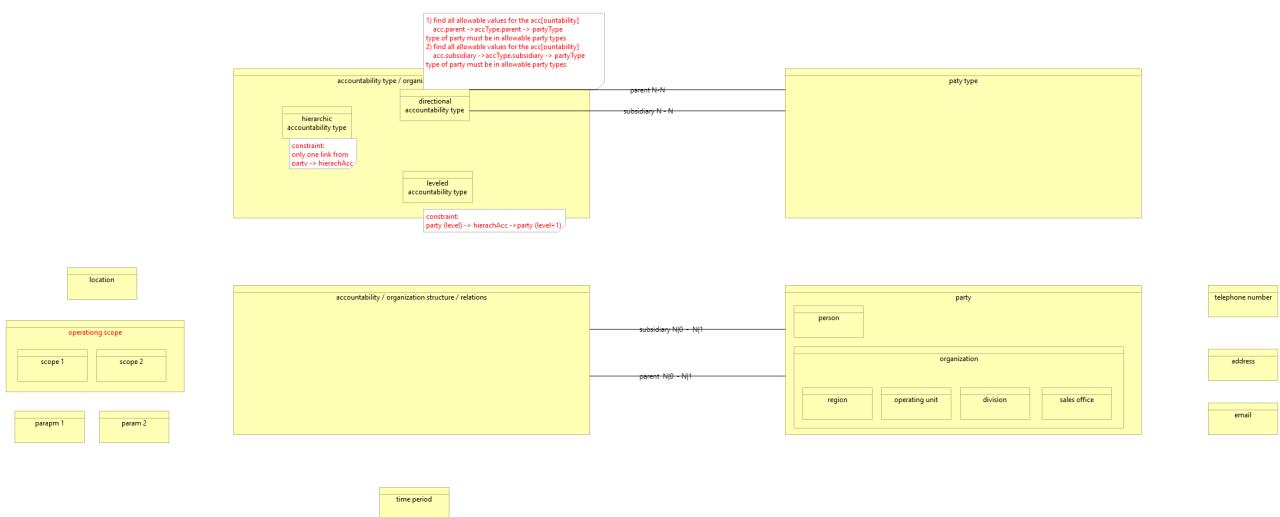
PARTY TYPE GENERALIZATIONS



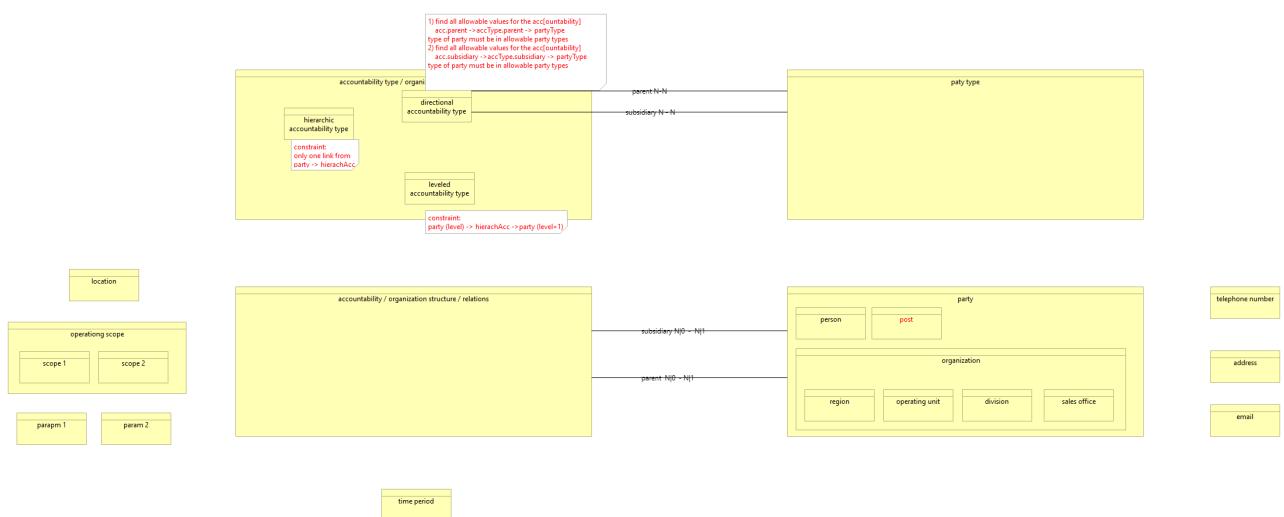
HIERARCHIC ACCOUNTABILITY



OPERATING SCOPES

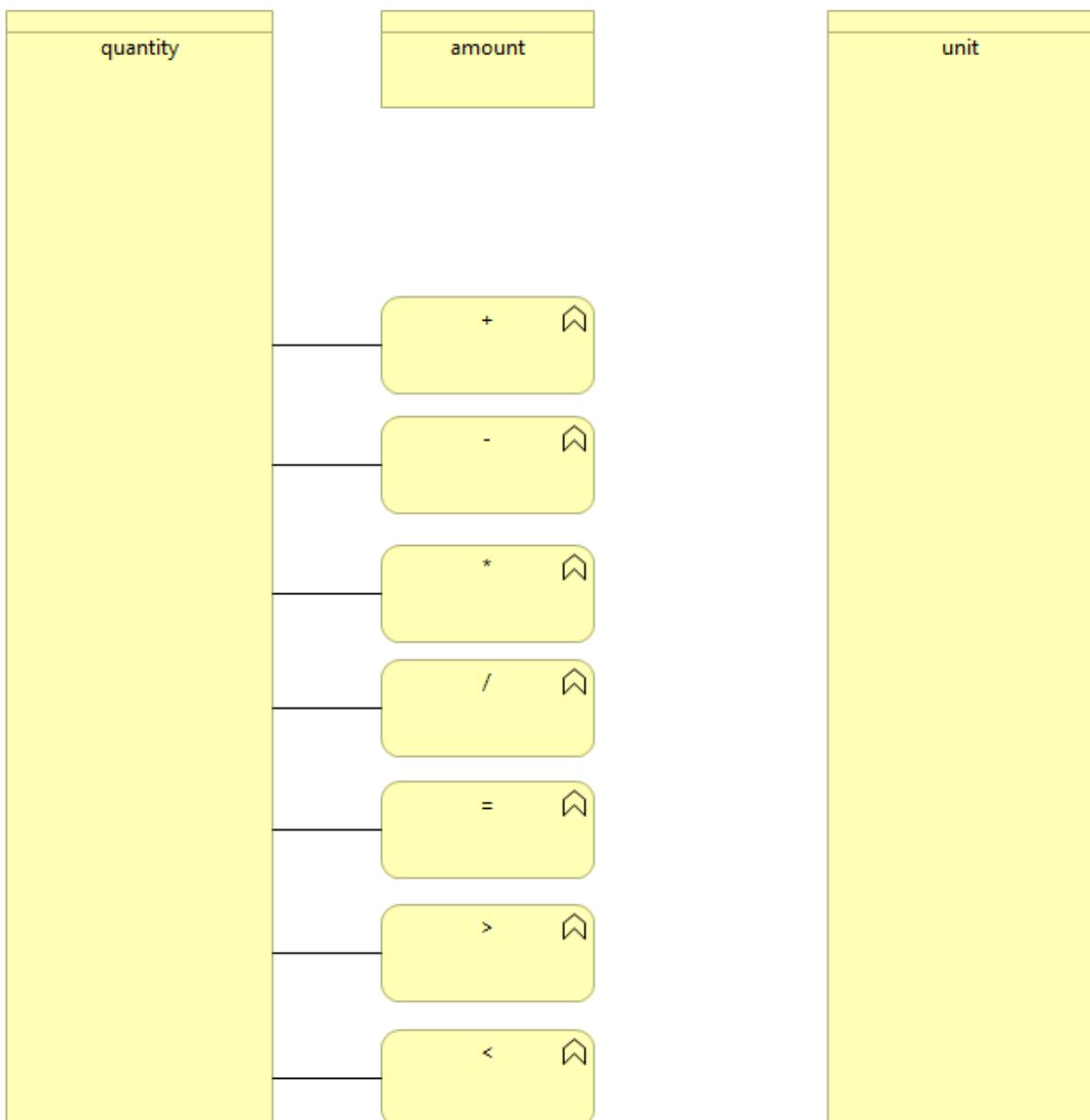


POST



OBSERVATIONS AND MEASUREMENTS

QUANTITY

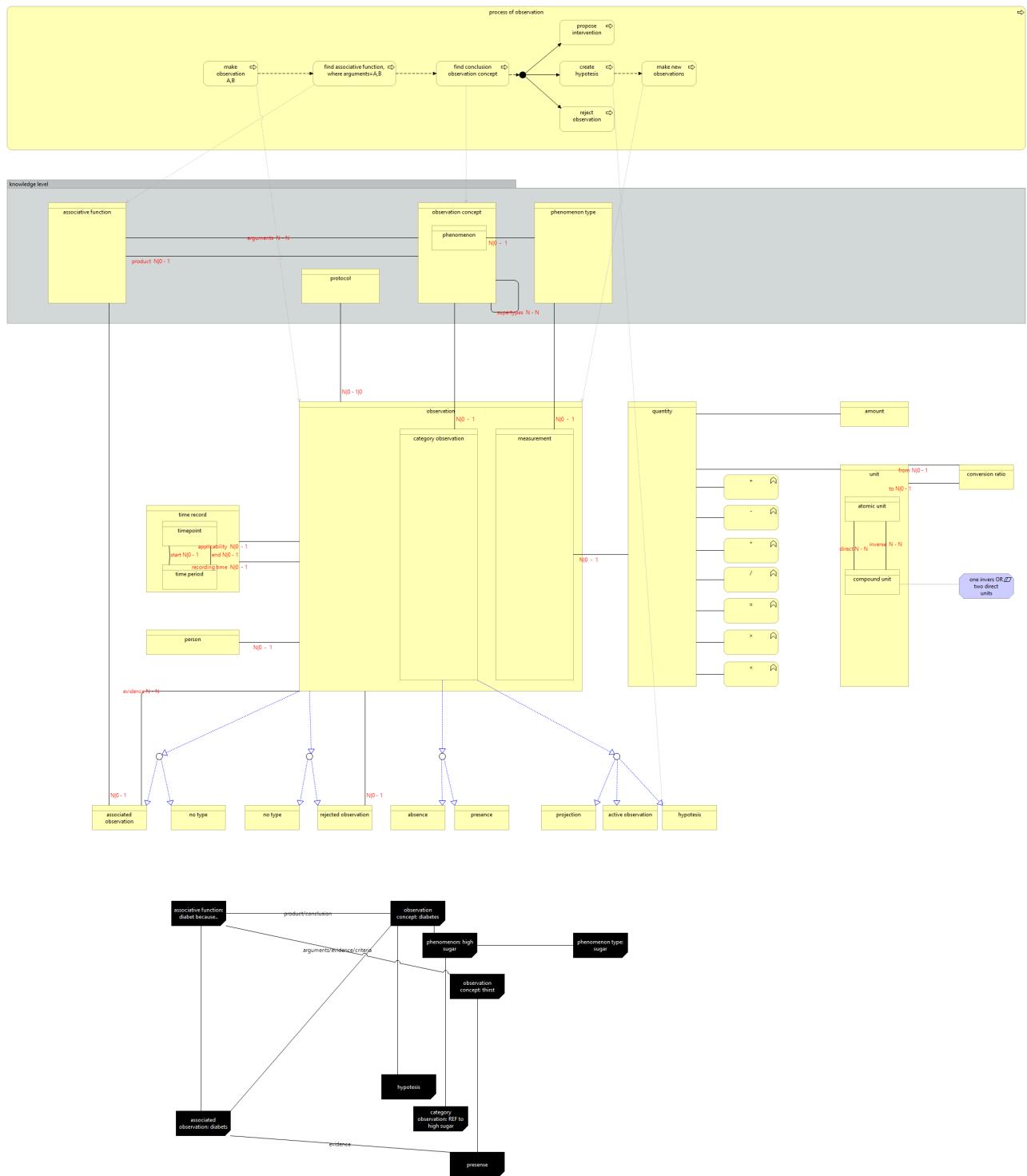


CONVERSION RATIO

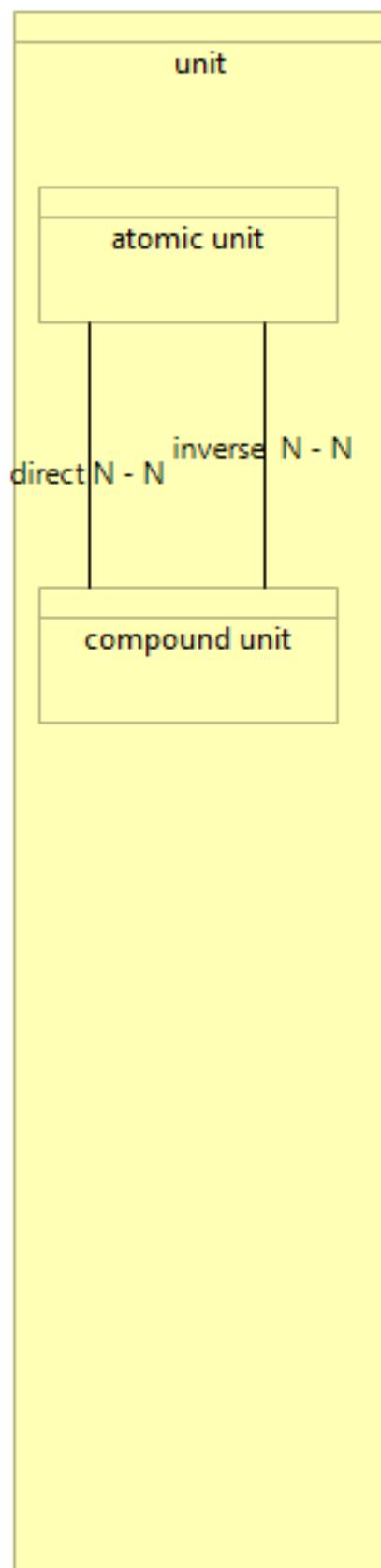
unit

conversion ratio

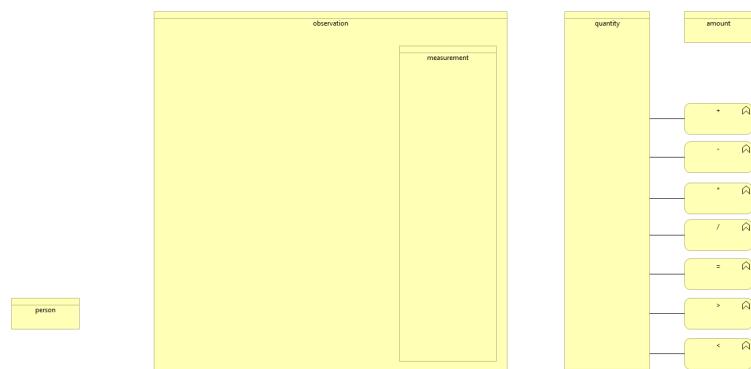
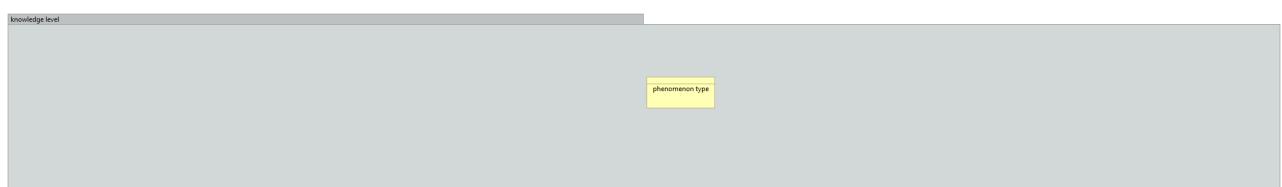
OBSERVATIONS AND MEASUREMENTS



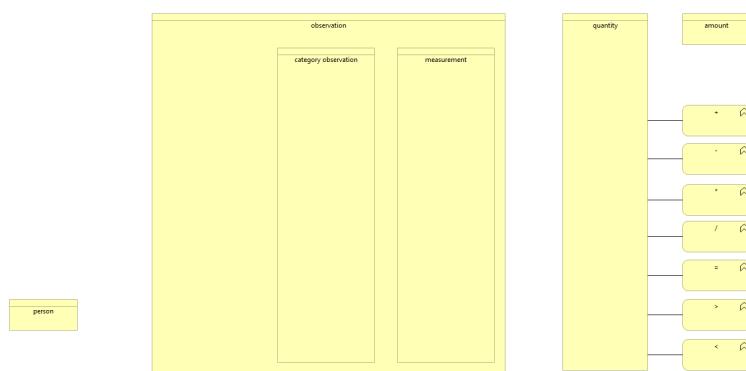
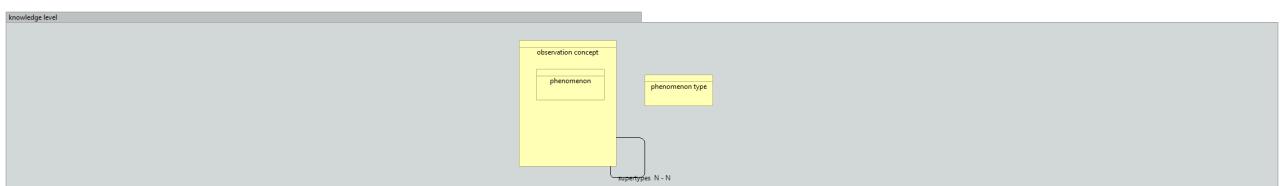
COMPOUND UNITS



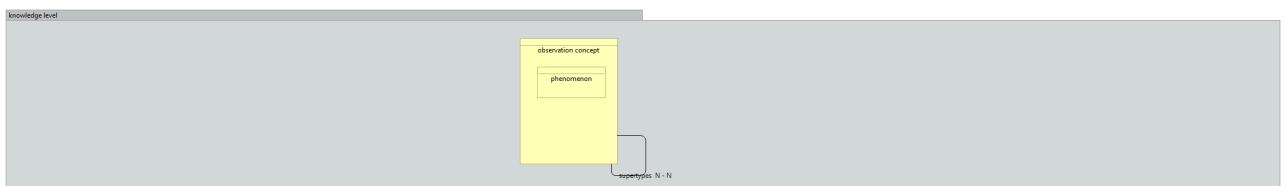
MEASUREMENT



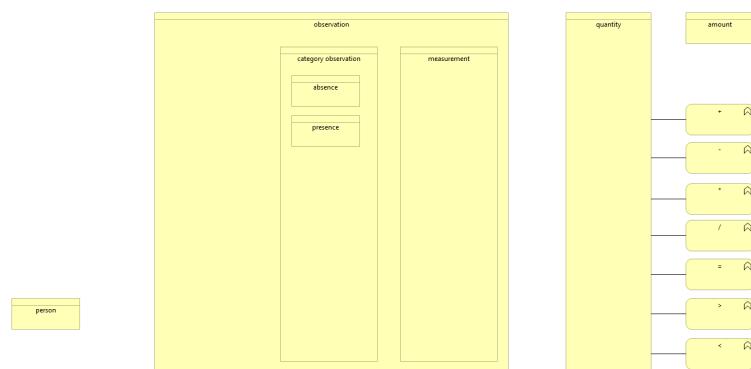
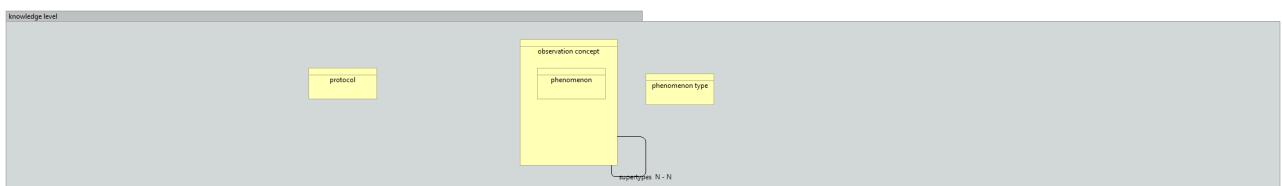
OBSERVATION



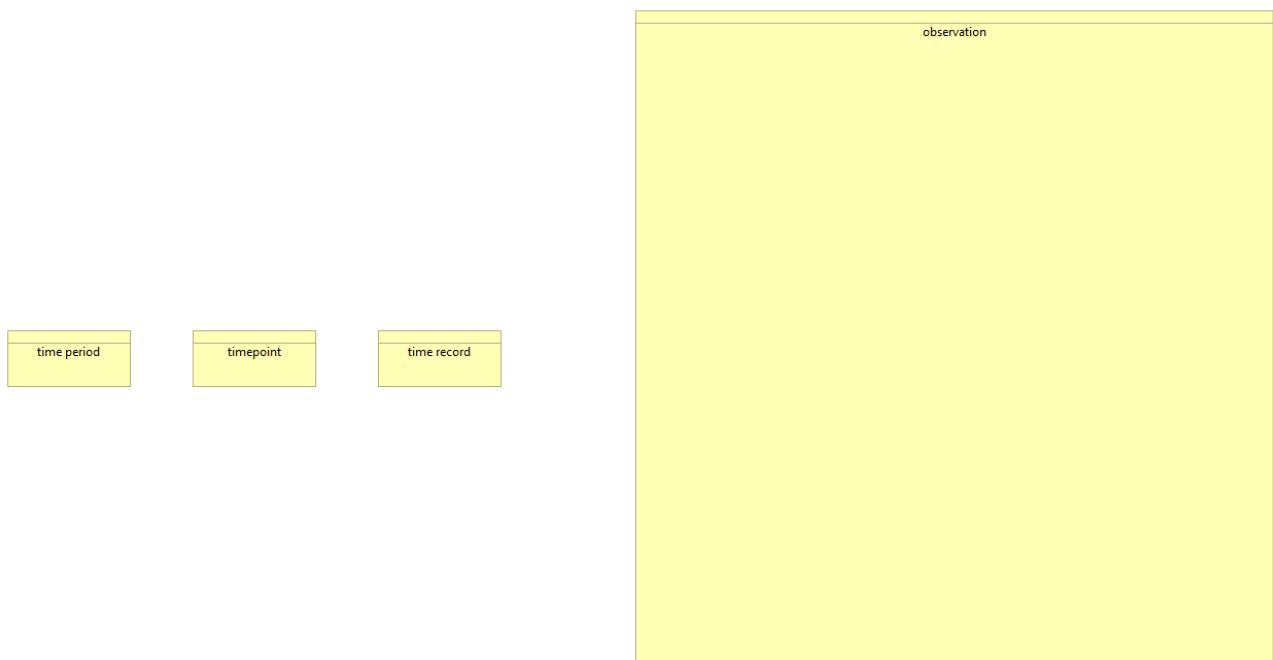
SUBTYPING OBSERVATION CONCEPTS



PROTOCOL



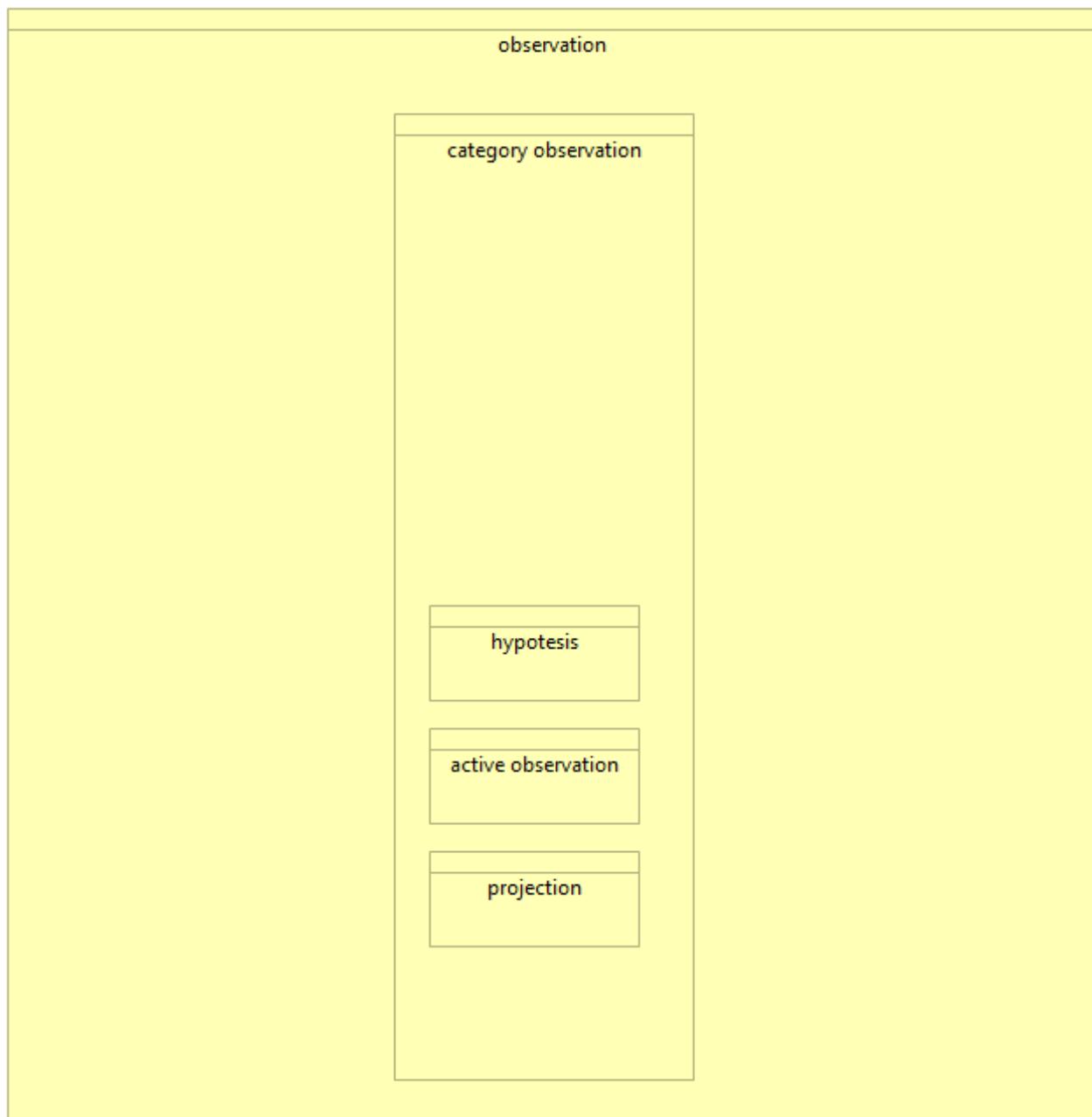
DUAL TIME RECORD



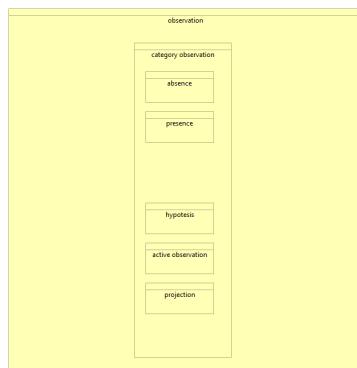
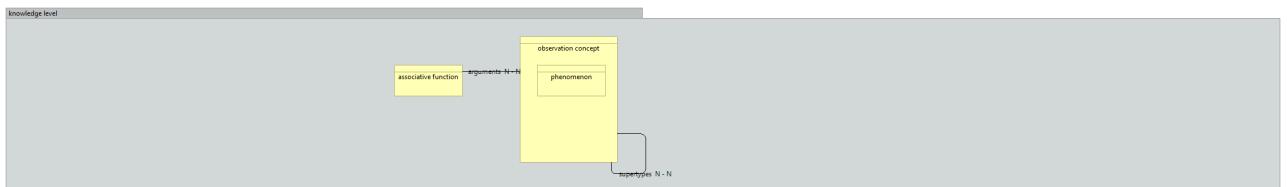
REJECTED OBSERVATION

observation

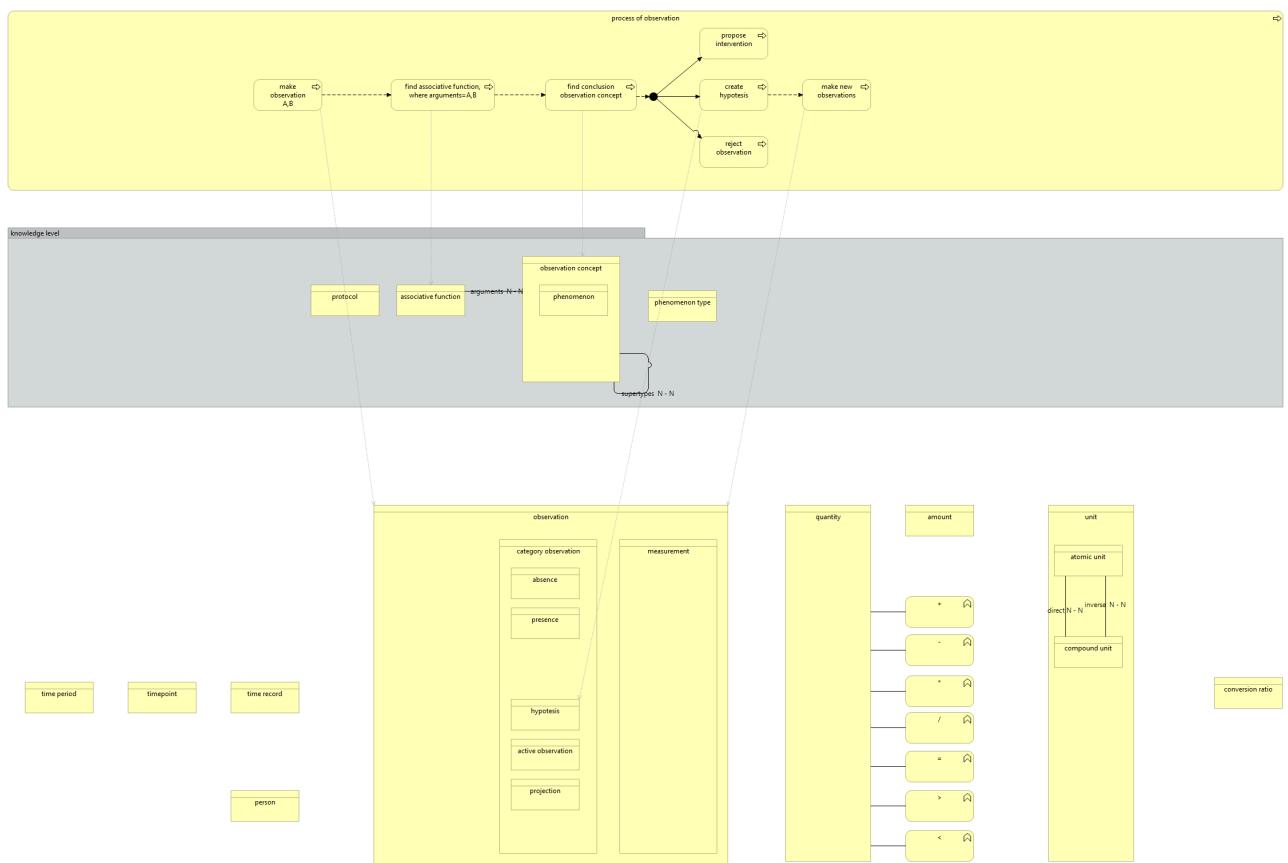
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION



ASSOCIATED OBSERVATION

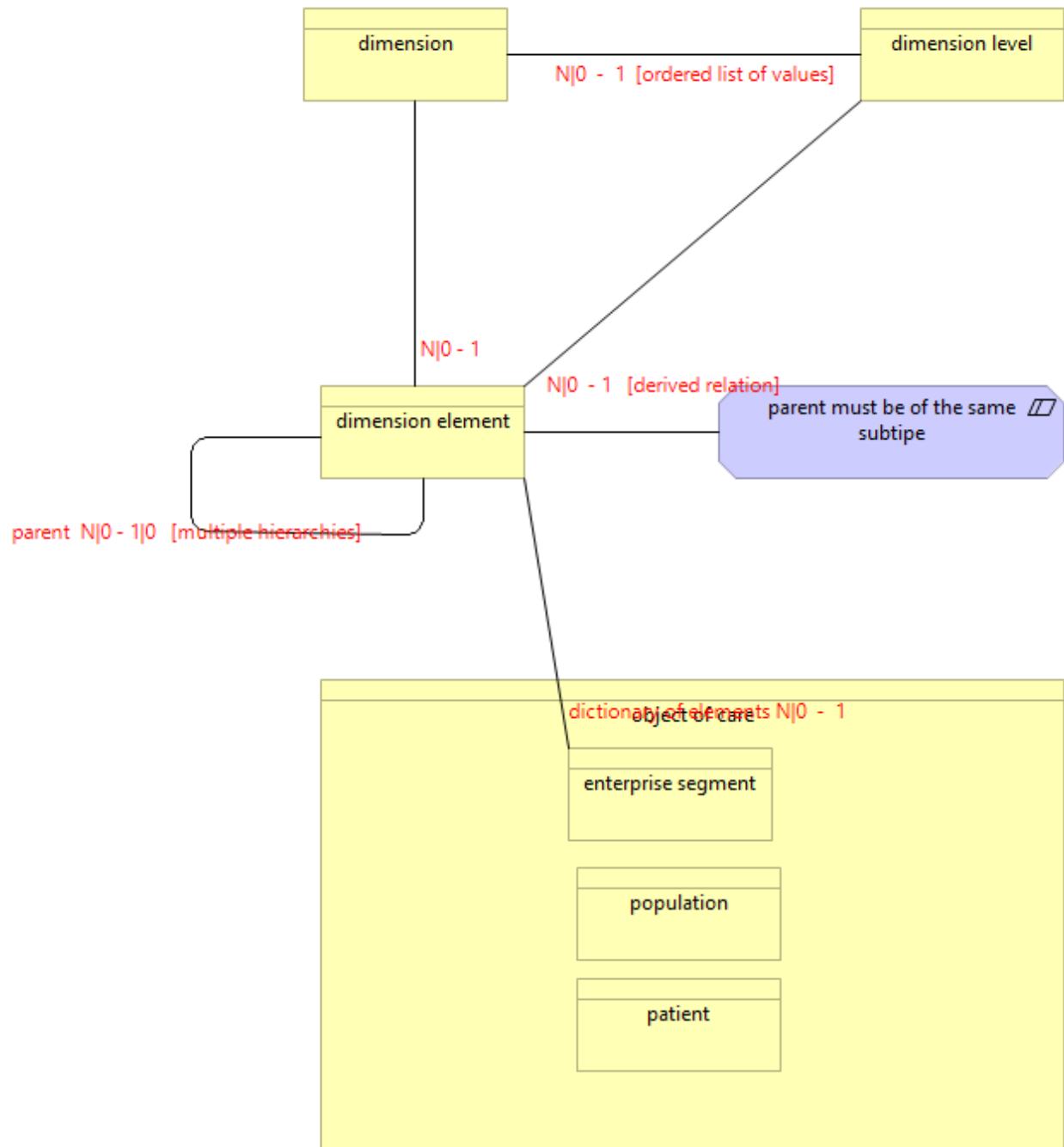


PROCESS OF OBSERVATION

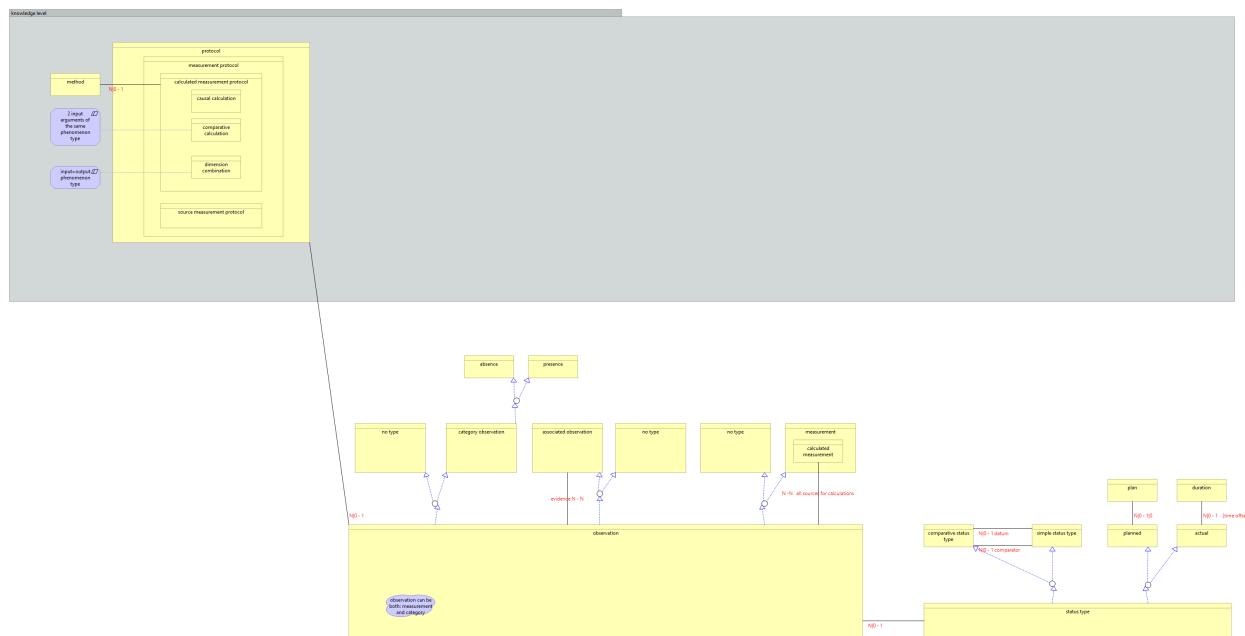


OBSERVATIONS FOR CORPORATE FINANCE

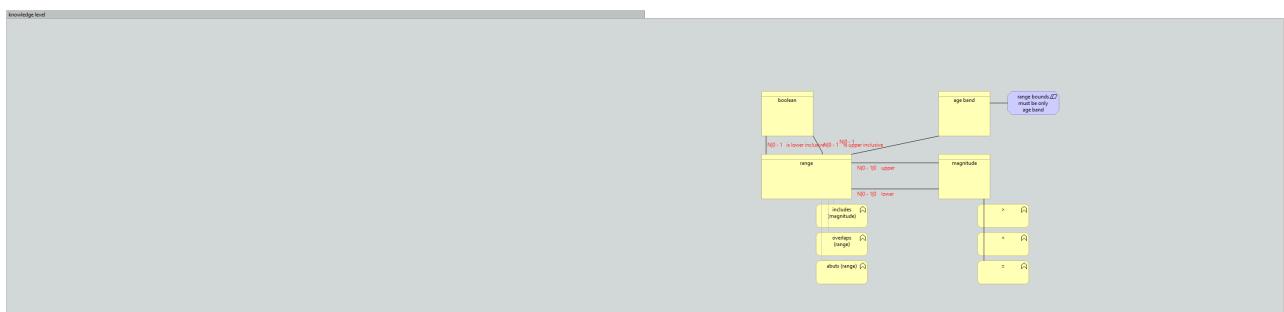
ENTERPRISE SEGMENT



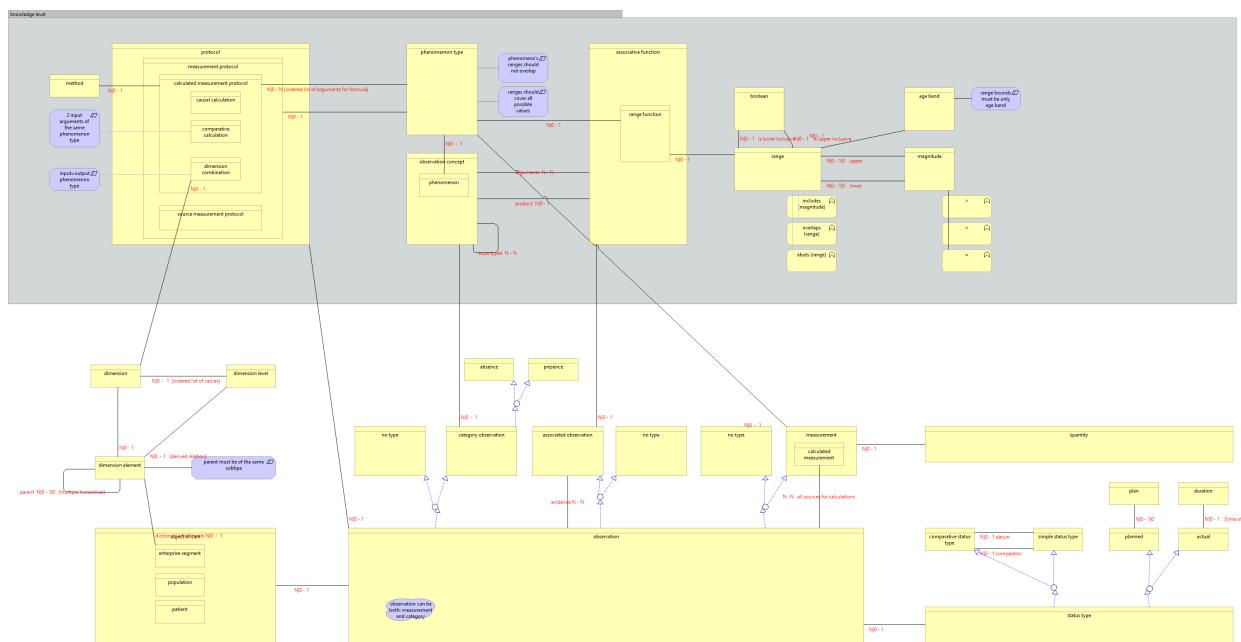
MEASUREMENT PROTOCOL

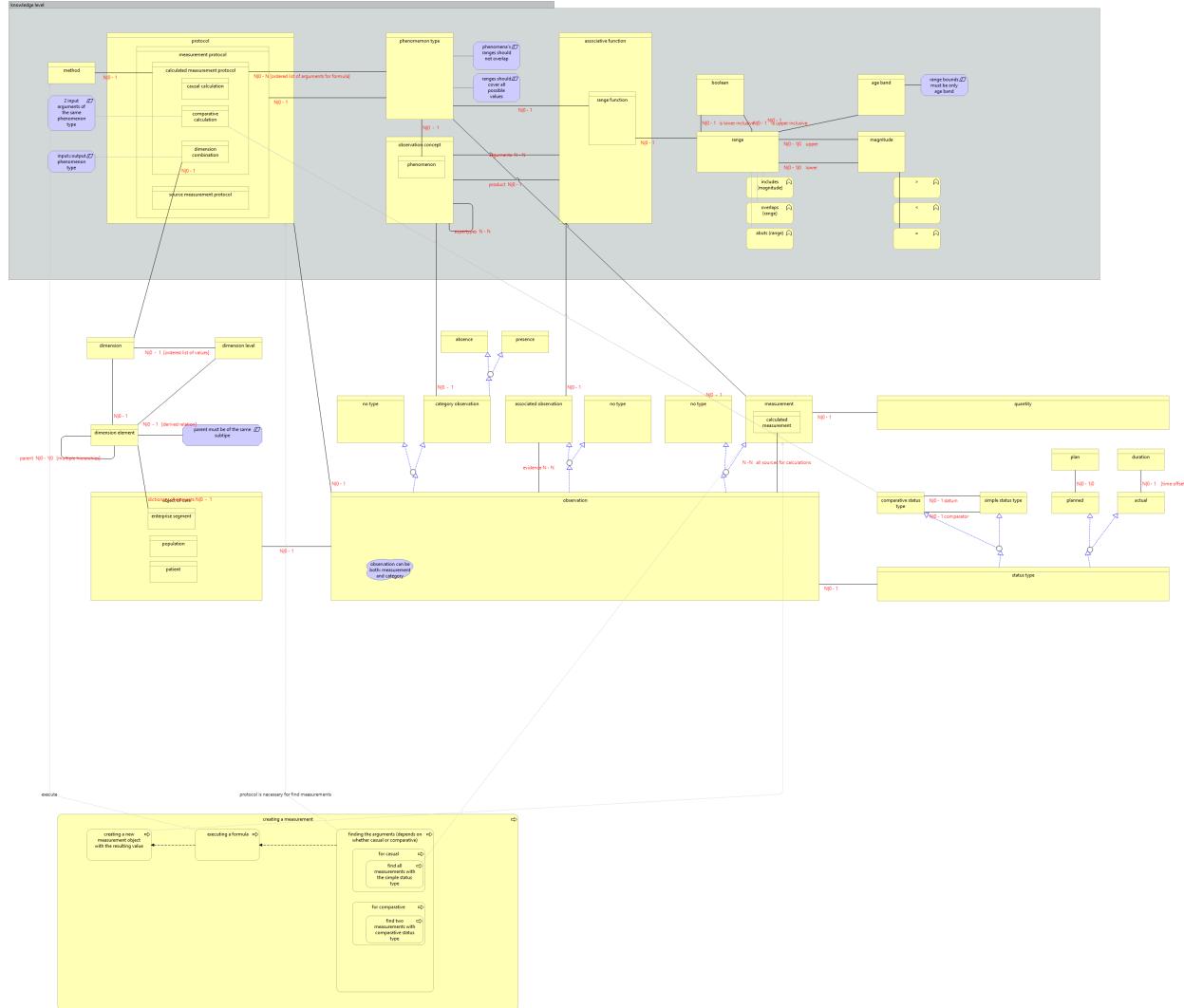


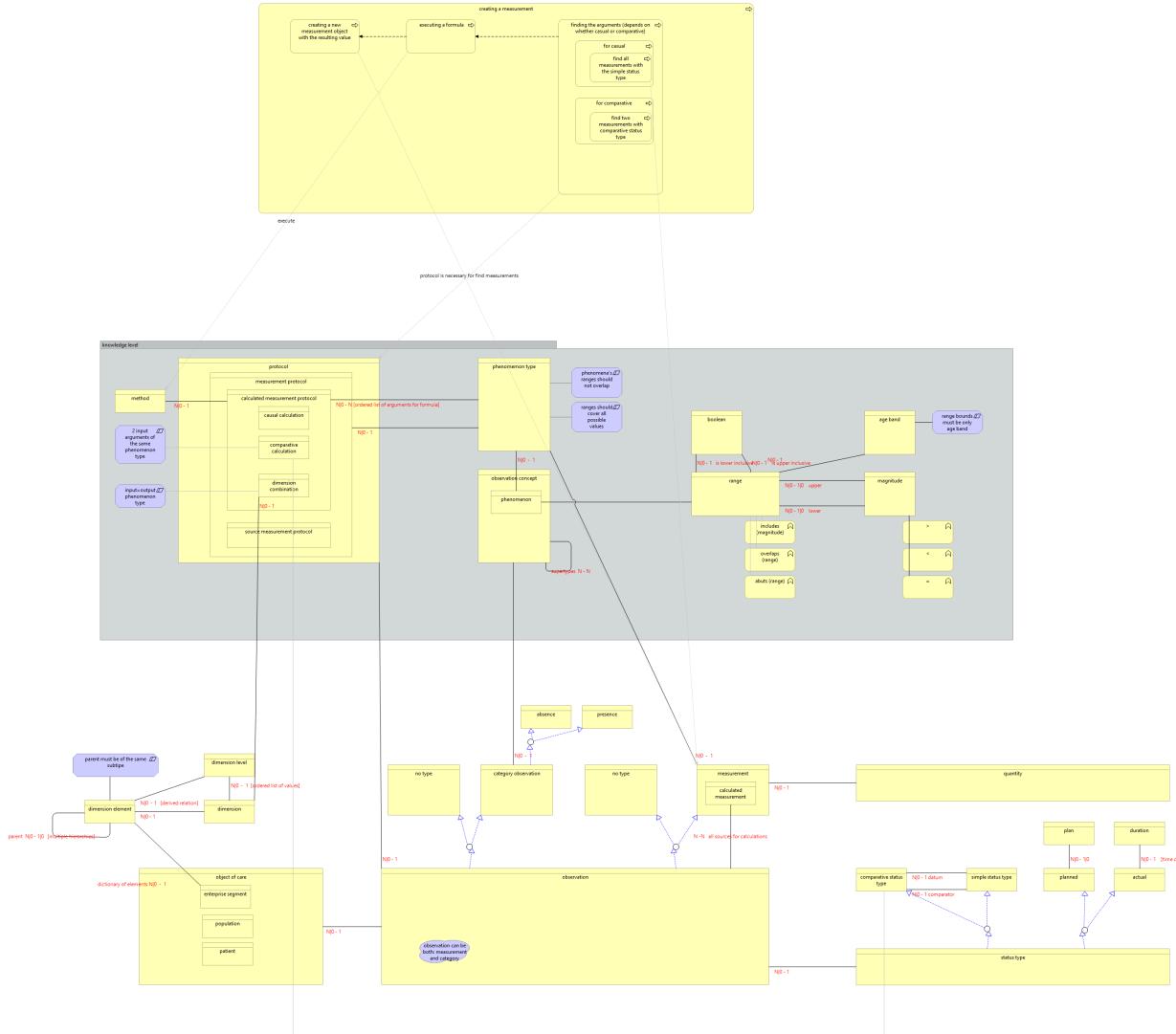
RANGE



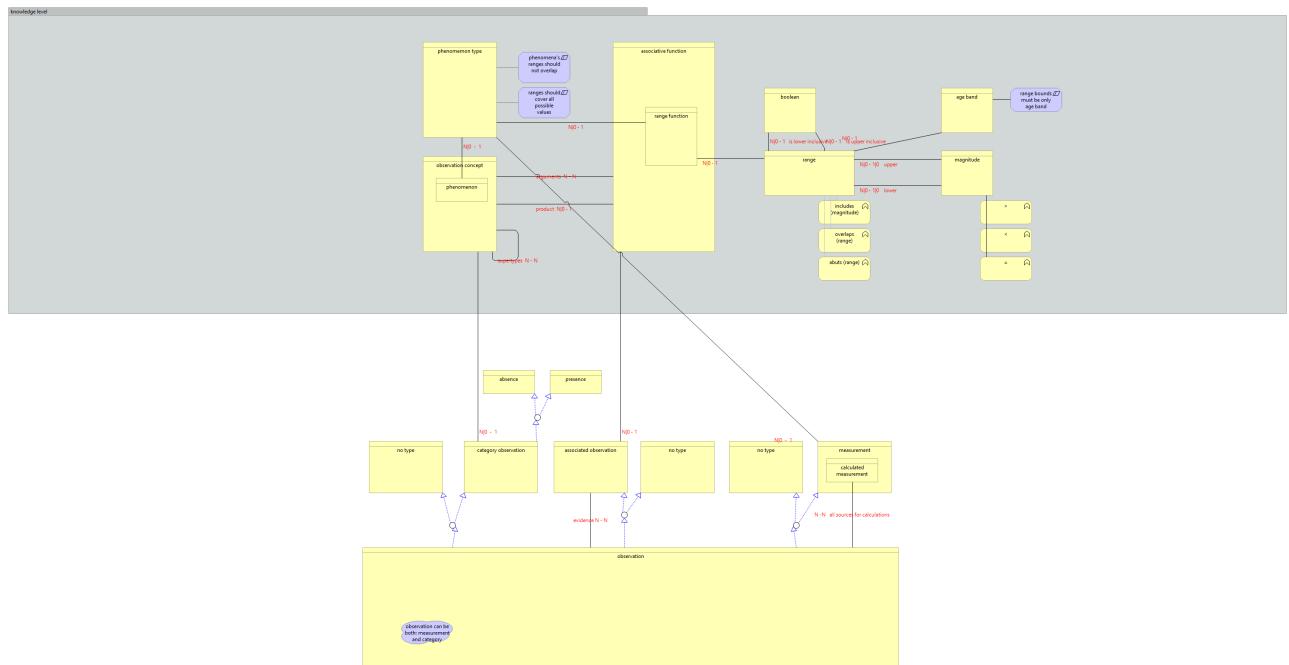
OBSERVATIONS FOR CORPORATE FINANCE





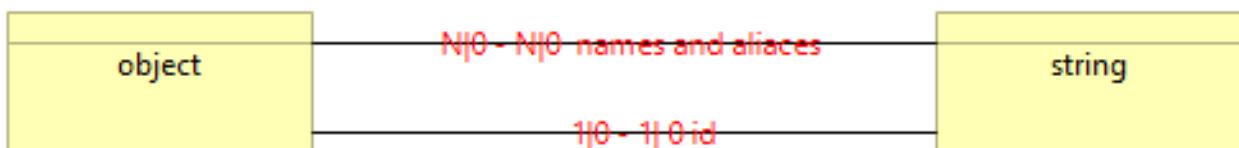


PHENOMENON WITH RANGE

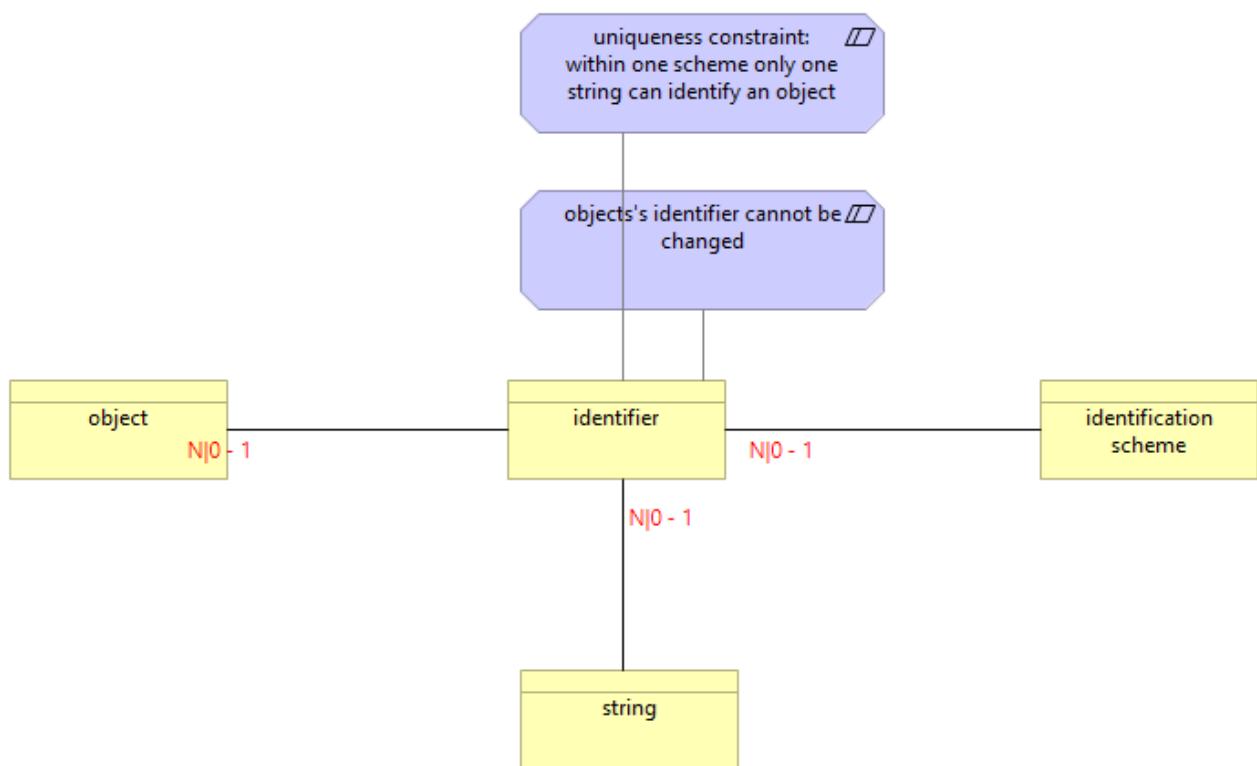


REFERRING TO OBJECTS

NAME



IDENTIFICATION SCHEME

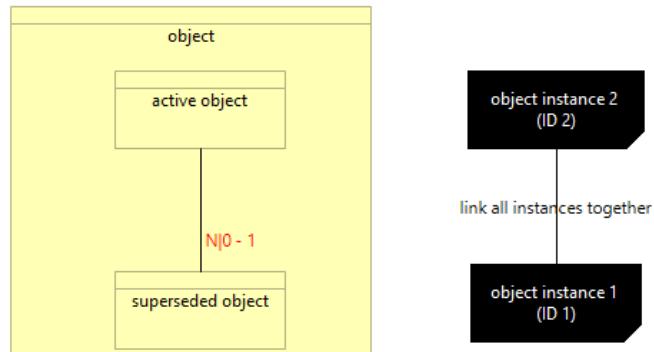


OBJECT MERGE

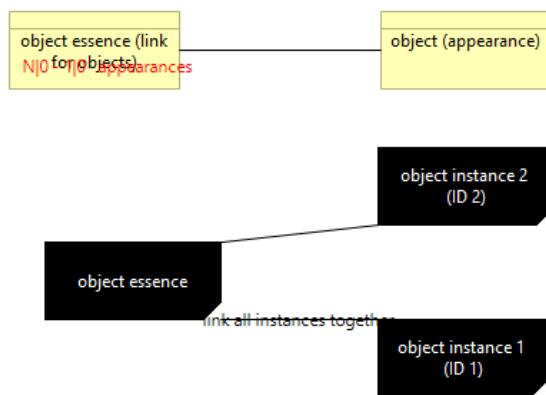
object merge strategy:
copy and replace



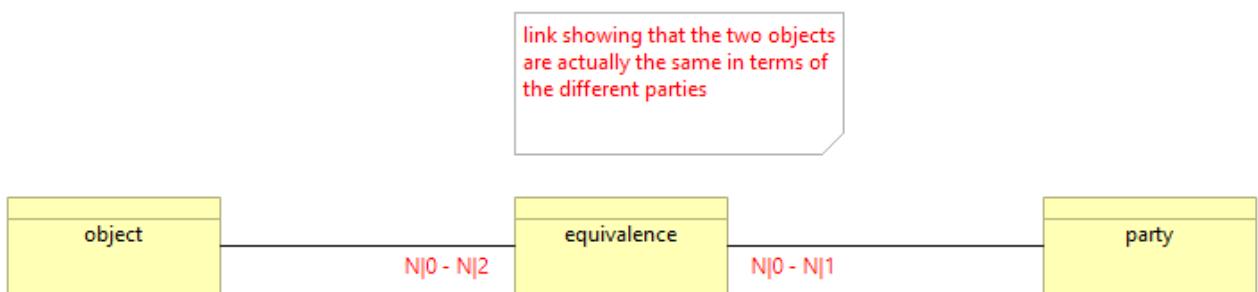
object merge strategy:
superseeded



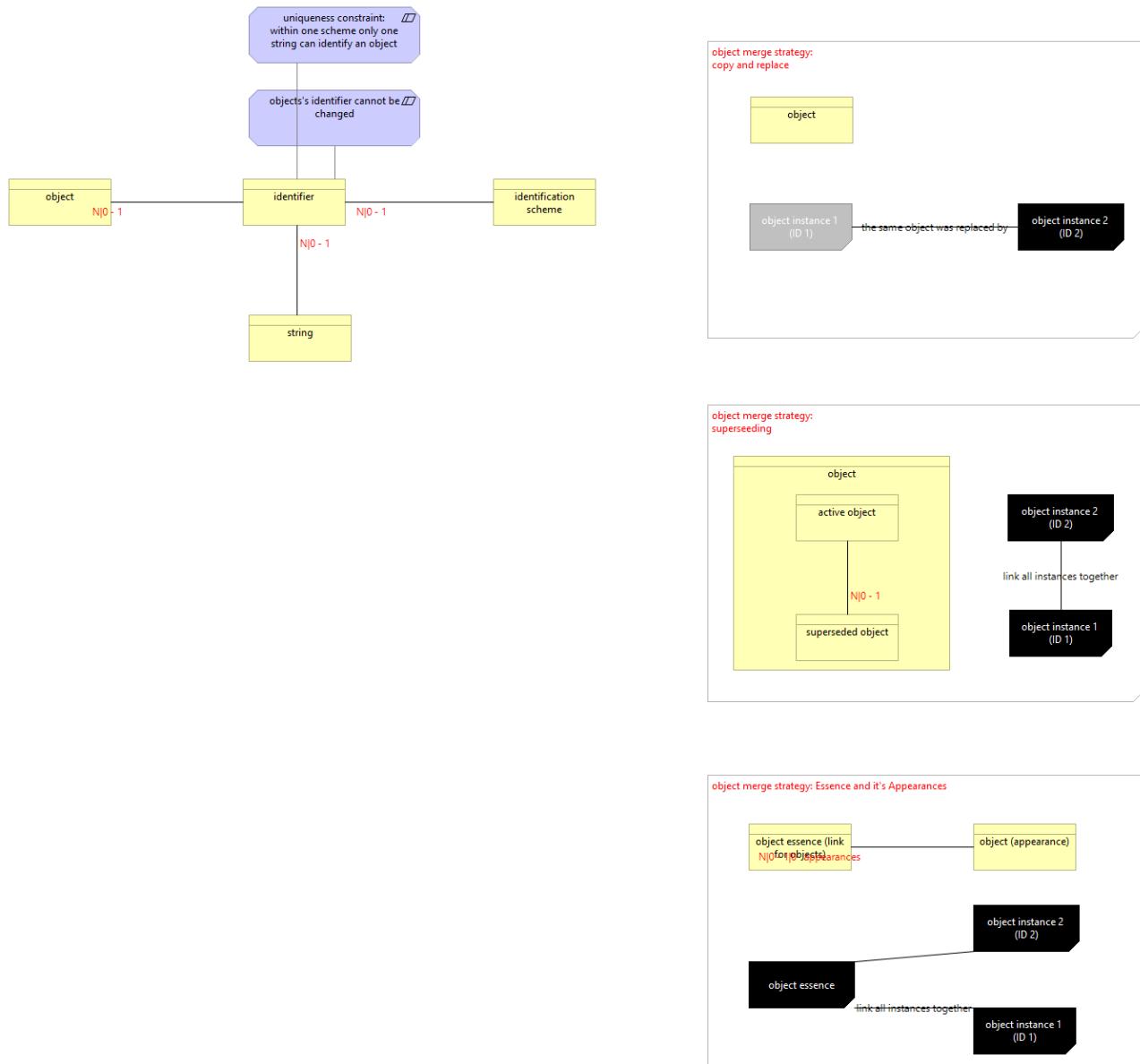
object merge strategy: Essence and its Appearances



OBJECT EQUIVALENCE

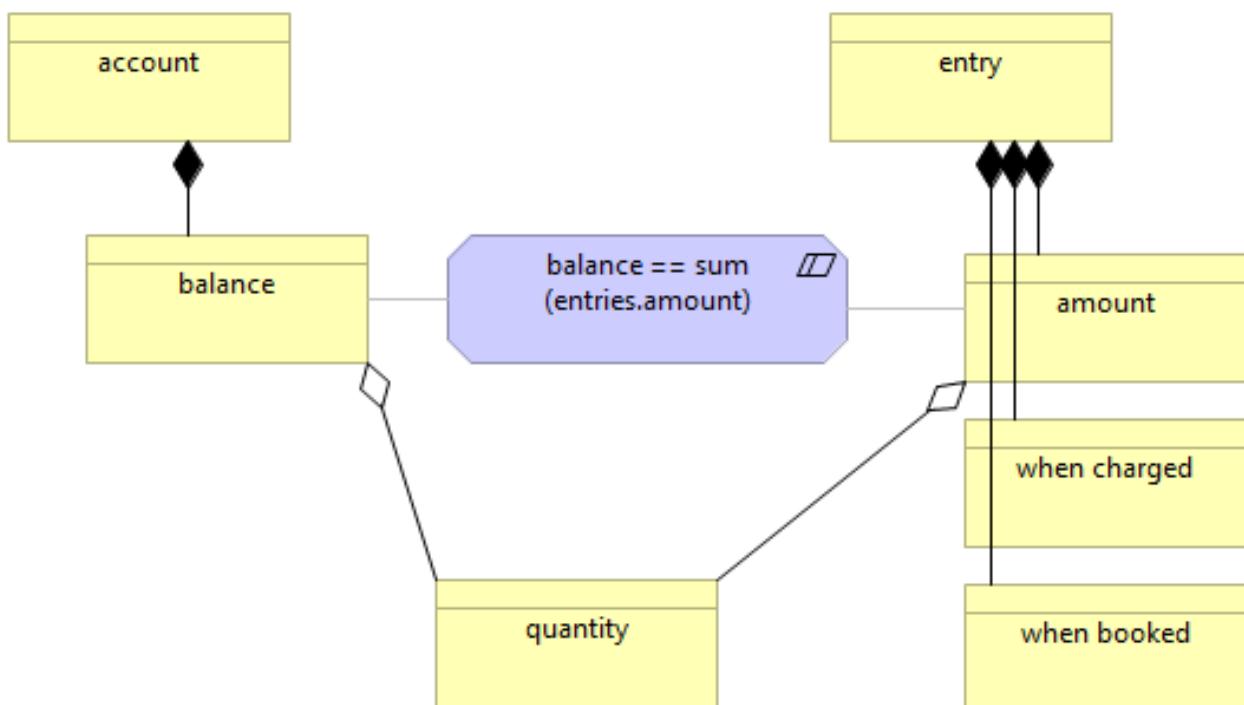


REFERRING TO OBJECTS

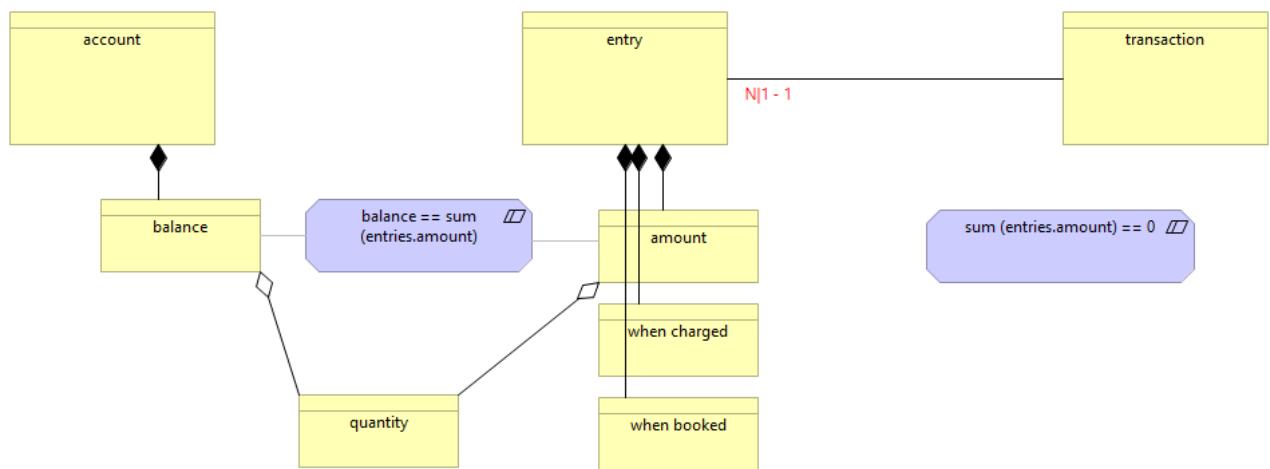


INVENTORY AND ACCOUNTING

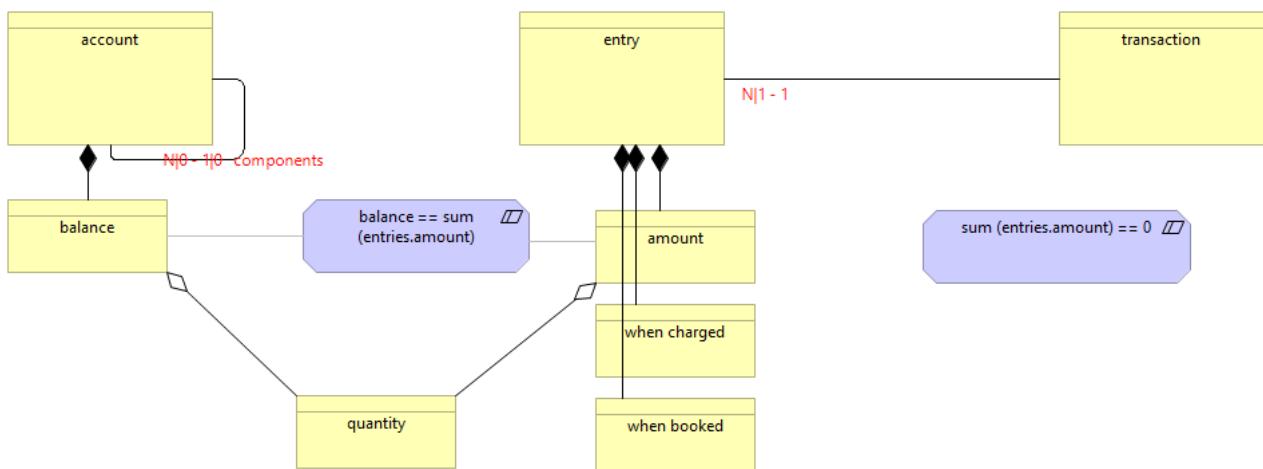
ACCOUNT



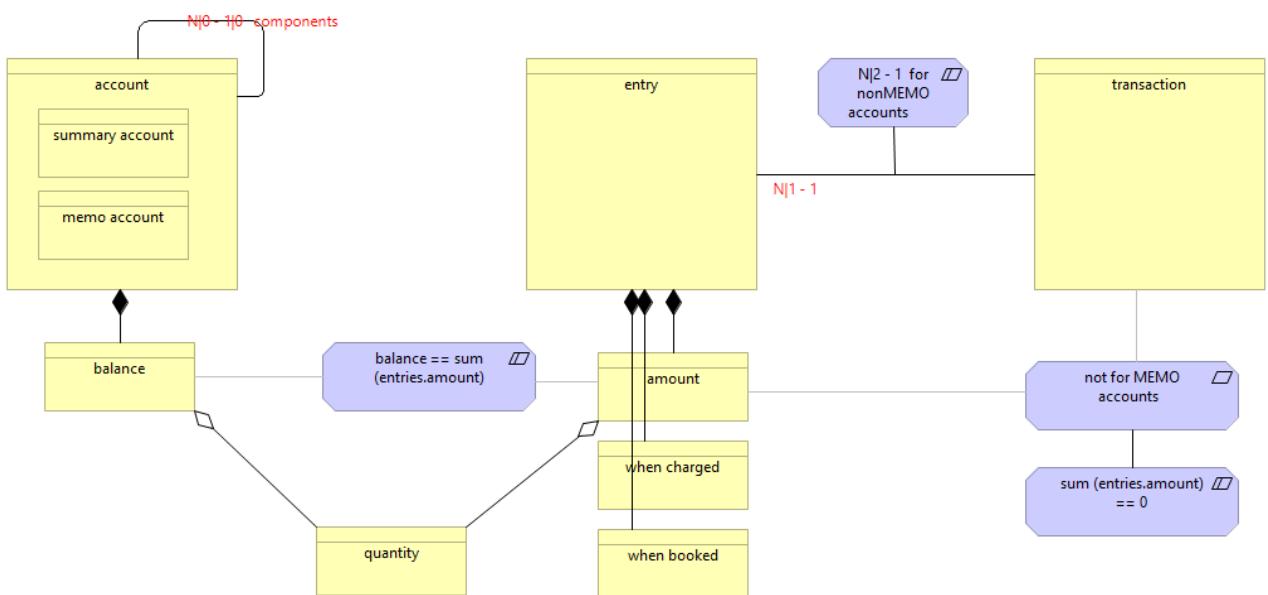
TRANSACTIONS



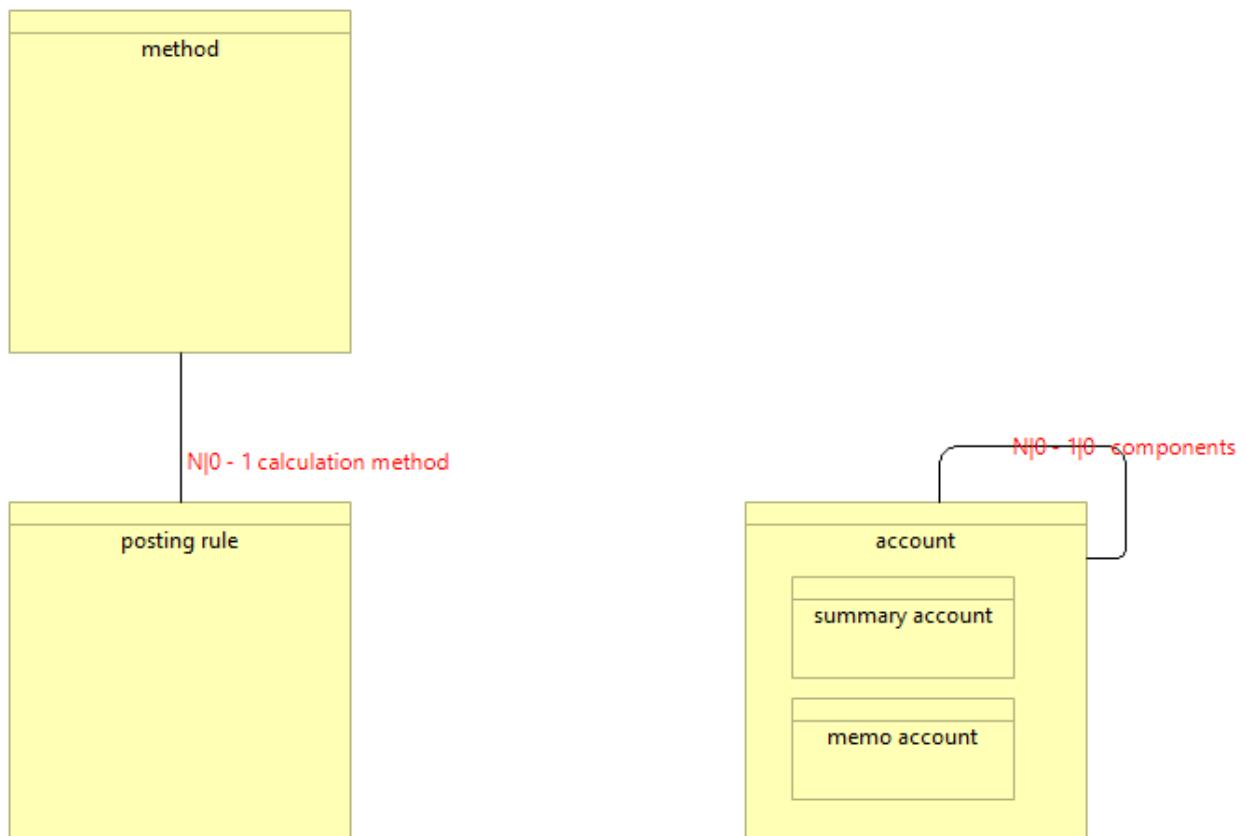
SUMMARY ACCOUNT



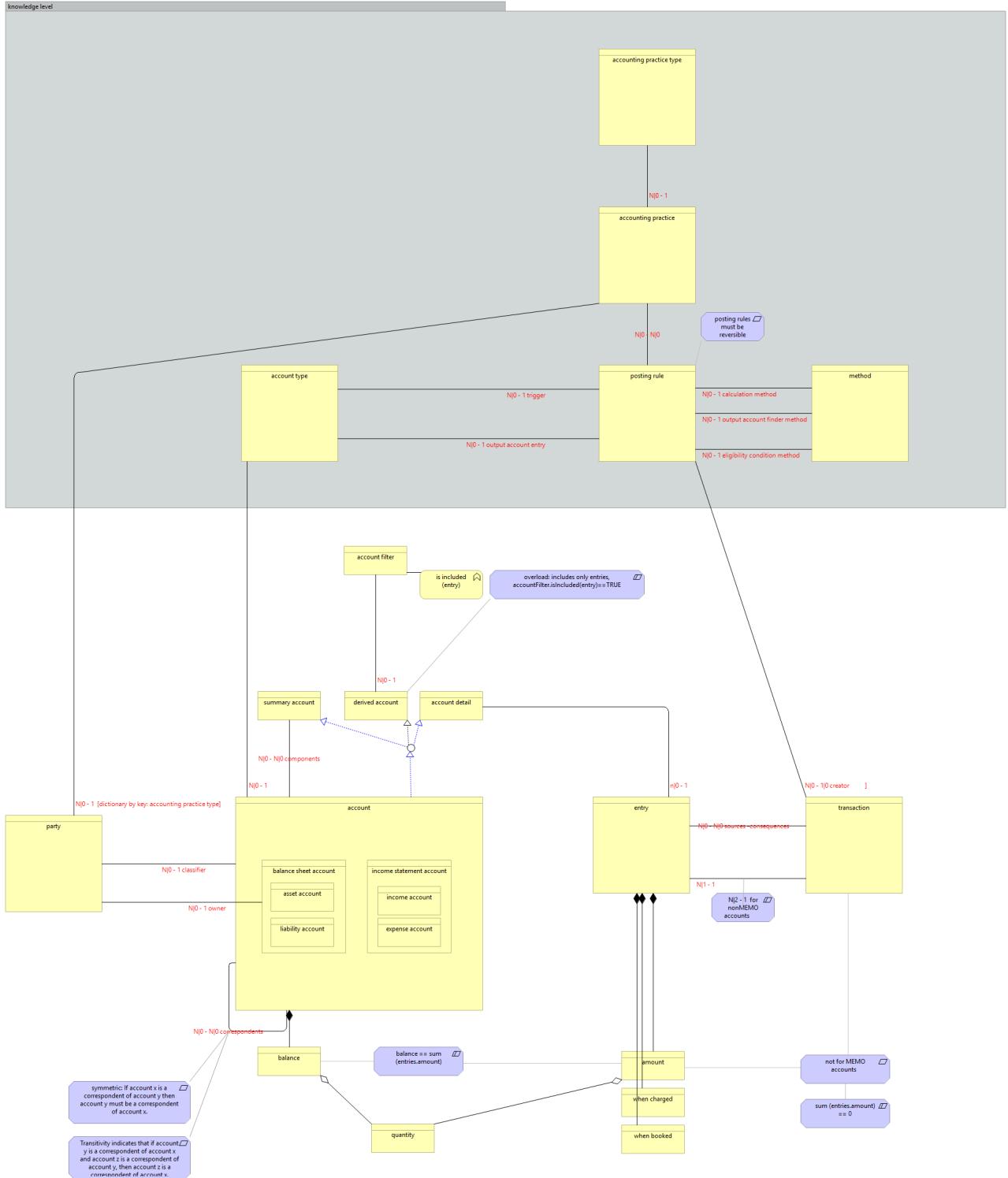
MEMO ACCOUNT



POSTING RULES



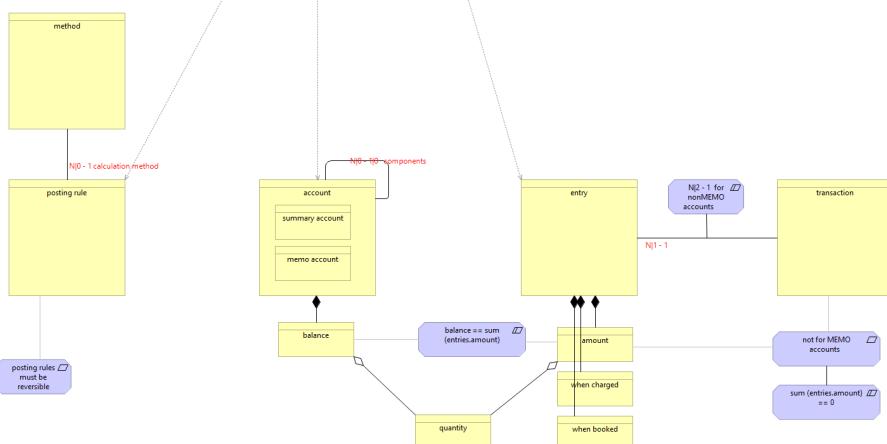
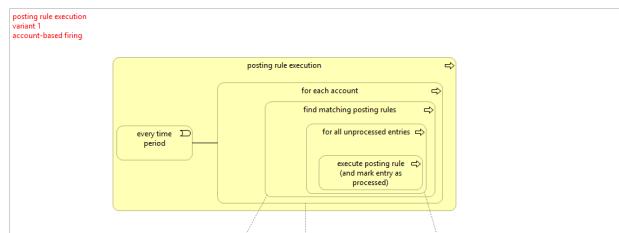
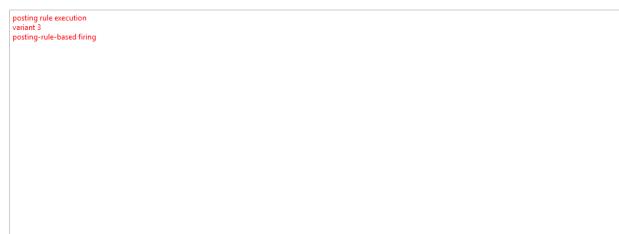
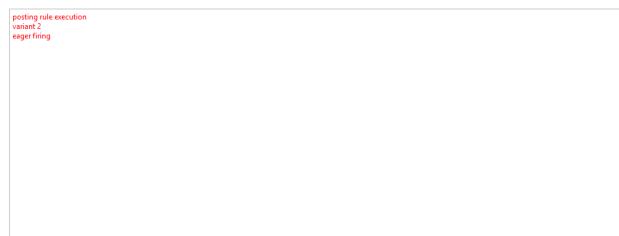
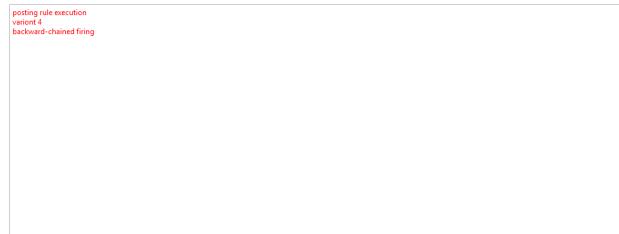
INVENTORY AND ACCOUNTING



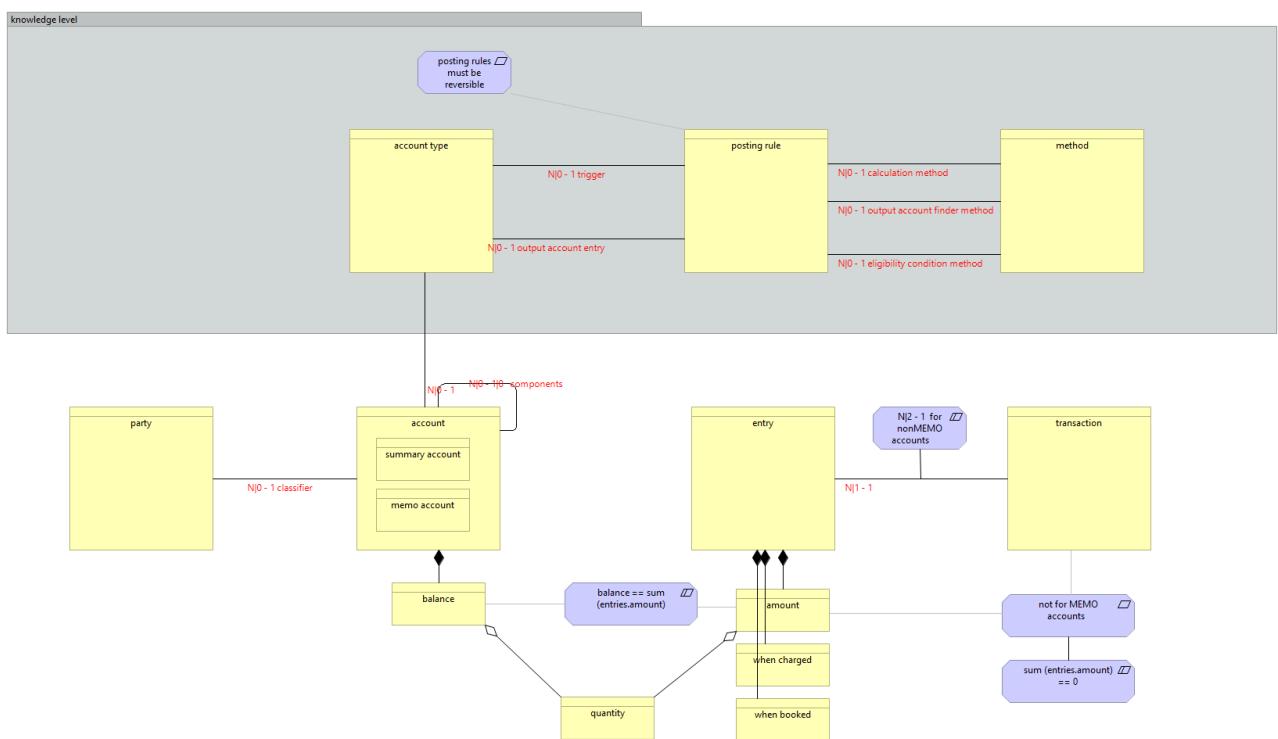
INDIVIDUAL INSTANCE METHOD



POSTING RULE EXECUTION



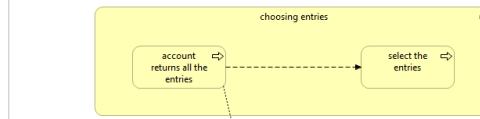
POSTING RULES FOR MANY ACCOUNTS



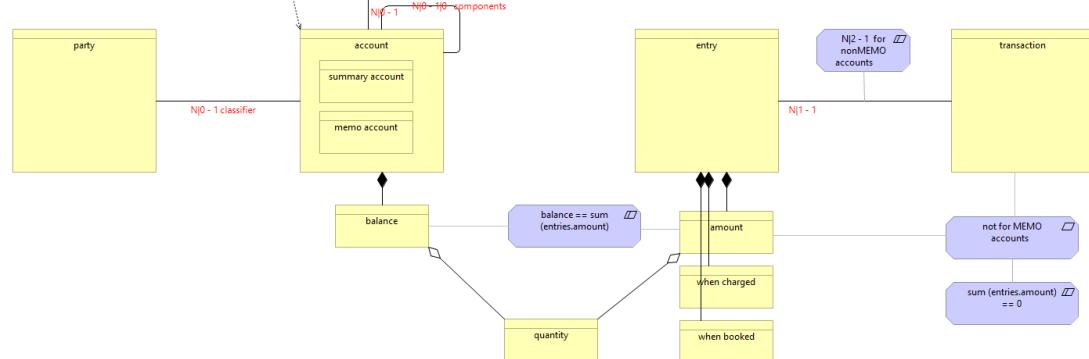
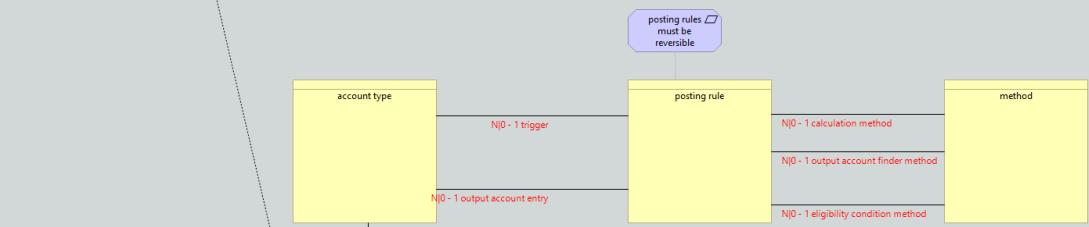
CHOOSING ENTRIES

selection of entries for the application of the posing rules:
variant 2
using account filter for select the entries

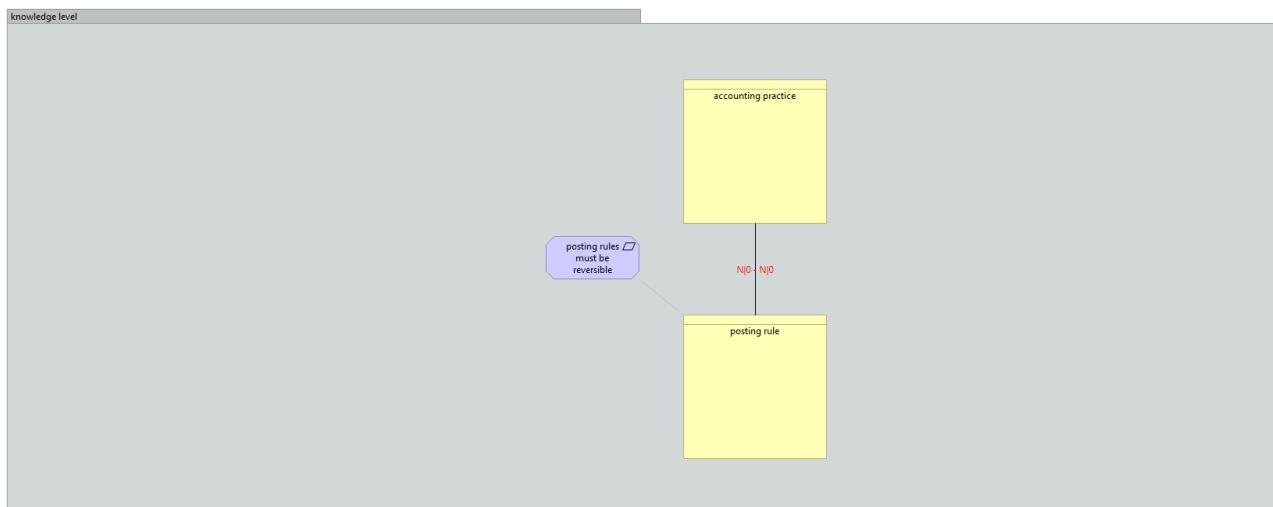
selection of entries for the application of the posing rules:
variant 1
The account returns all the entries, and the client selects the entries it needs.



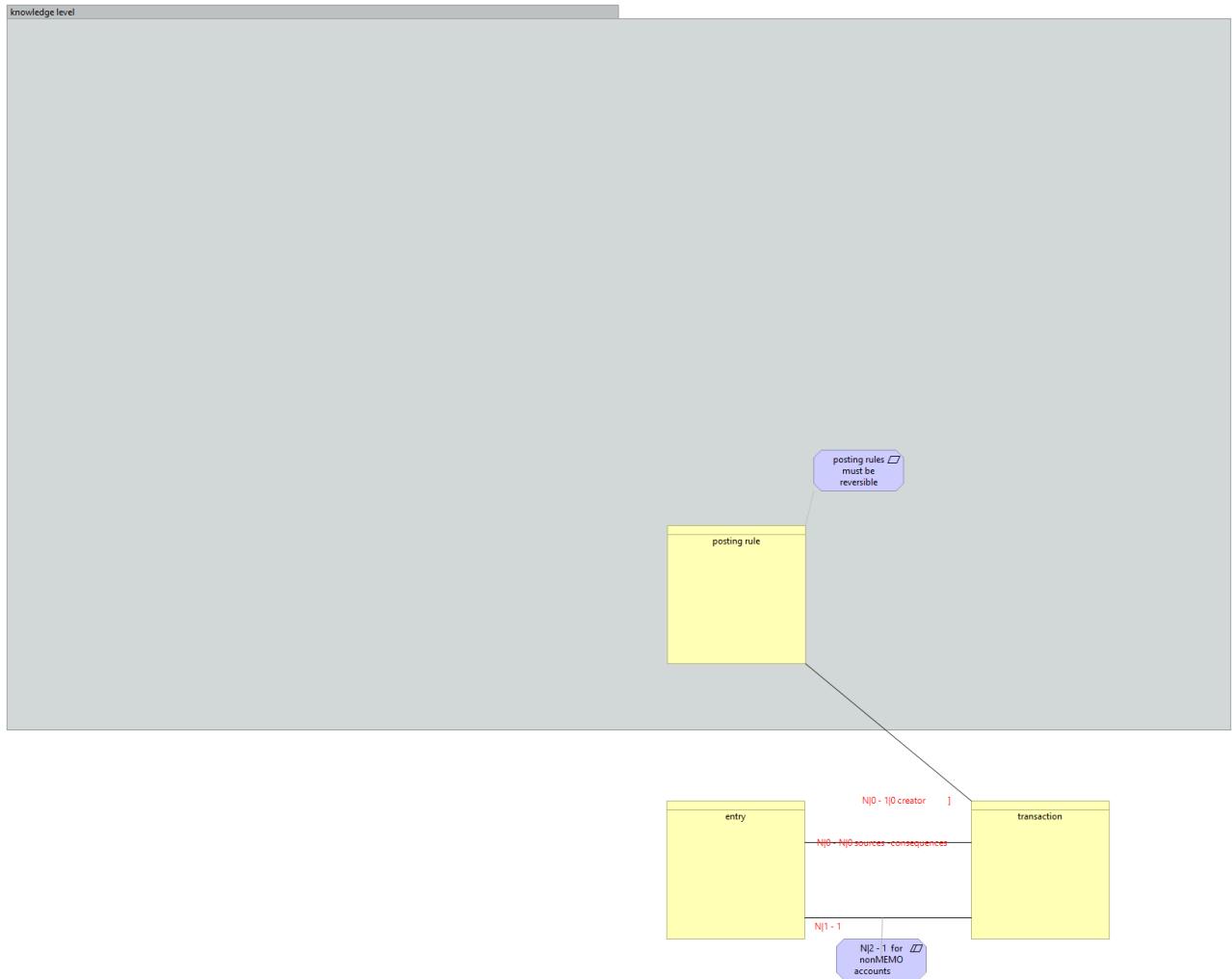
knowledge level



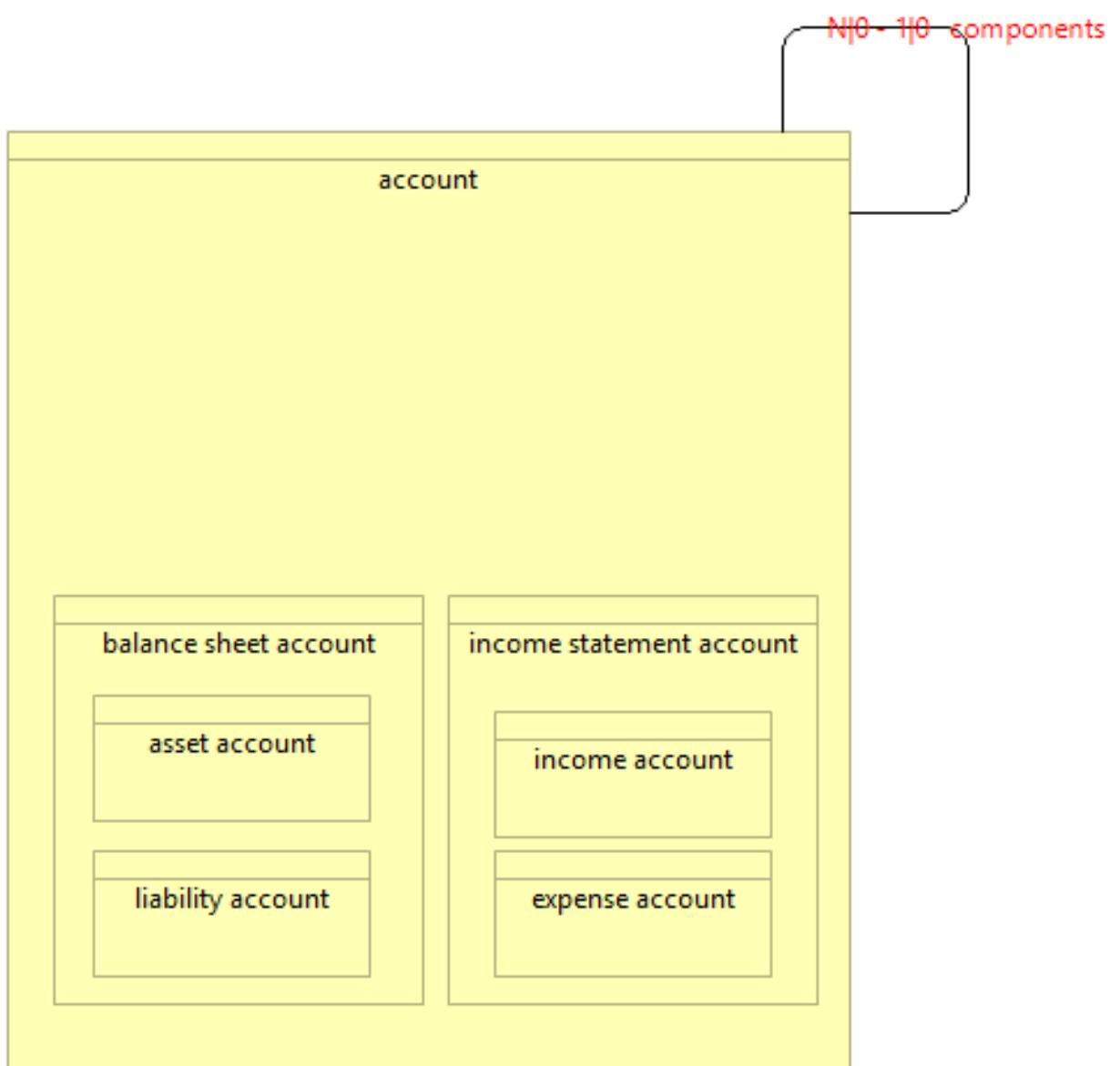
ACCOUNTING PRACTICE



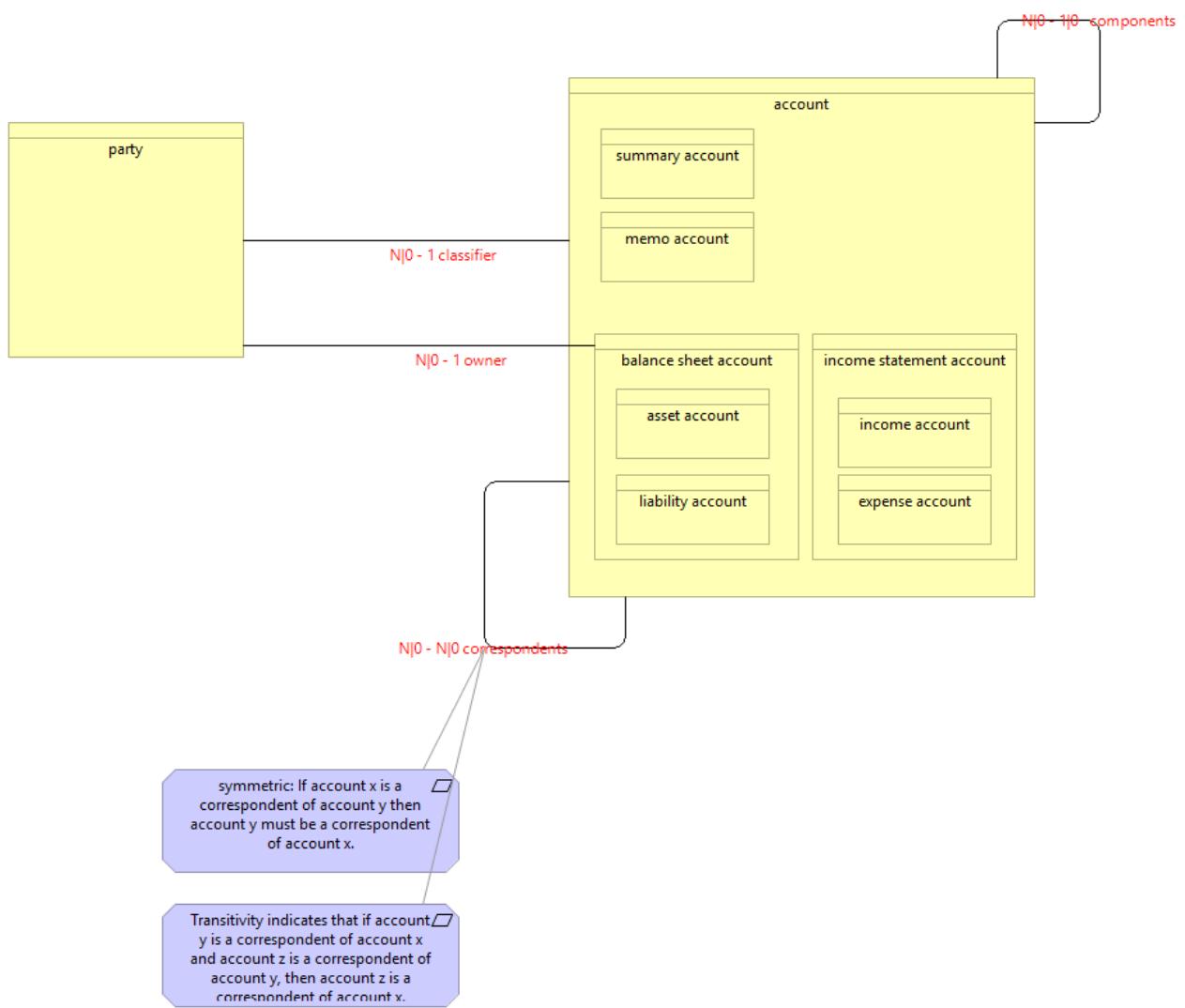
SOURCES OF AN ENTRY



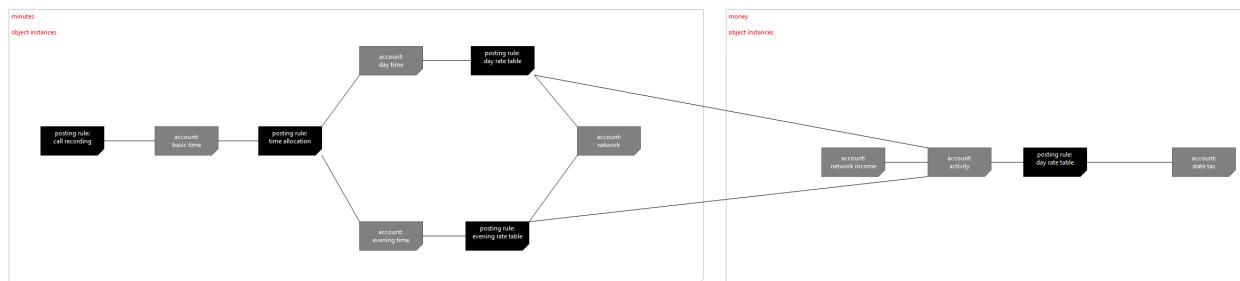
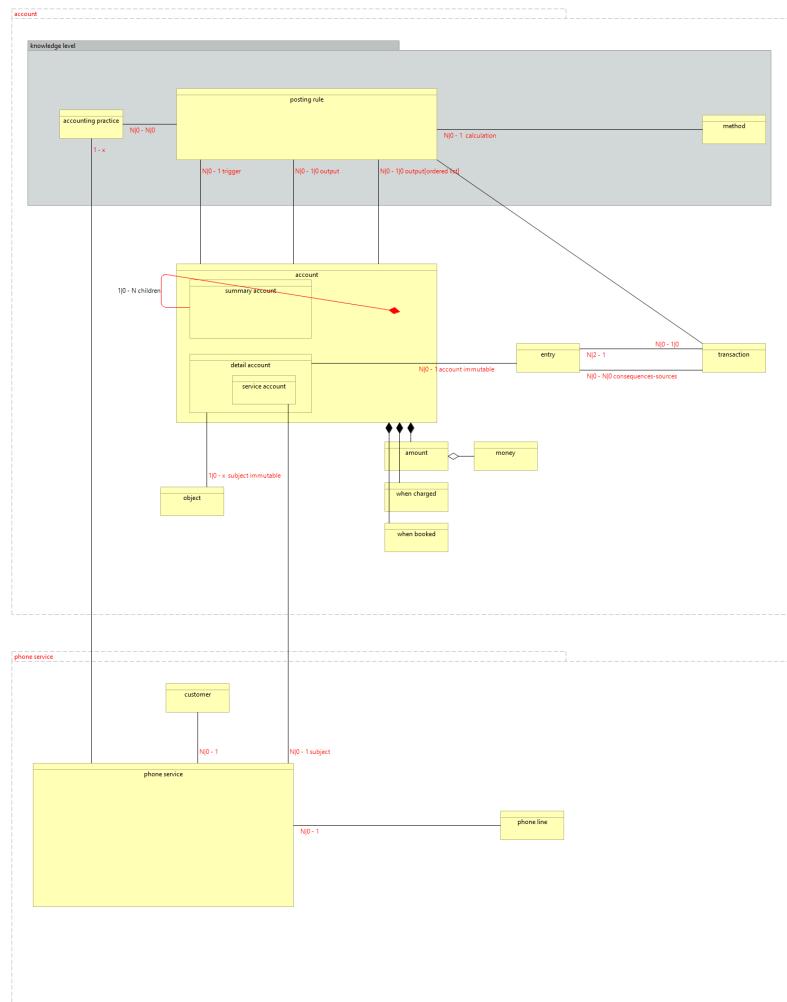
BALANCE SHEET AND INCOME STATEMENT



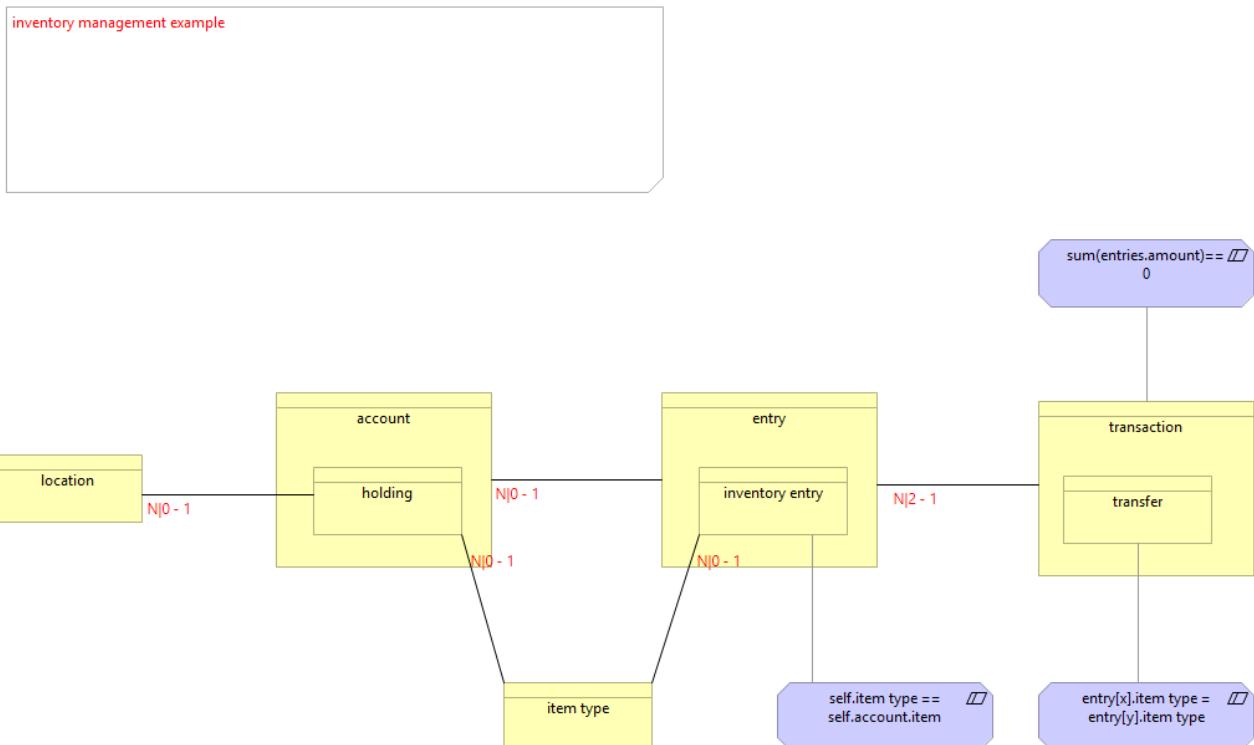
CORRESPONDING ACCOUNT



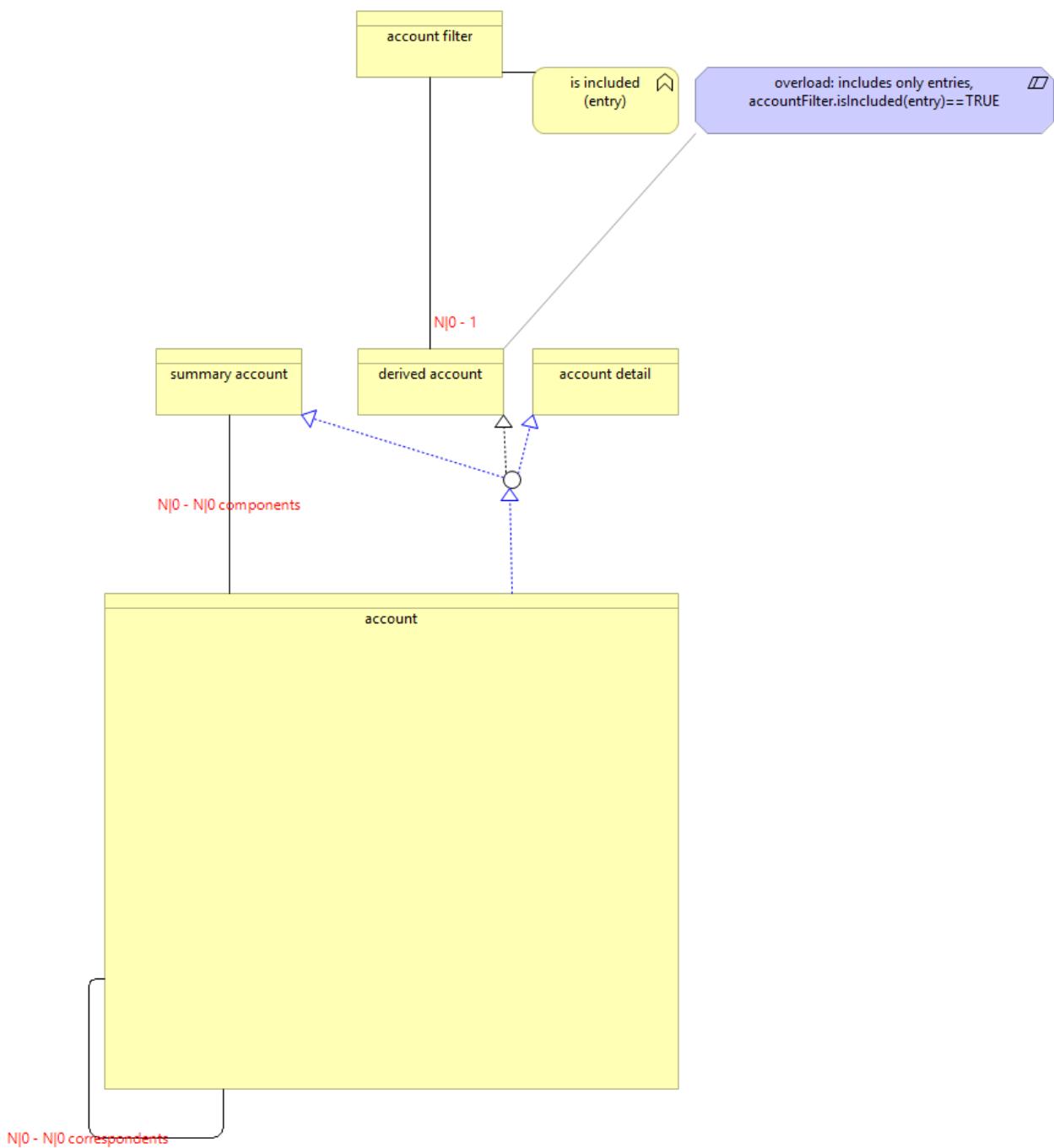
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)



SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)

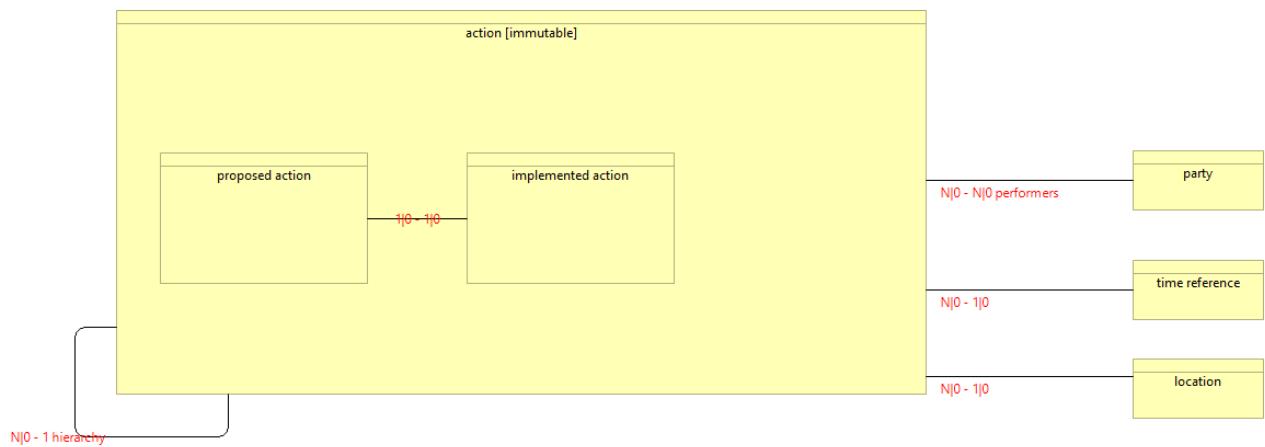


BOOKING ENTRIES TO MULTIPLE ACCOUNTS

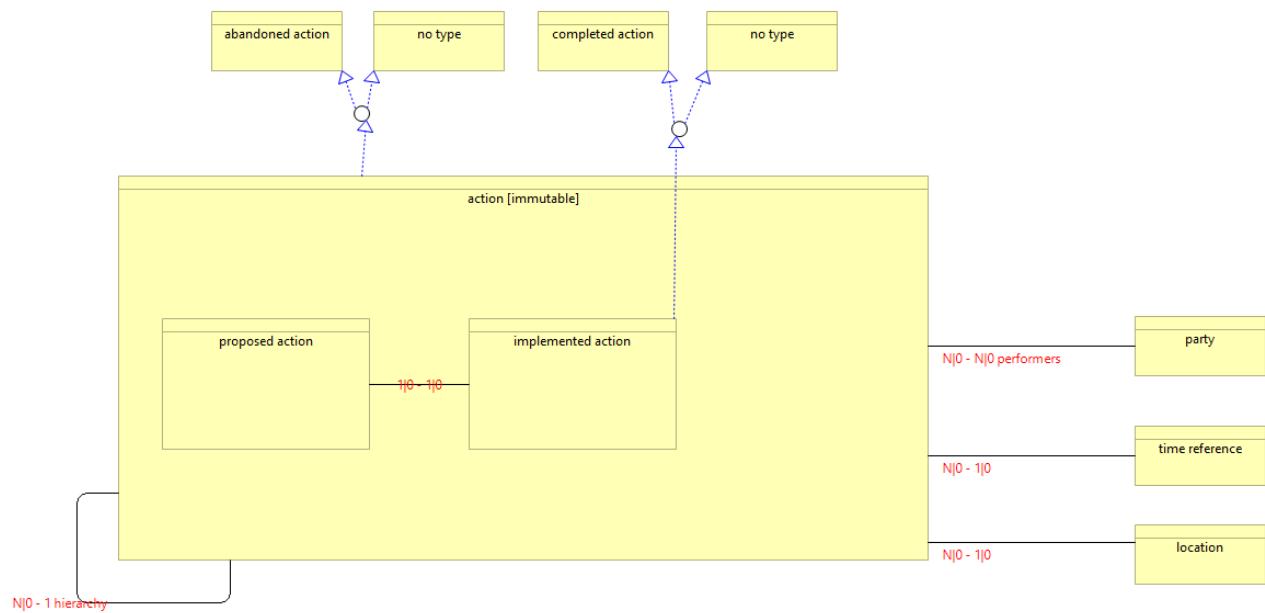


PLANNING

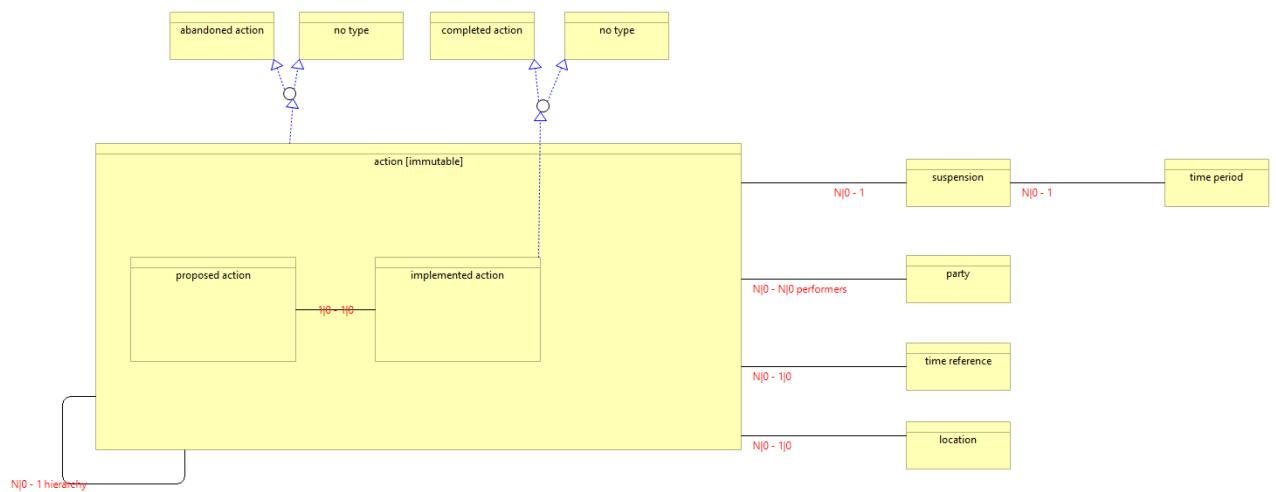
PROPOSED AND IMPLEMENTED ACTION



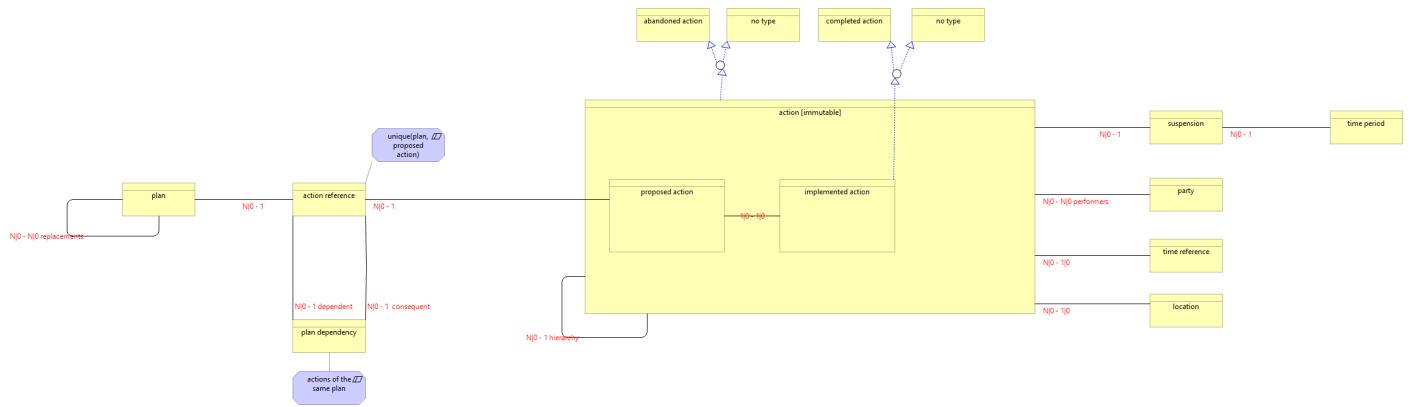
COMPLETED AND ABANDONED ACTIONS



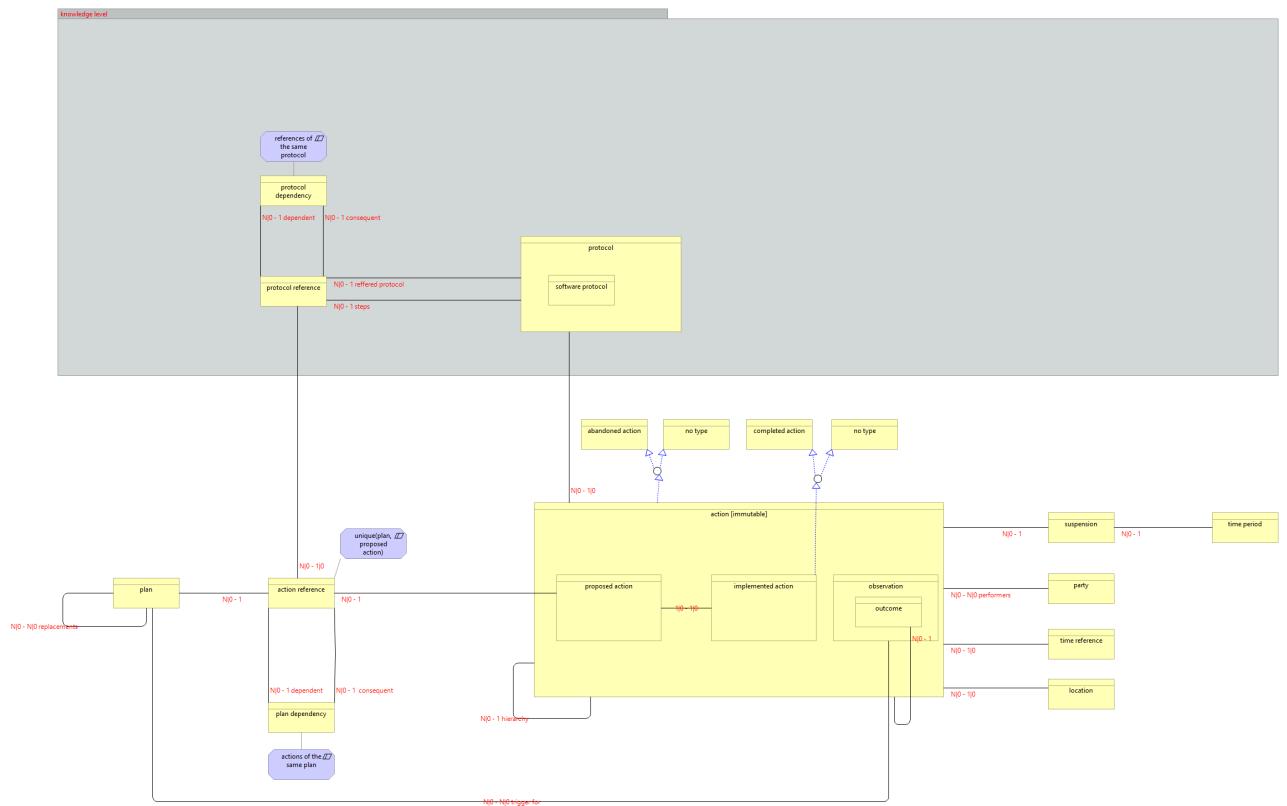
SUSPENSION



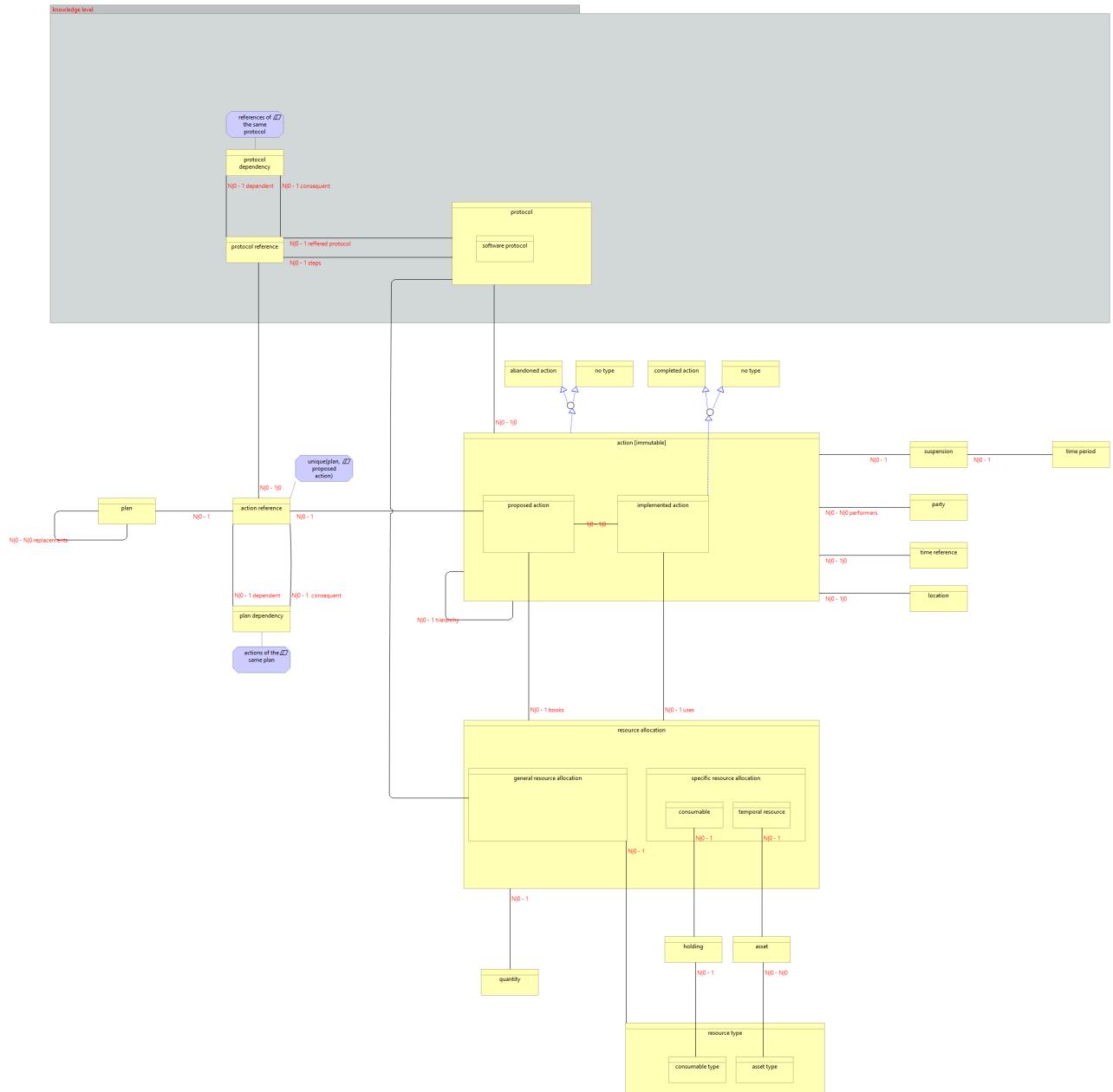
PLAN



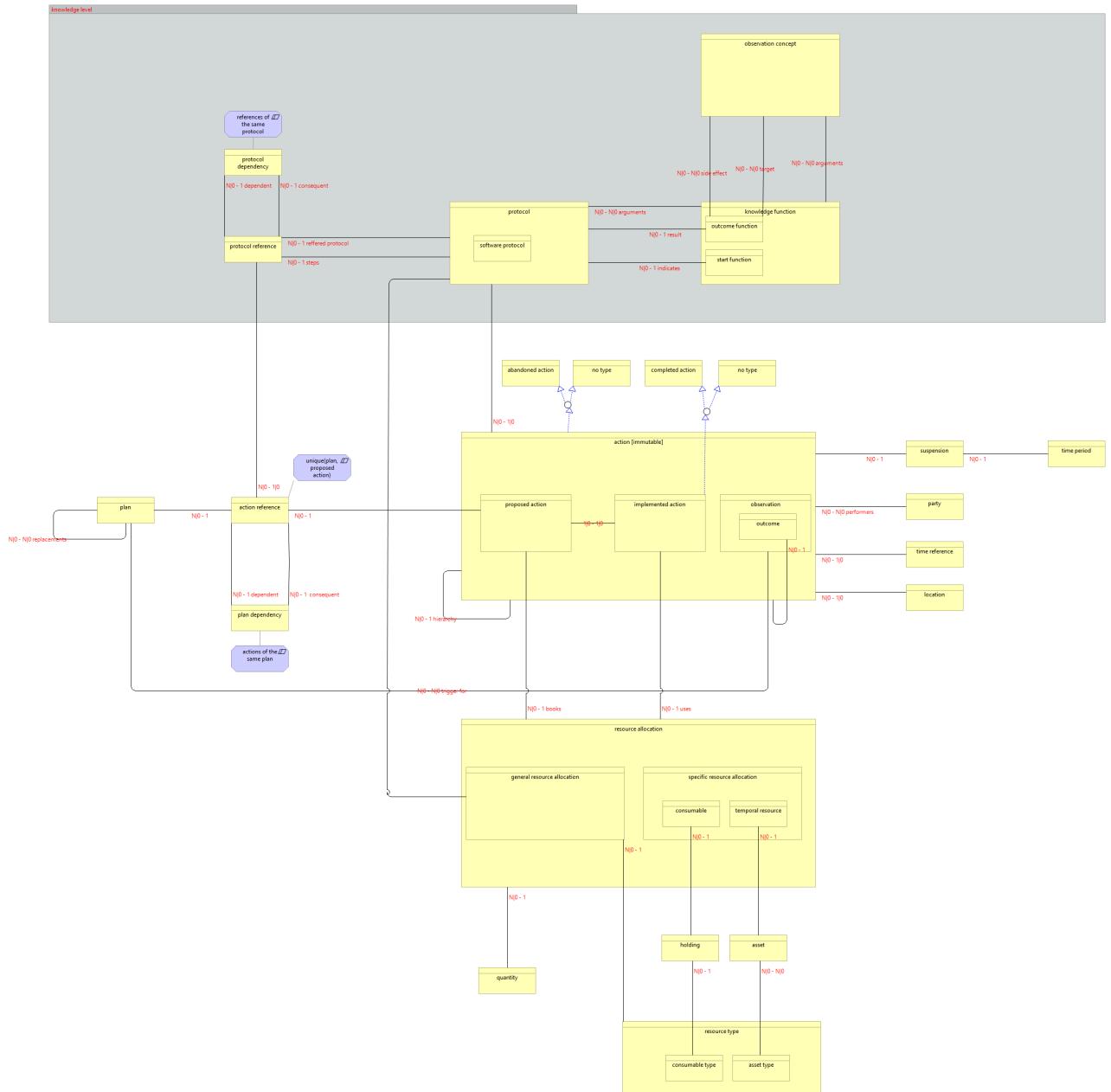
PROTOCOL



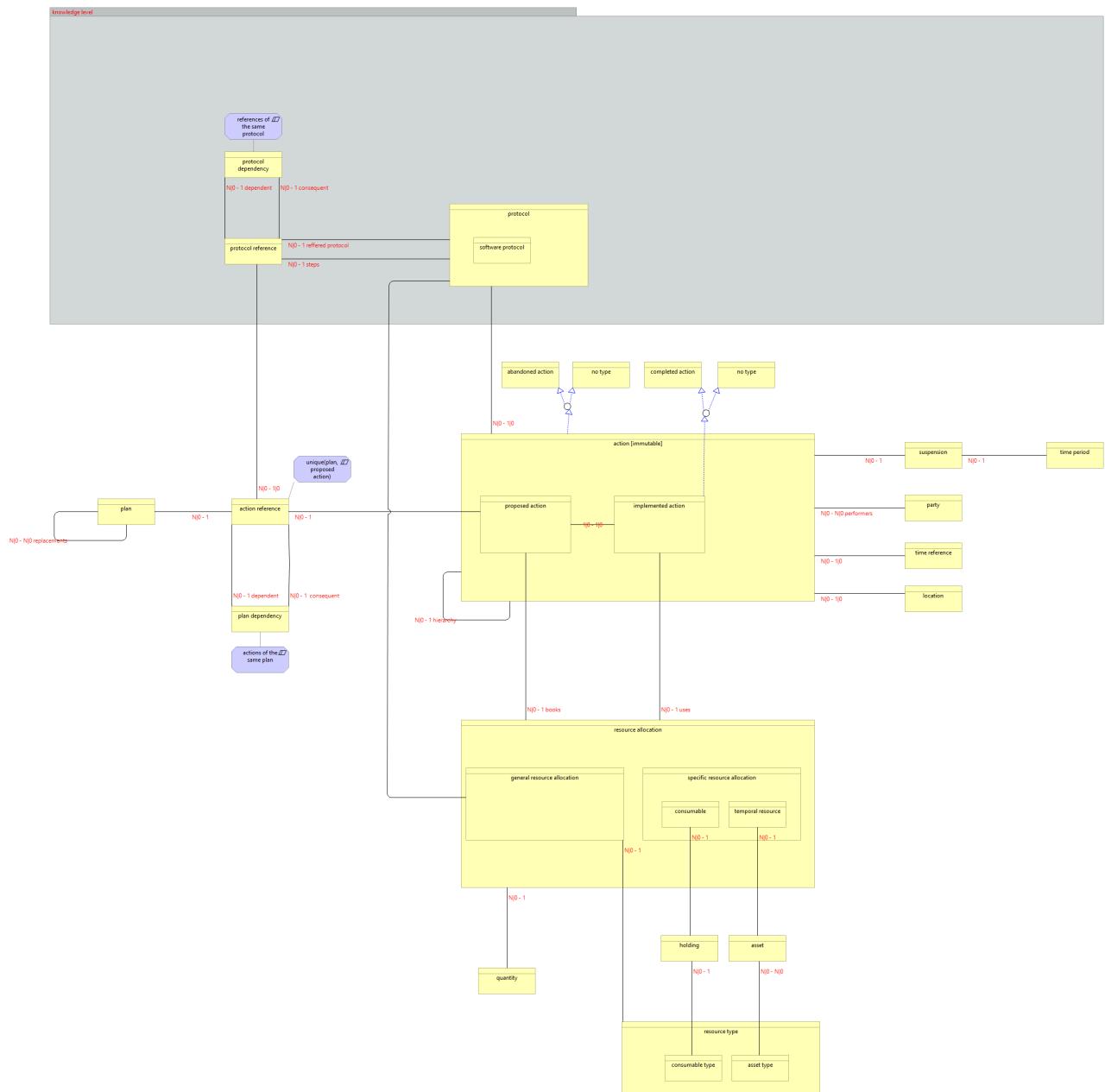
RESOURCE ALLOCATION



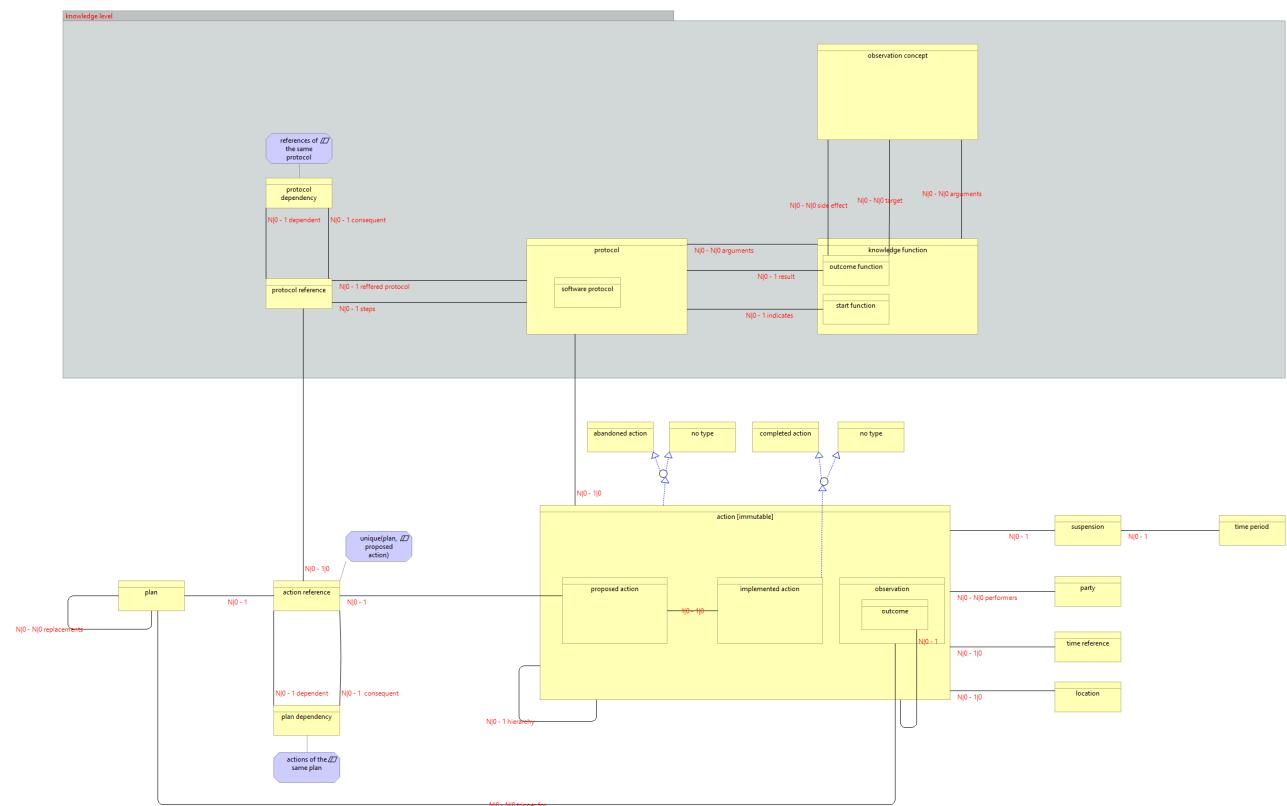
PLANNING



PLANNING (NO OUTCOME)

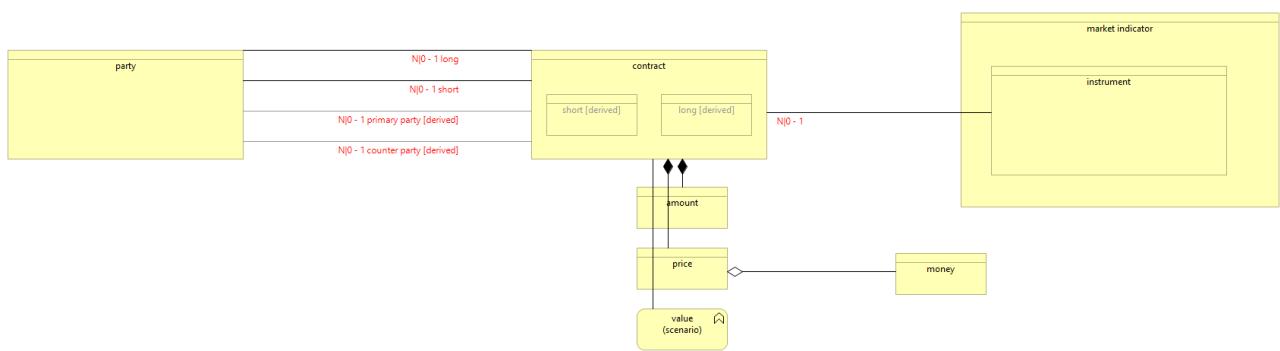


OUTCOME AND START FUNCTIONS

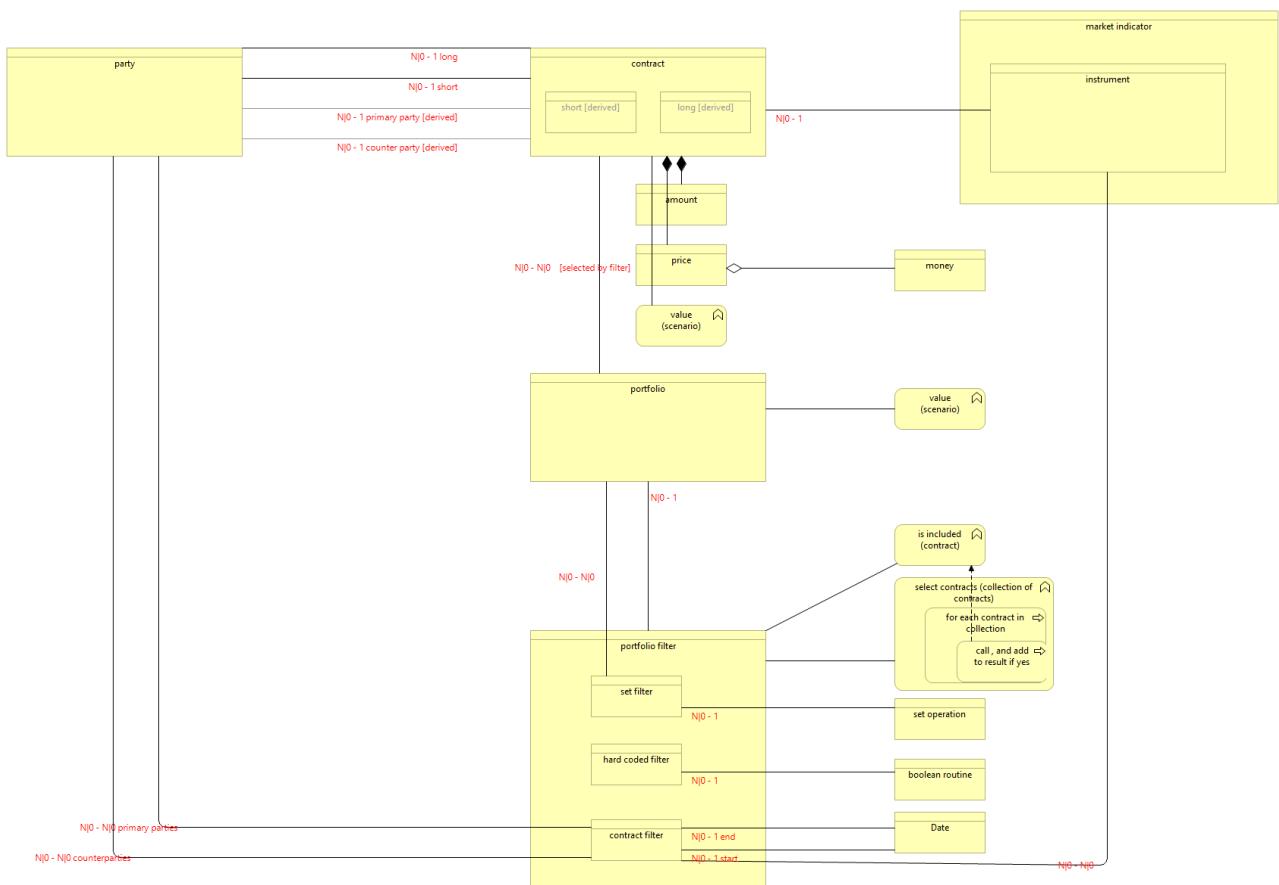


TRADING

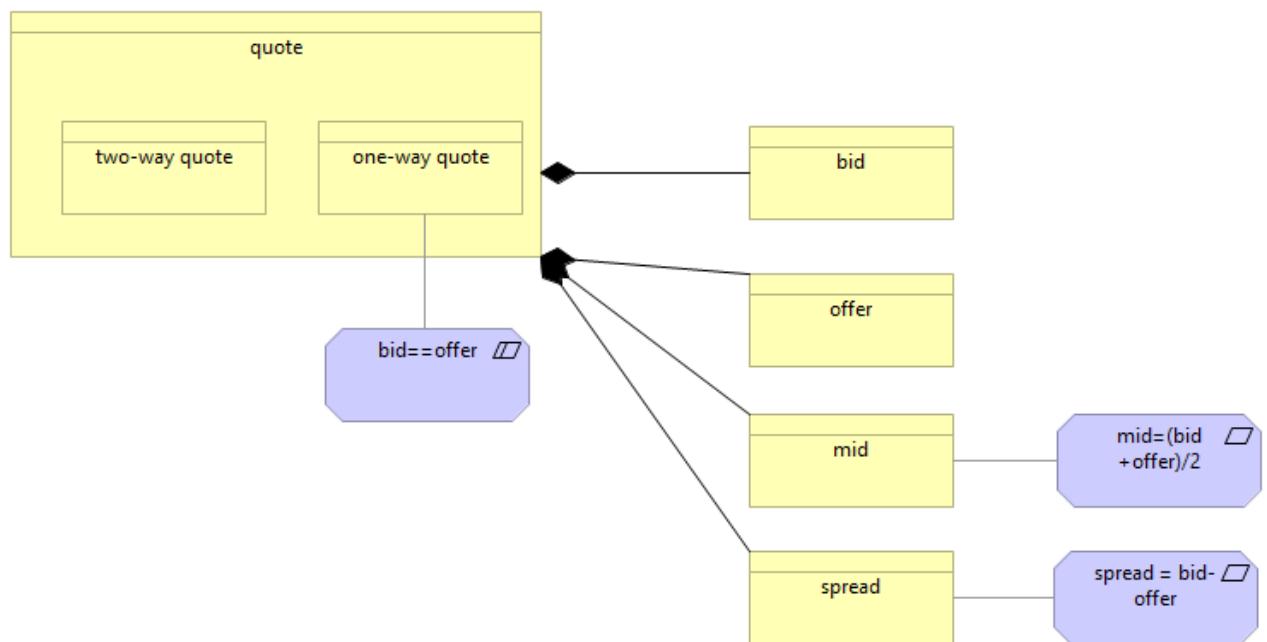
CONTRACT



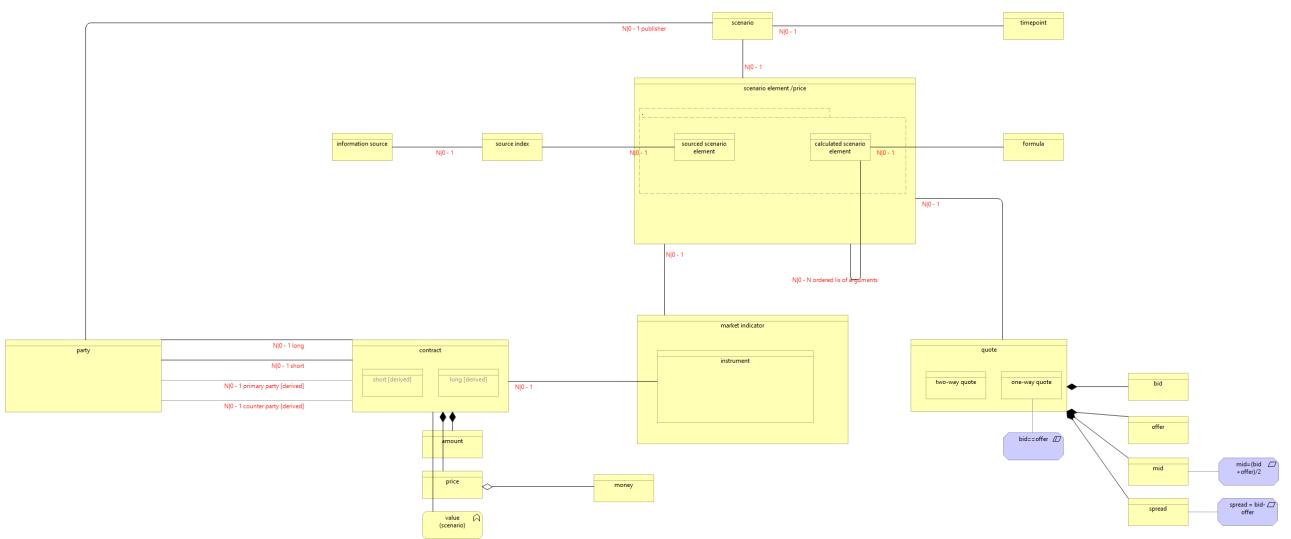
PORTFOLIO



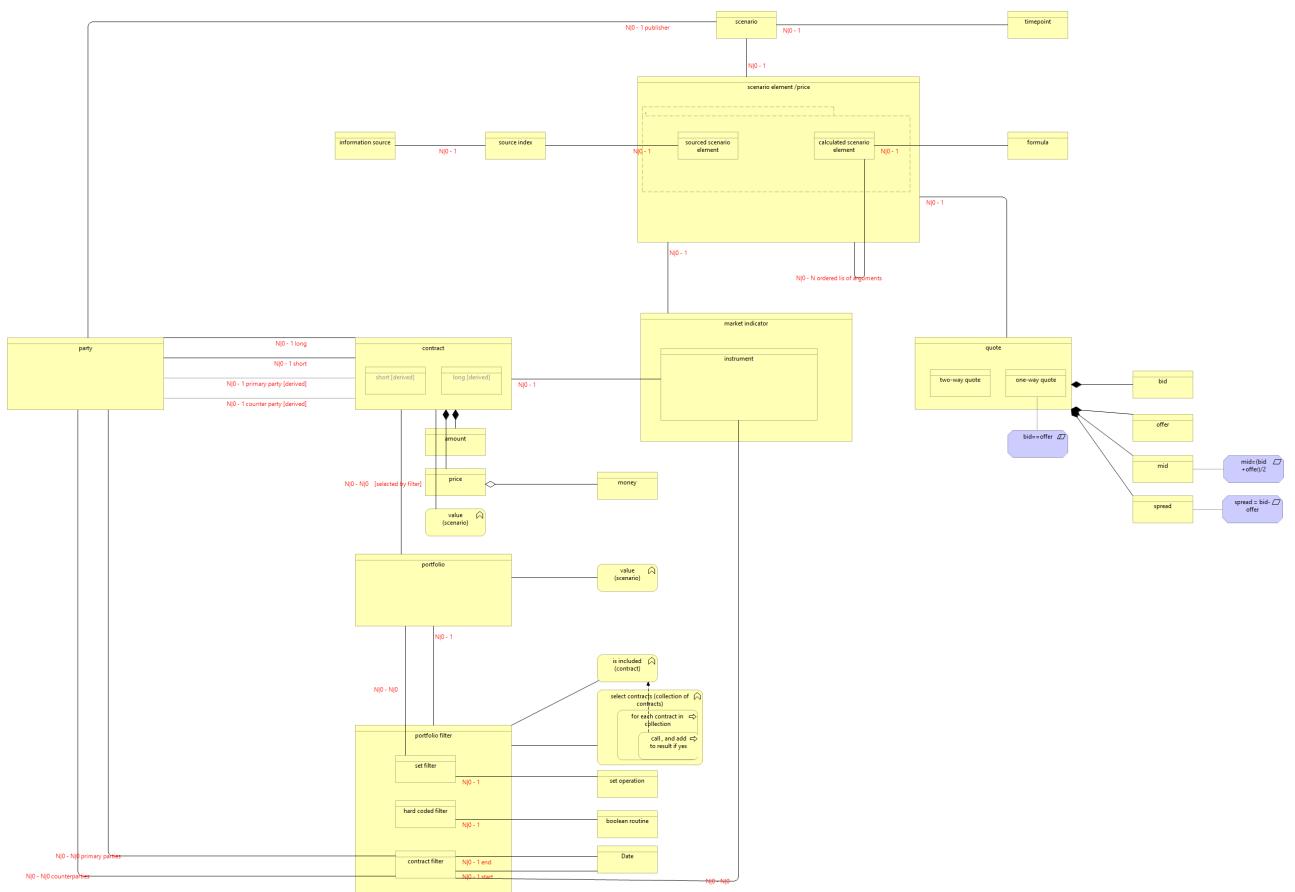
QUOTE



SCENARIO



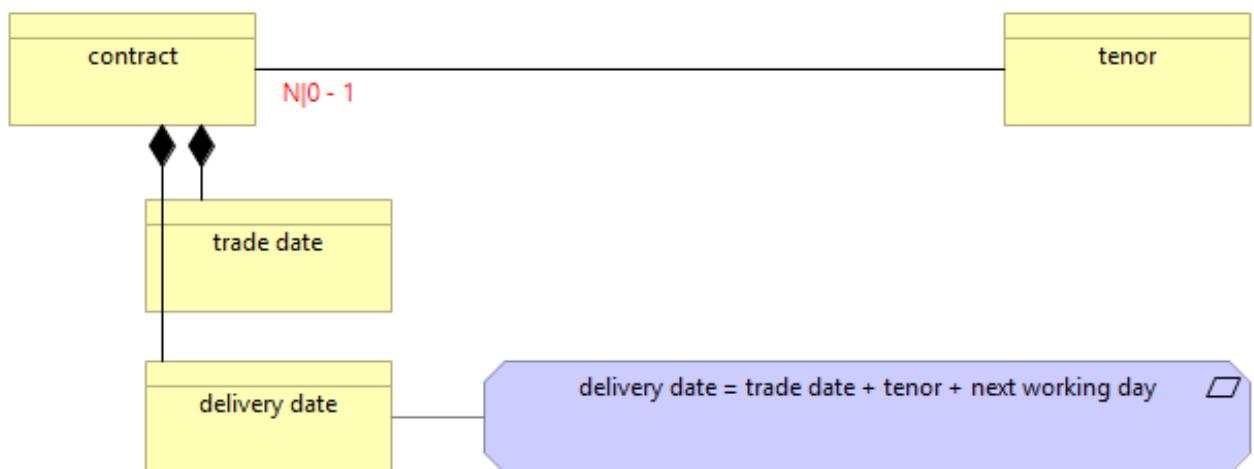
TRADING



DERIVATIVE CONTRACTS

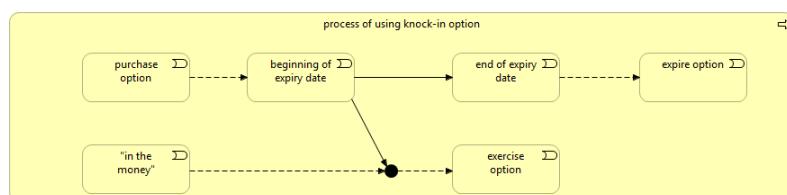
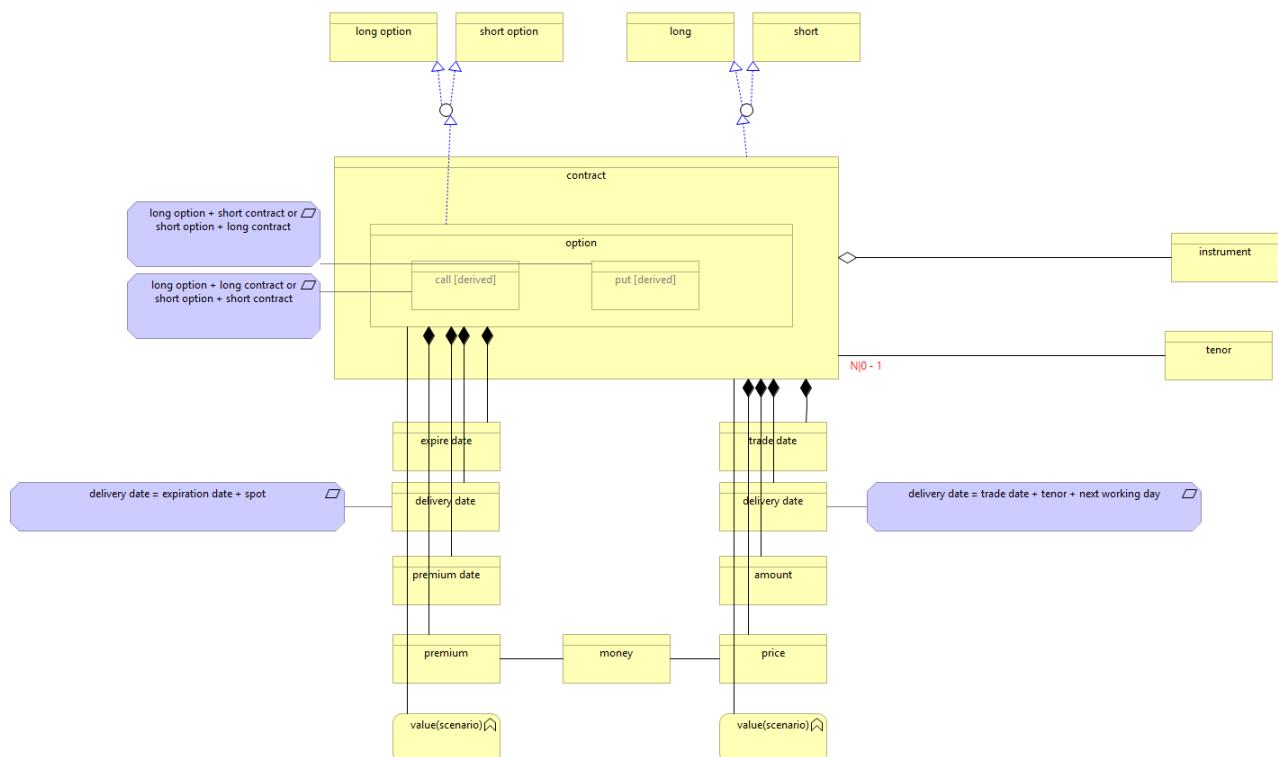
FORWARD CONTRACTS

- forward contracts
- Delivery usually occurs in a couple of months



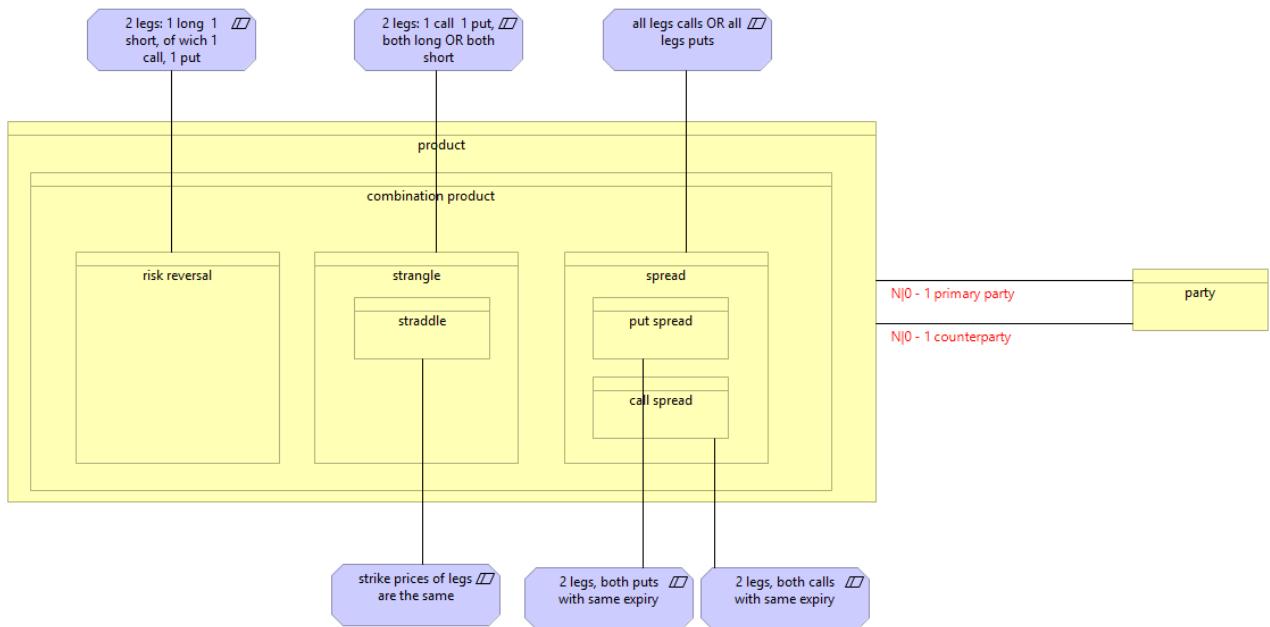
OPTIONS

An option gives the buyer the right to buy dollars at a prearranged exchange rate if the holder wishes



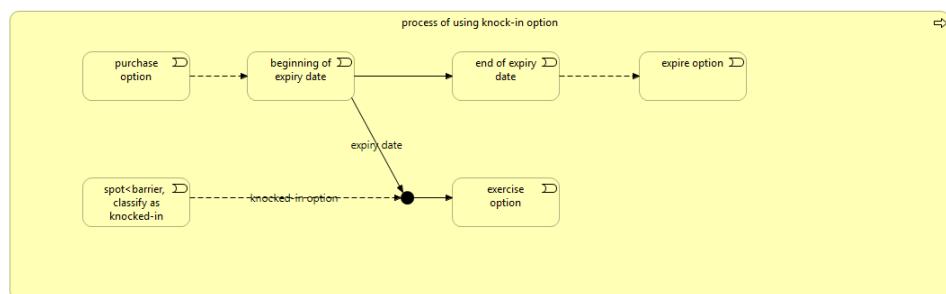
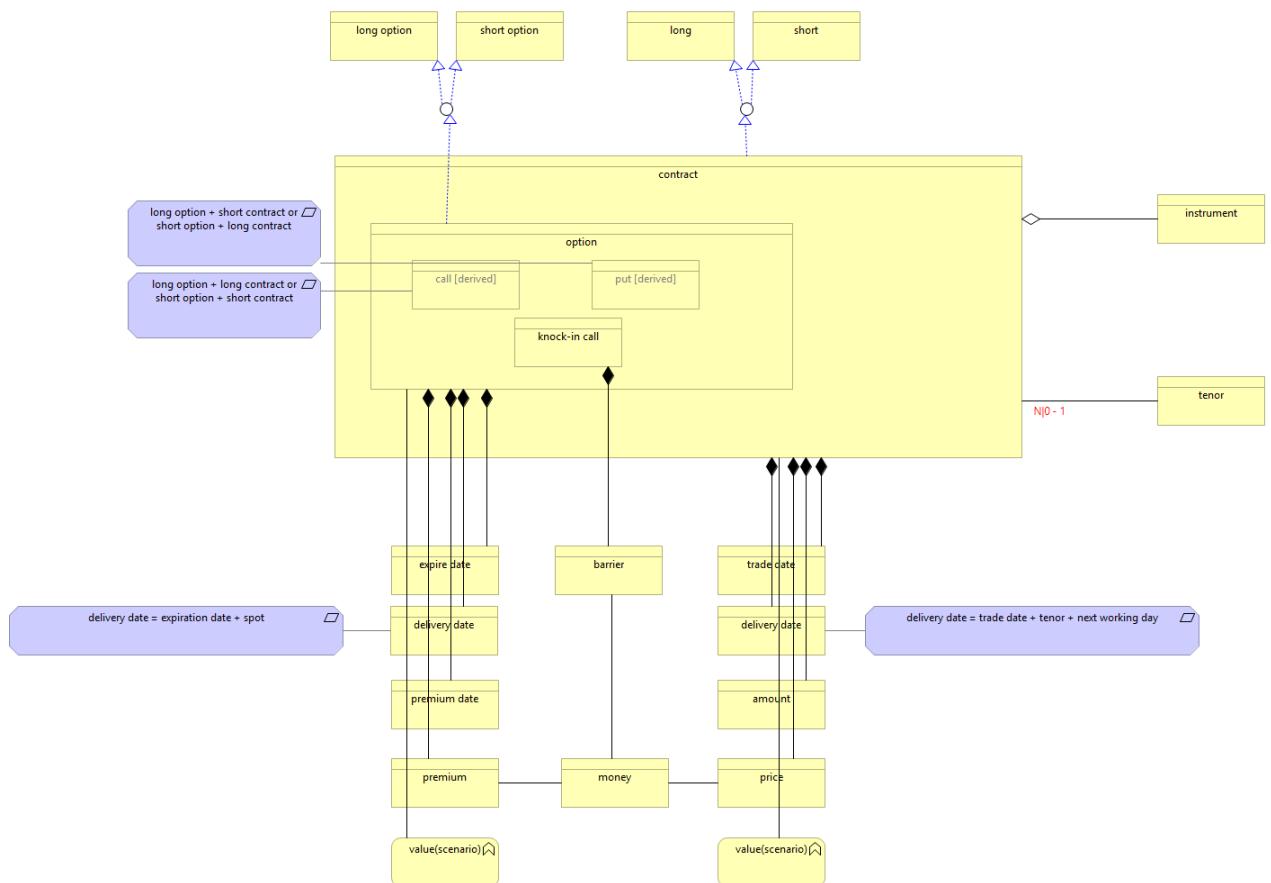
PRODUCT

- combination options can be seen as a composite of other options.



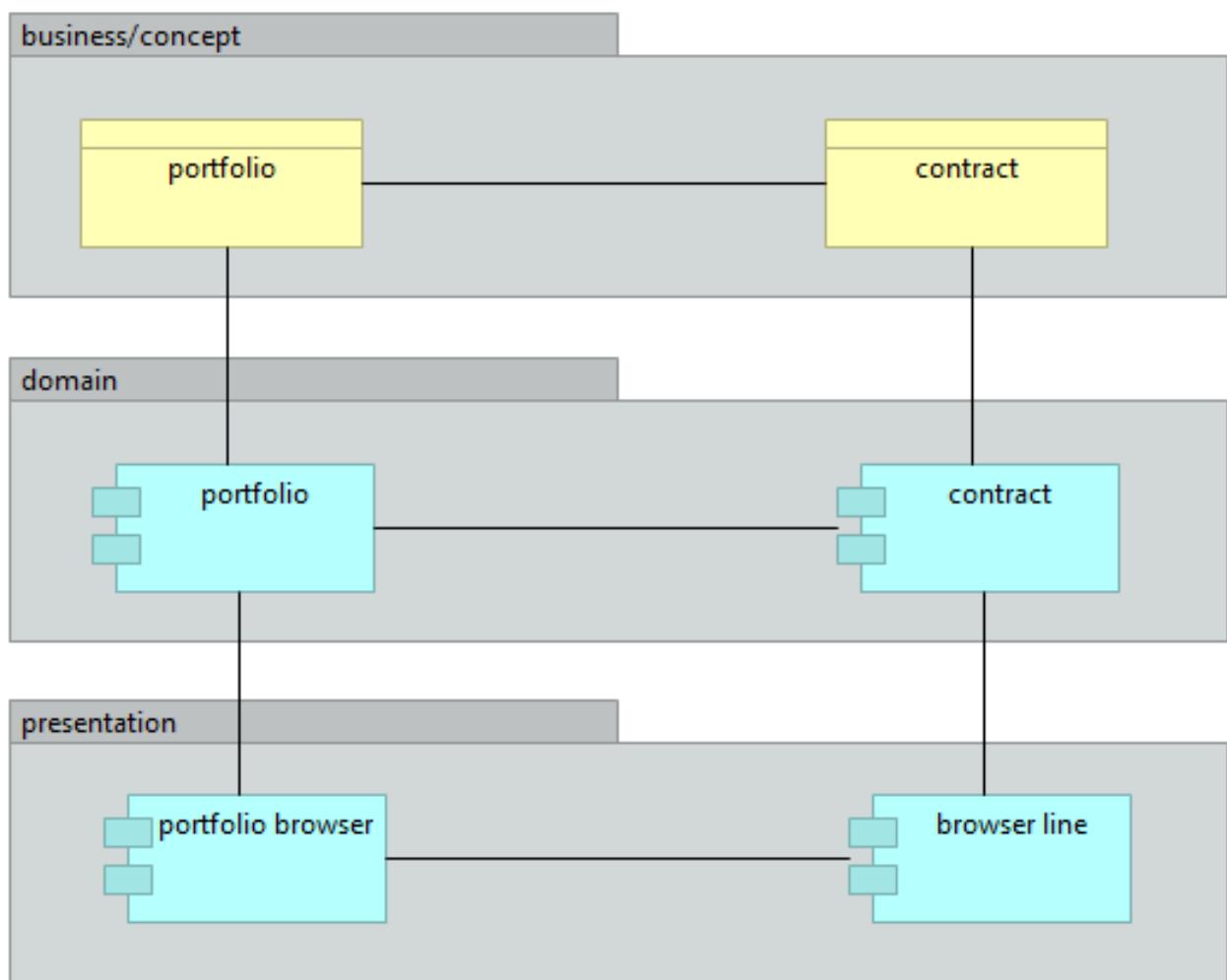
SUBTYPE STATE MACHINES

A barrier option can either appear or disappear when the price of the instrument, as quoted on some agreed market pricing (such as a Reuters page), reaches a particular limit.

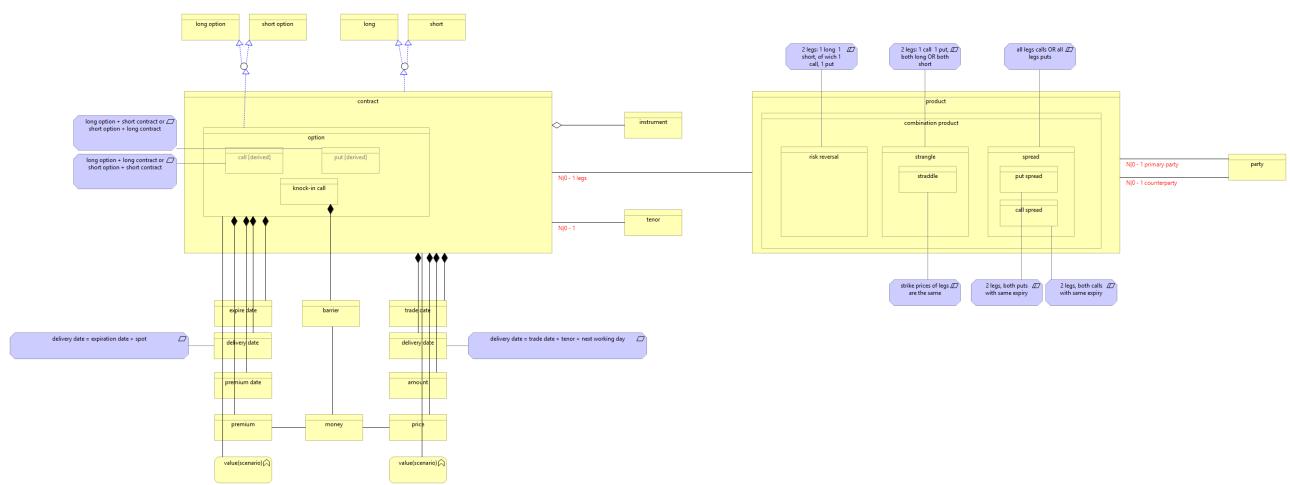


PARALLEL APPLICATION AND DOMAIN HIERARCHIES

example, a report containing a
list of contracts



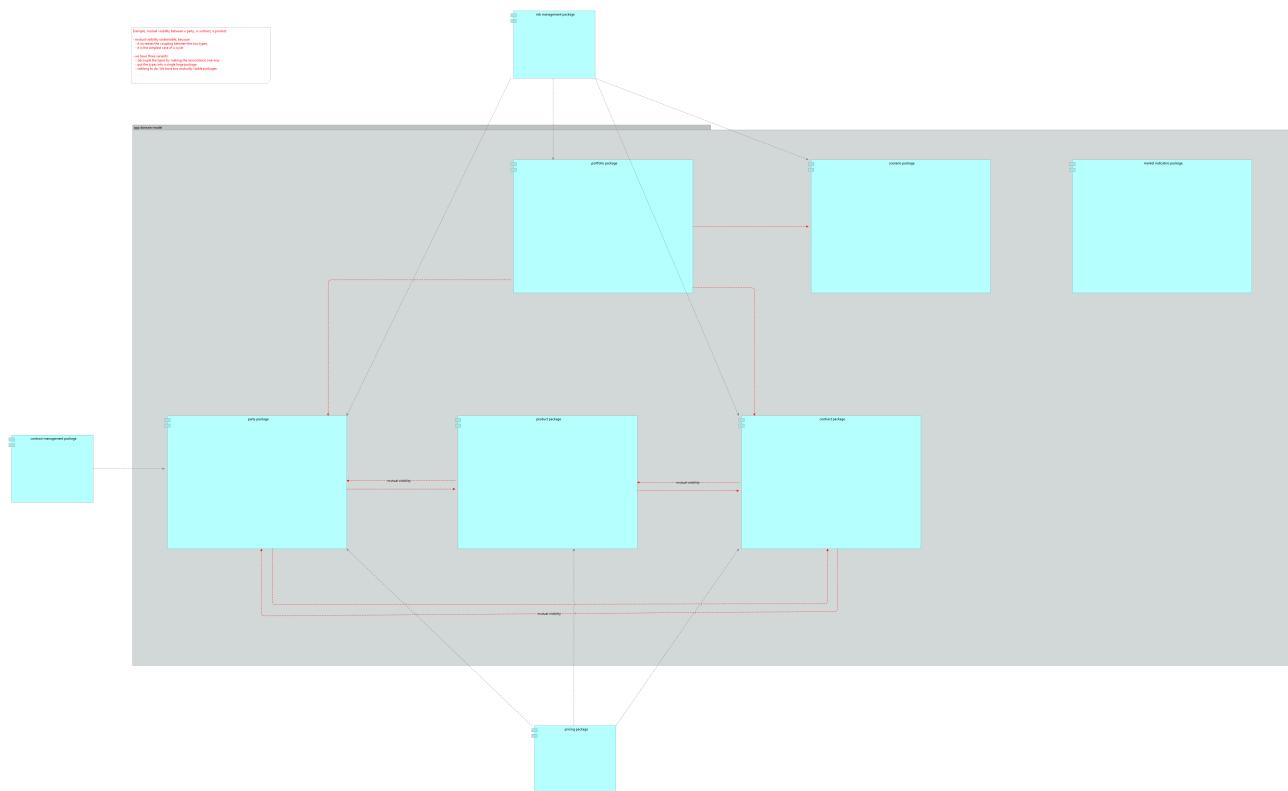
DERIVATIVE CONTRACTS



TRADING PACKAGES

MULTIPLE ACCESS LEVELS TO A PACKAGE

MUTUAL VISIBILITY

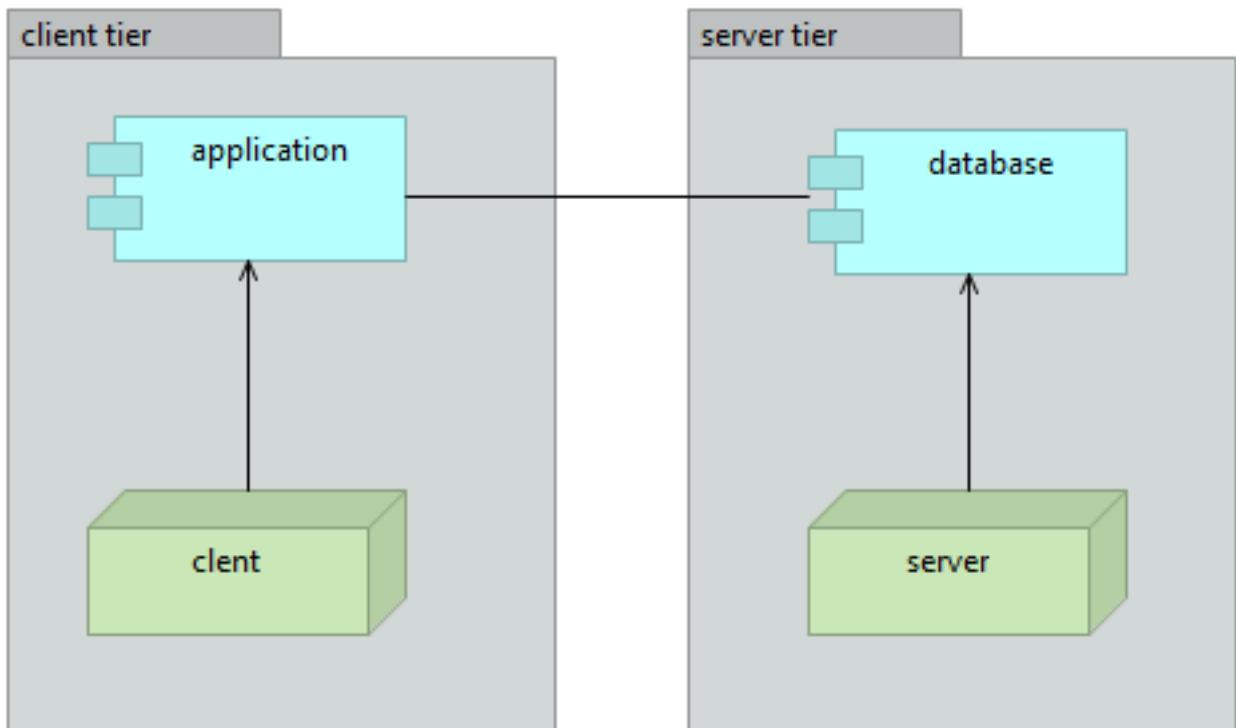


TRADING PACKAGES

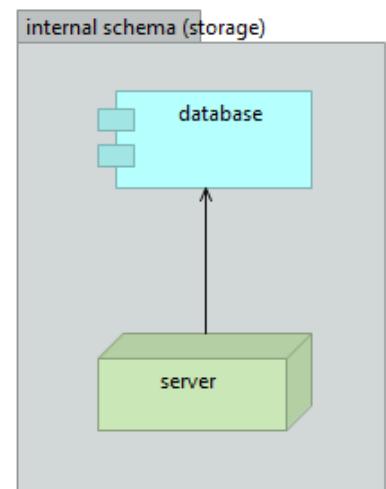
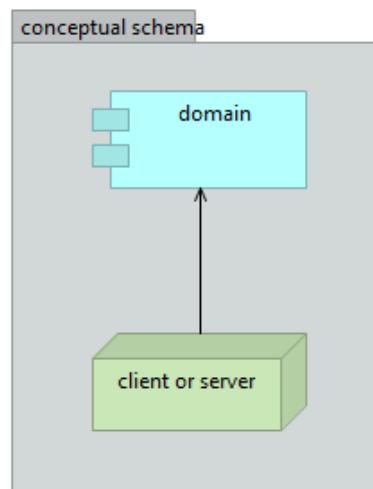
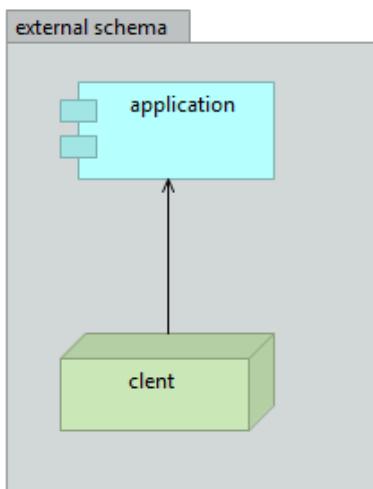
LAYERED ARCHITECTURE

TWO-TIER ARCHITECTURE

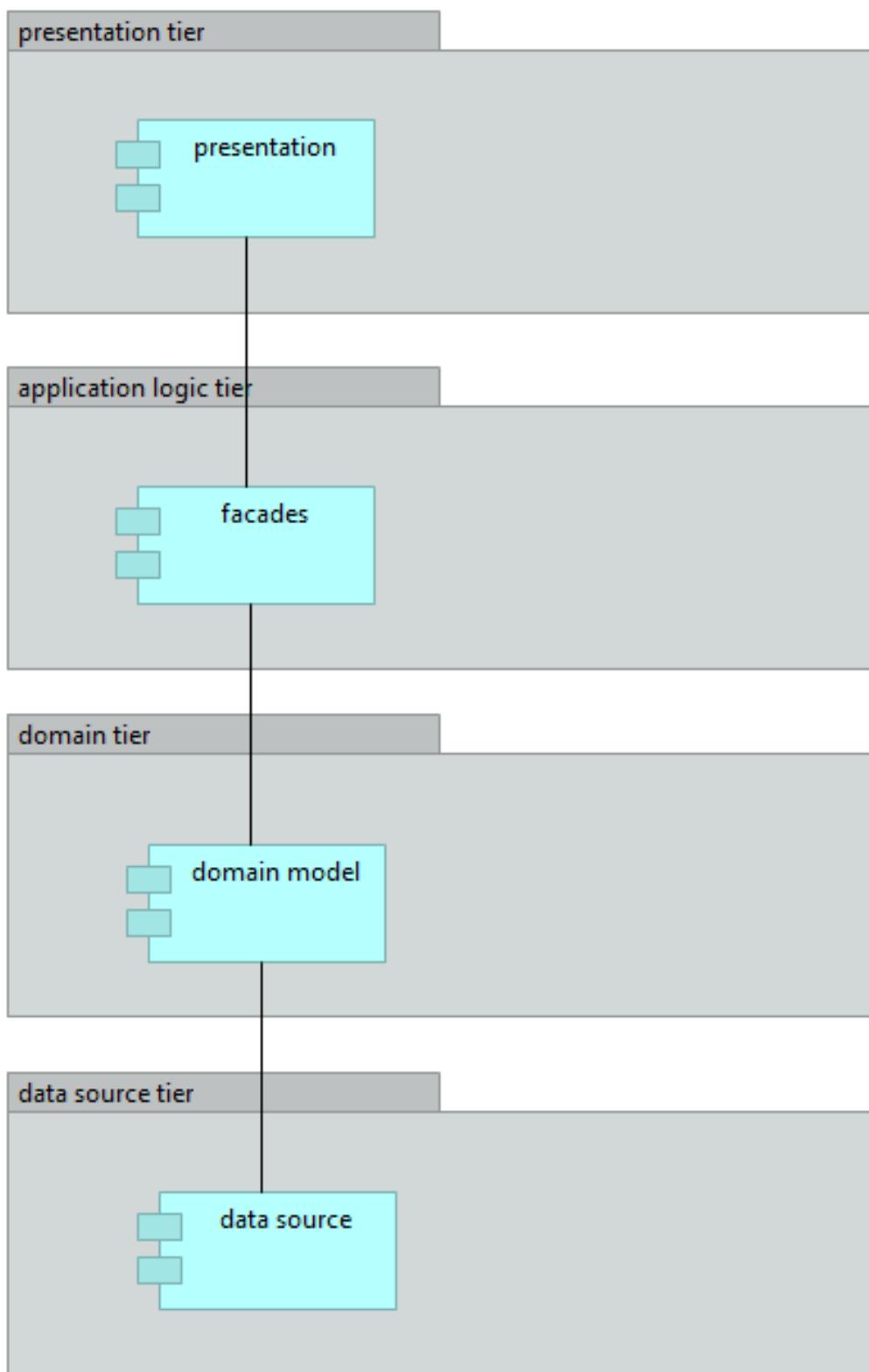
two-tier architecture



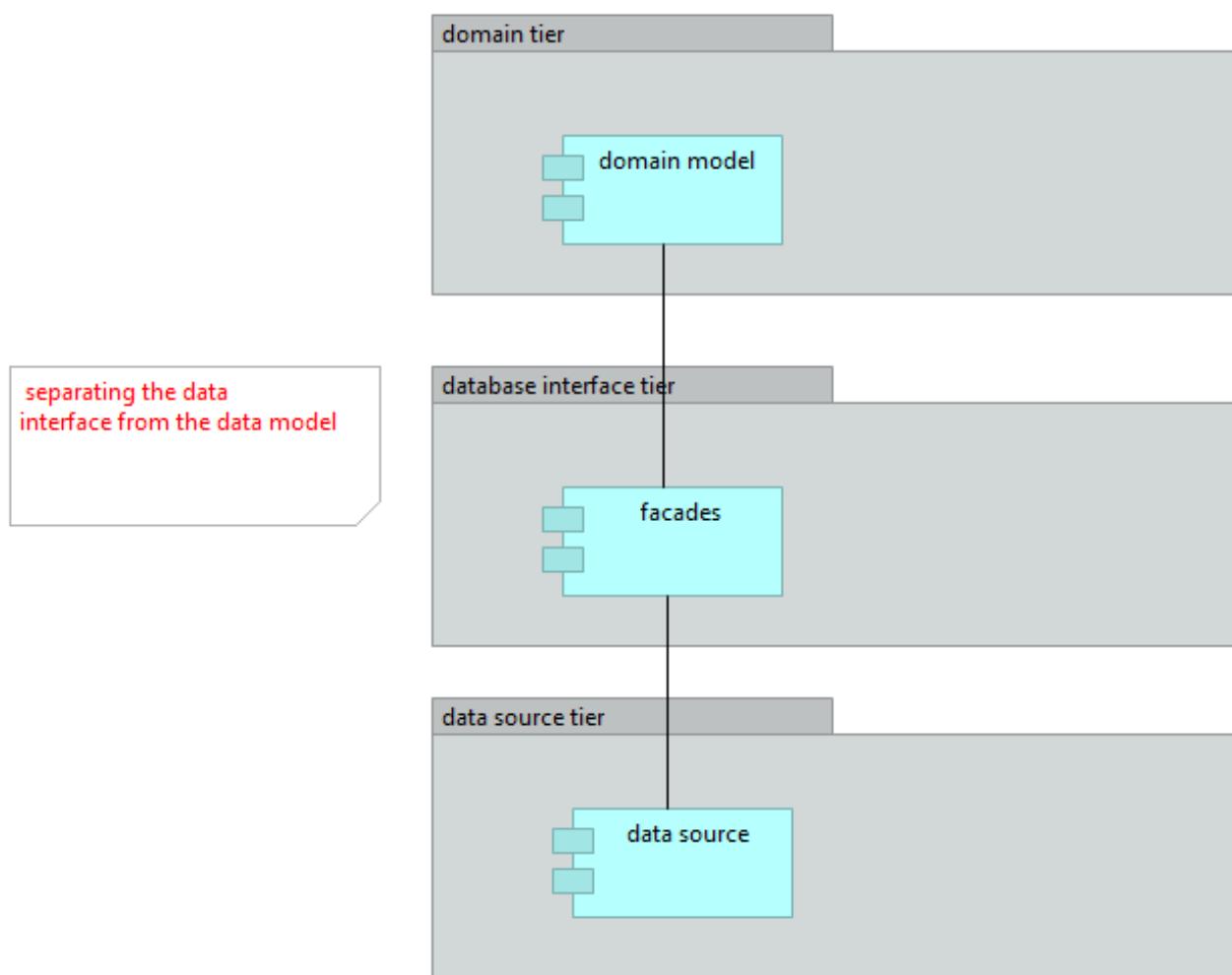
THREE-TIER ARCHITECTURE



PRESENTATION AND APPLICATION LOGIC

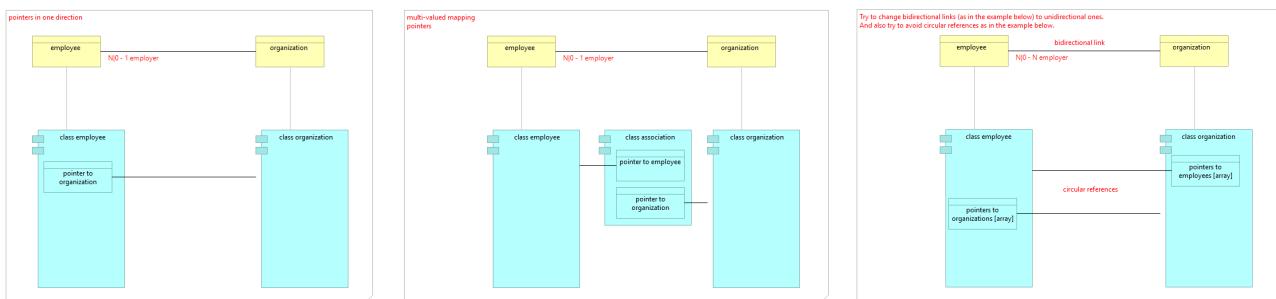
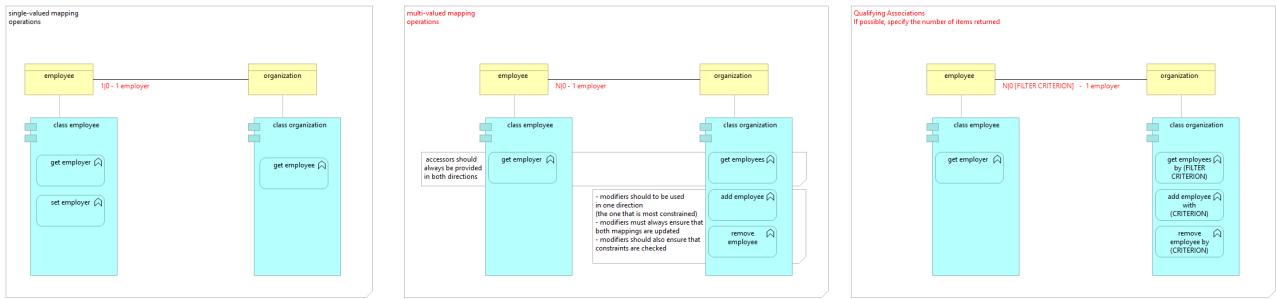


DATABASE INTERACTION



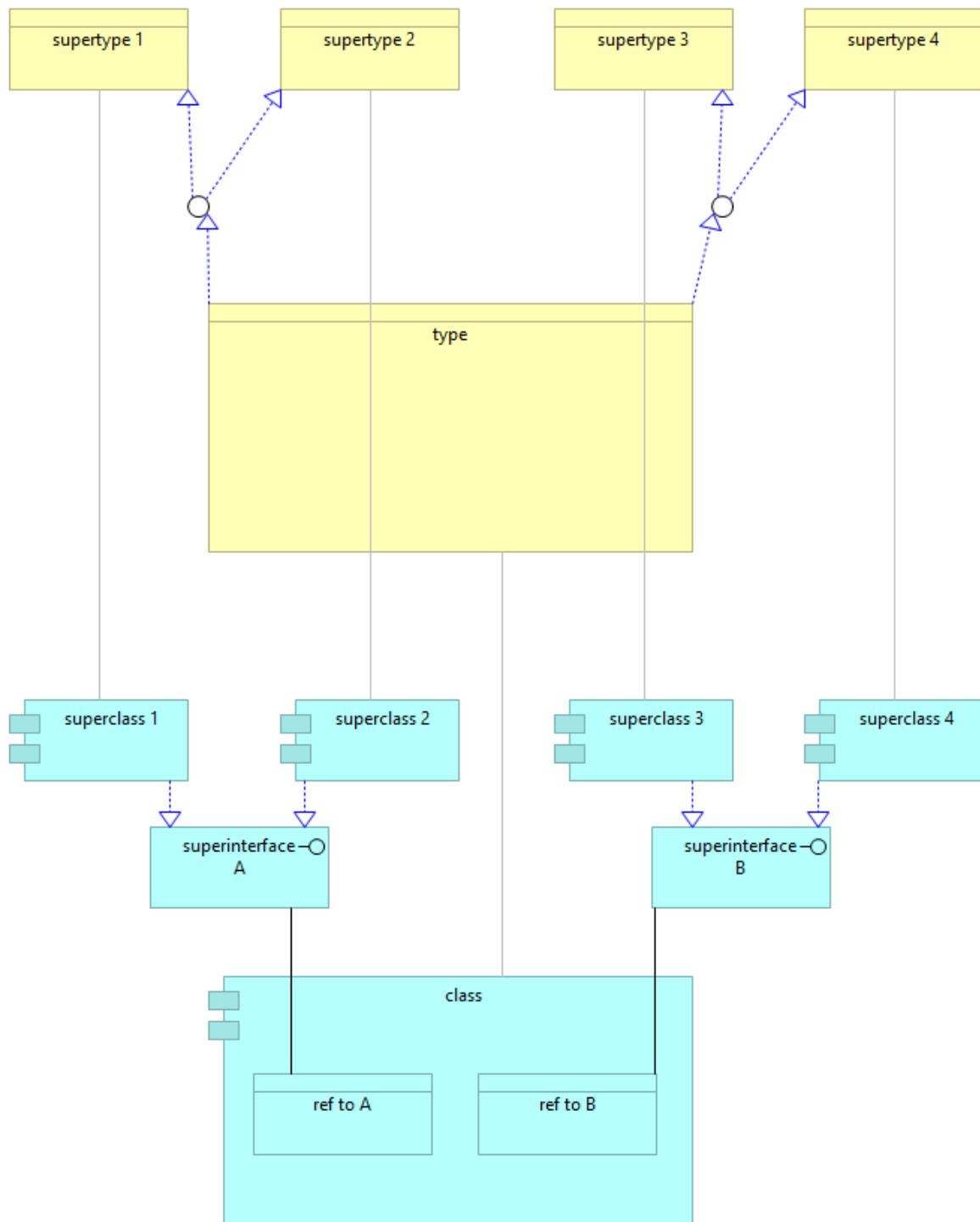
TYPE MODEL DESIGN

IMPLEMENTING ASSOCIATIONS

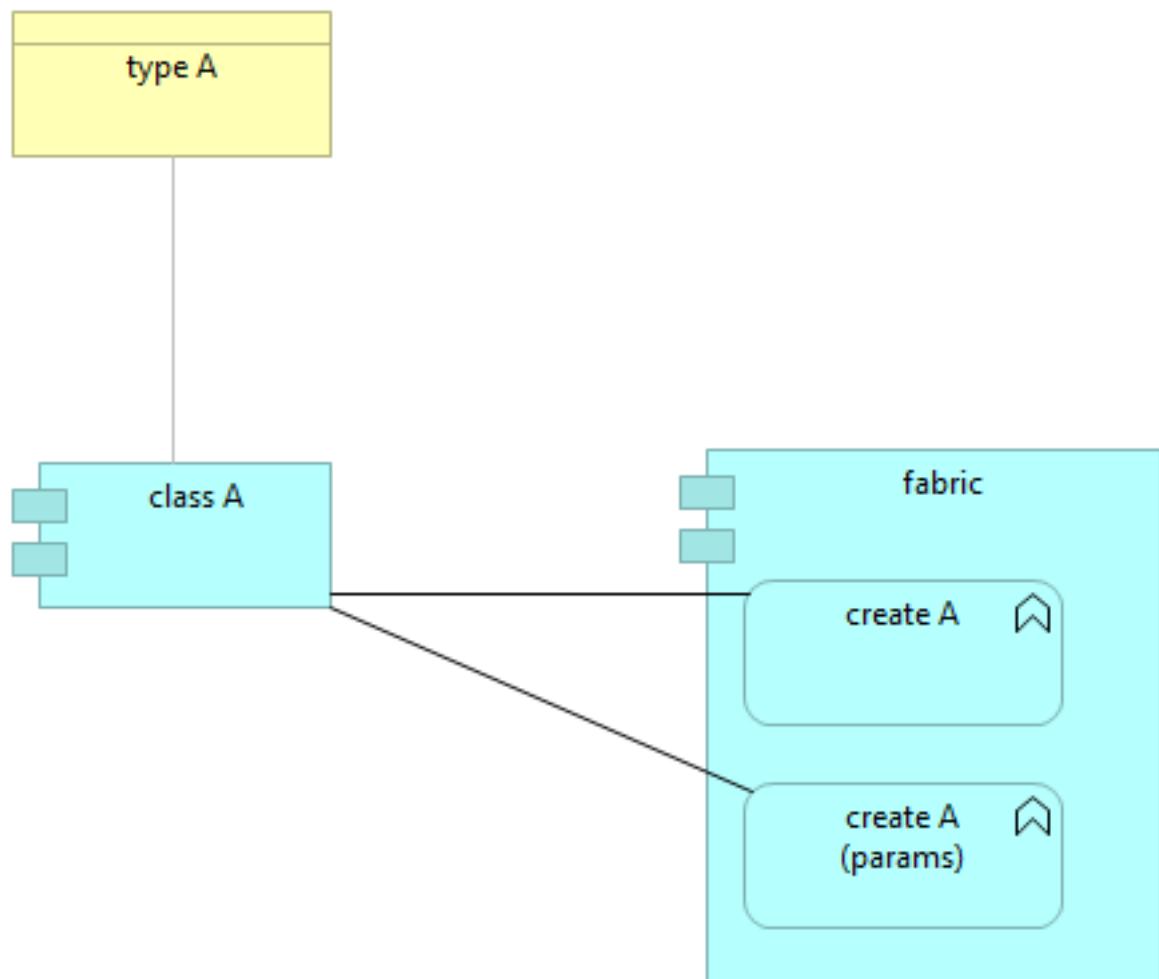


IMPLEMENTING GENERALIZATION

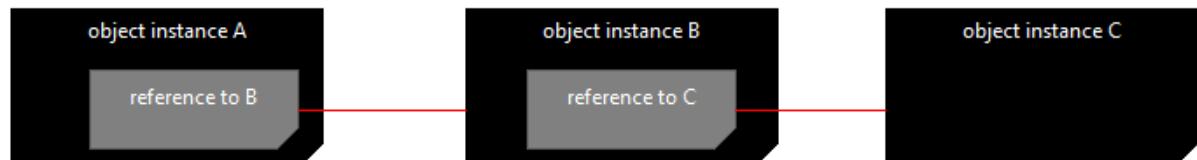
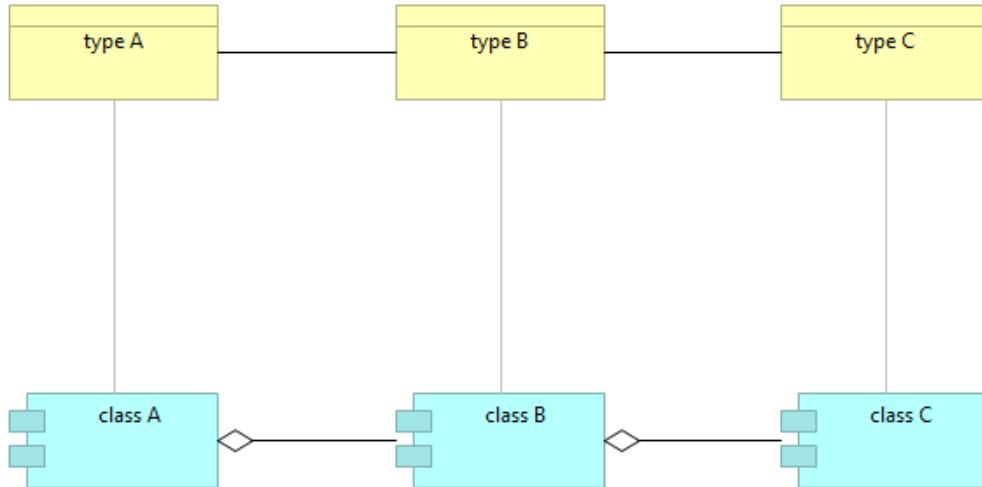
- example of implementation of multiple inheritance
- for example, the composition is applicable instead of inheritance



OBJECT CREATION



OBJECT DESTRUCTION

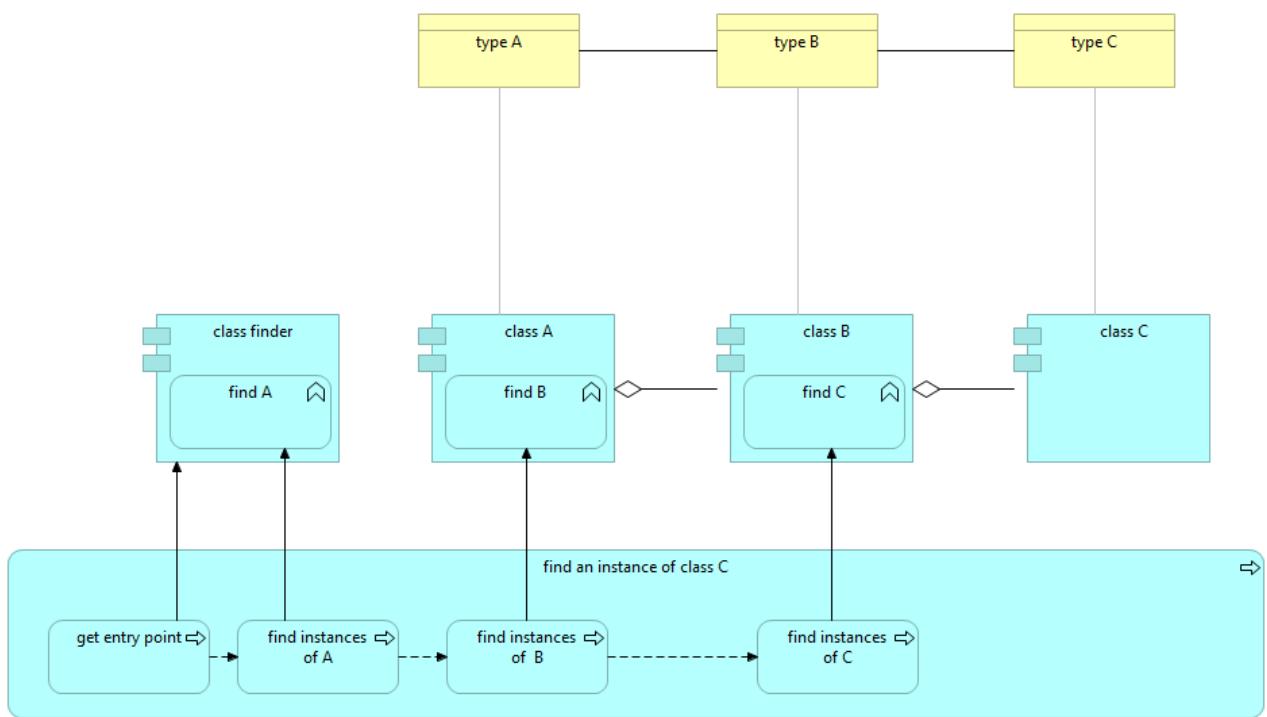


for example, delete object B

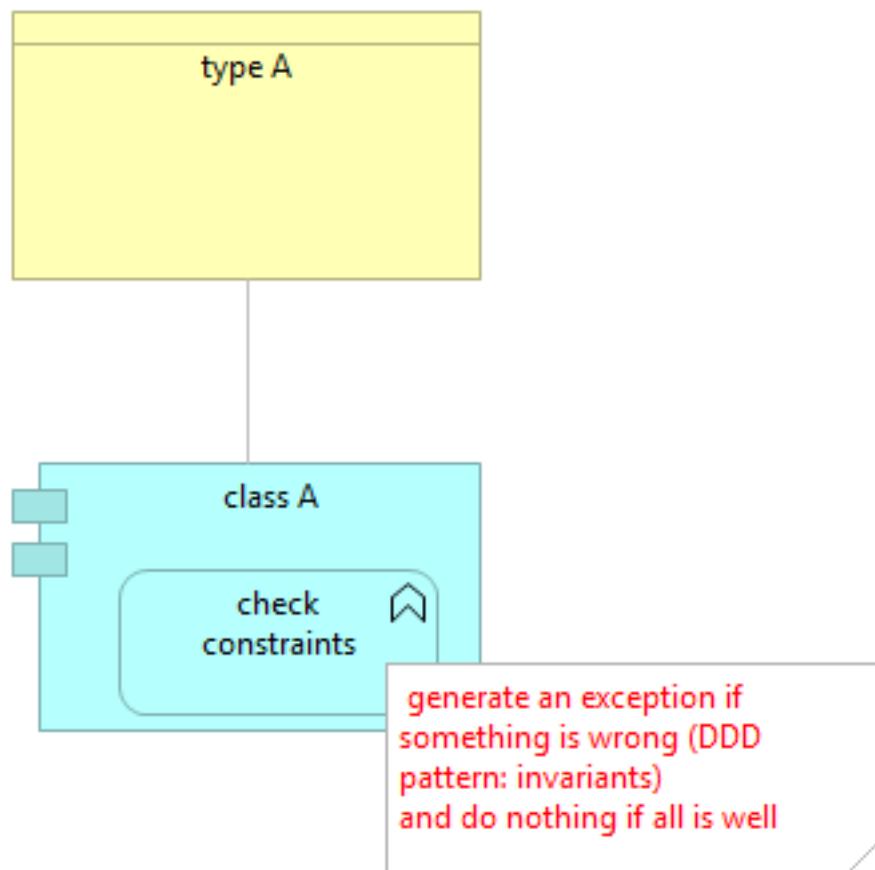
- all constraints are met
- no dangling references

object instance A

ENTRY POINT.



IMPLEMENTING CONSTRAINTS

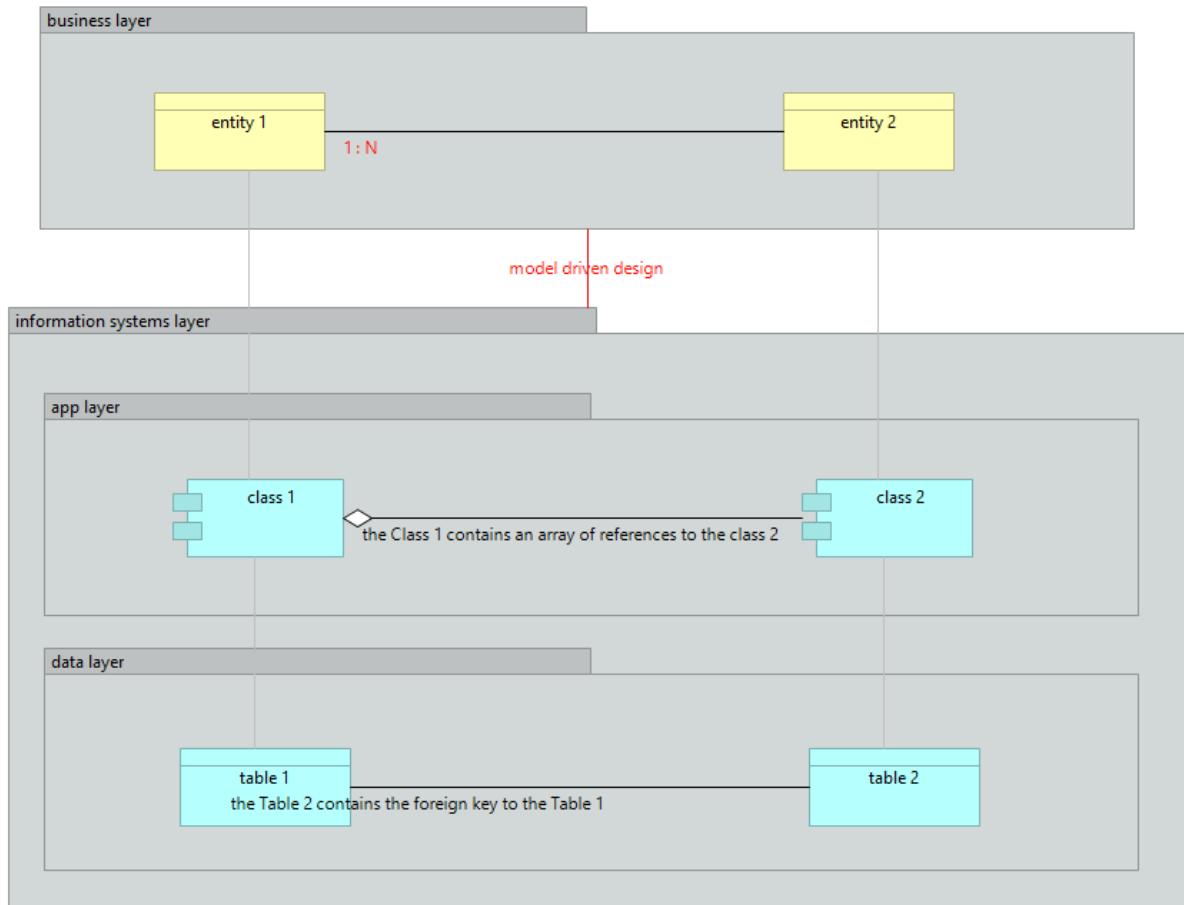


4 Domain Driven Design

MODEL AND STRUCTURAL ELEMENTS

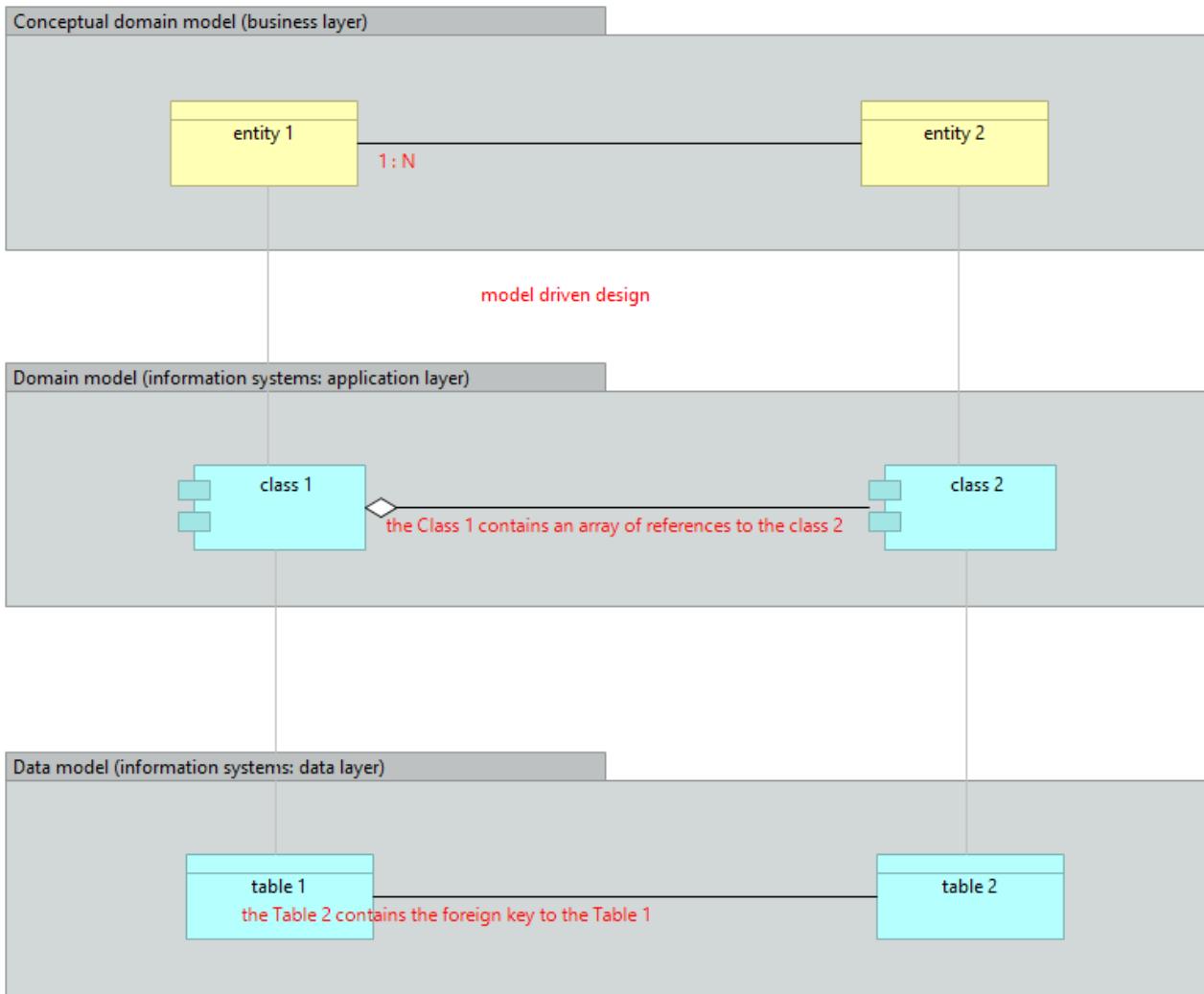
MODEL-DRIVEN DESIGN

layers - TOGAF's architecture domains



The creation of the model takes place simultaneously with the software implementation





Single responsibility principle from the SOLID model

Conceptual domain model (business layer)

Business Actor 1 ♀

Business Actor 2 ♂

If we go to the cause of class changes – those to the business customer, it turns out that each class is associated with one such 'person'. This is due to Conway's law that the structure of the program reflects the structure of the organization.

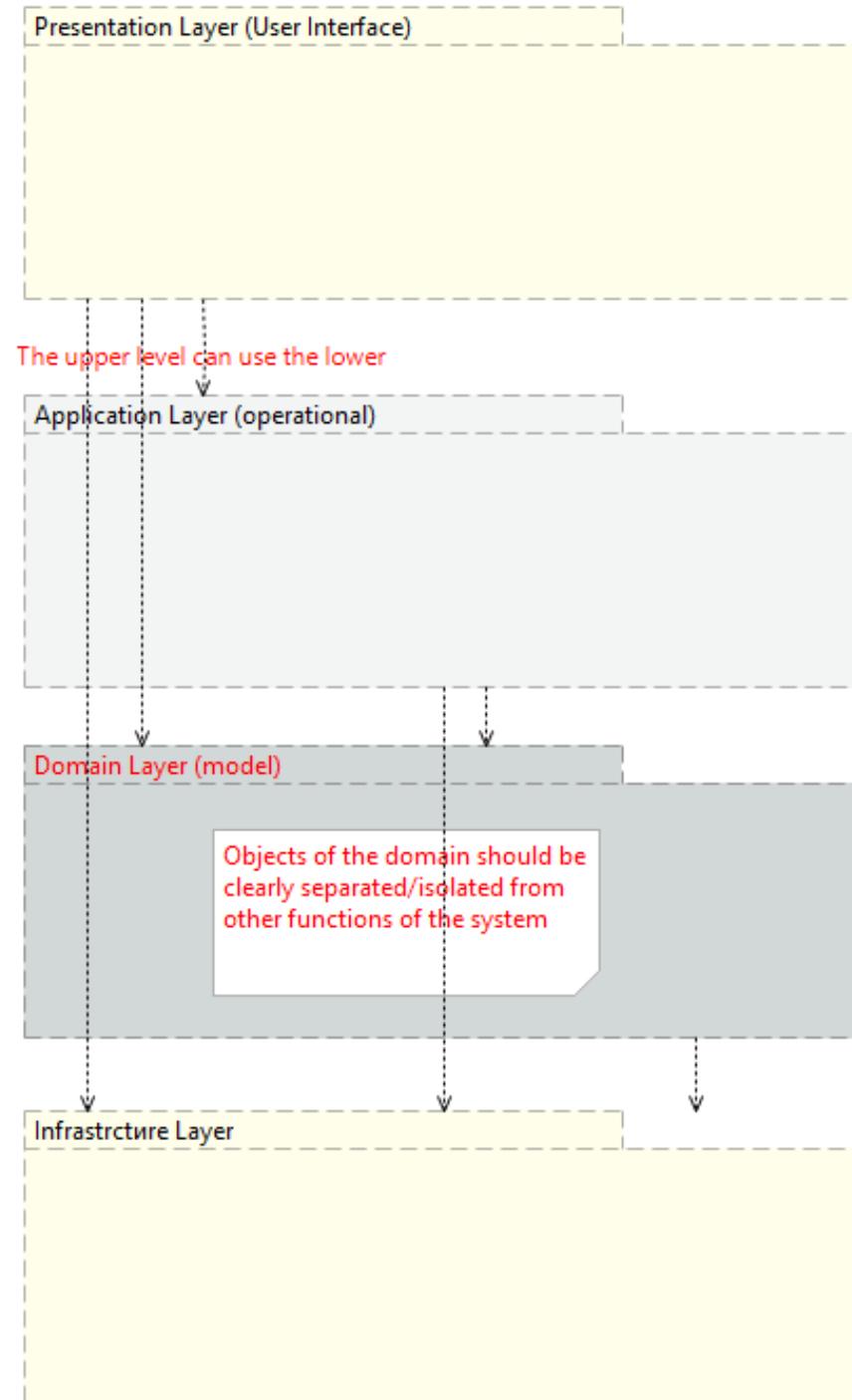
Domain model (information systems: application layer)

class 1

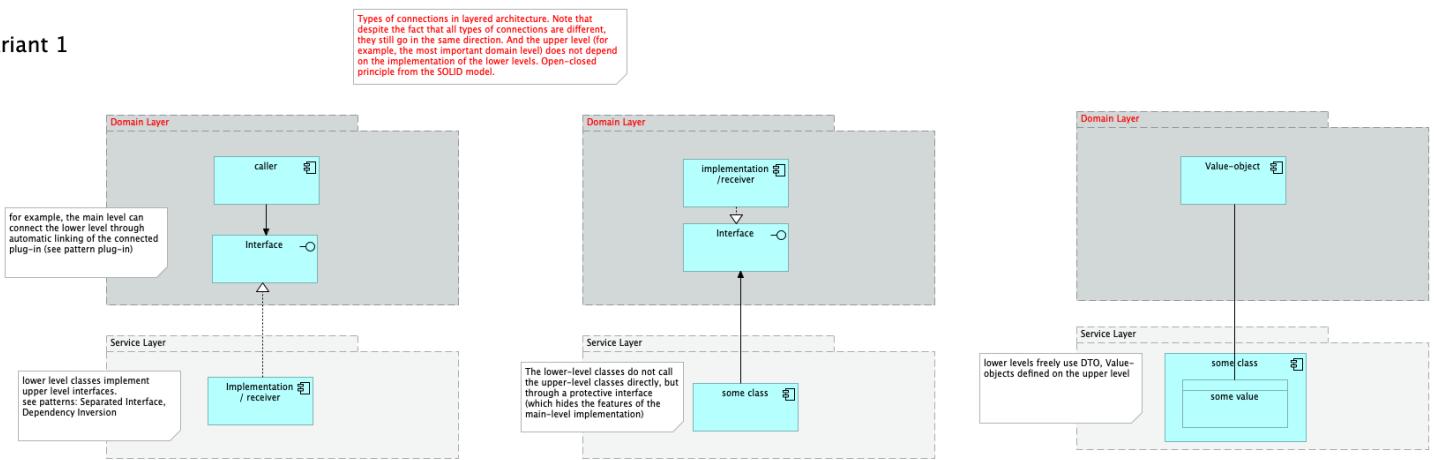
class 2

LAYERED ARCHITECTURE (ASYMMETRIC)

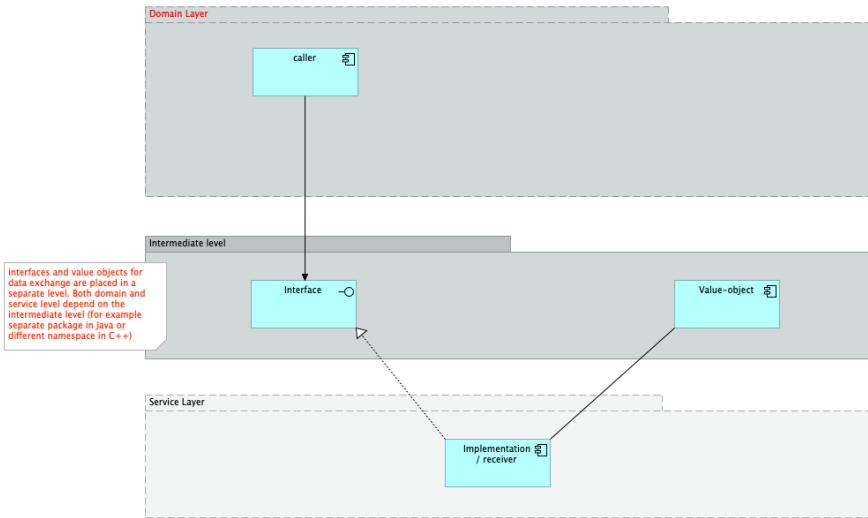
layered architecture
Only domain layer is required



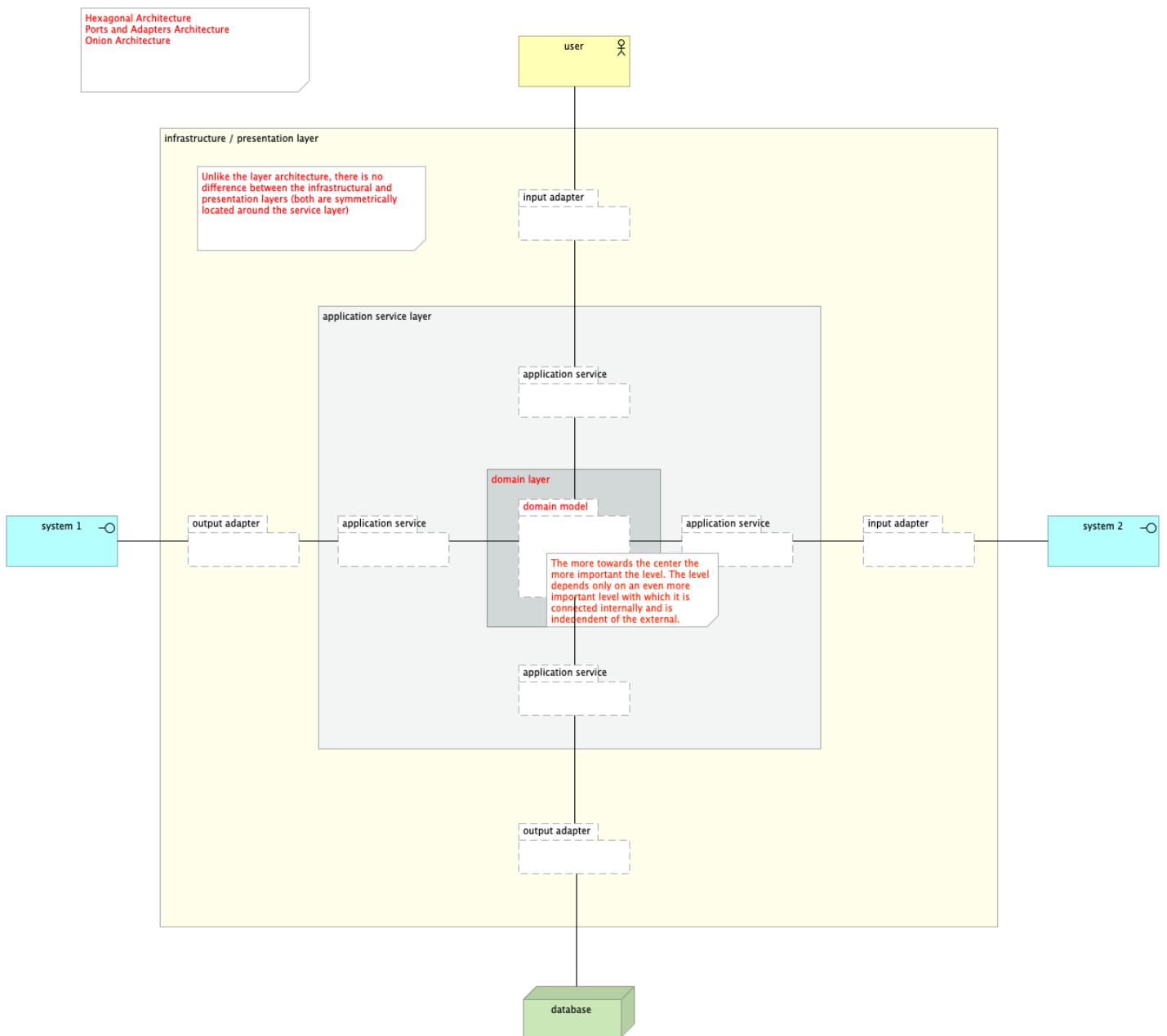
variant 1



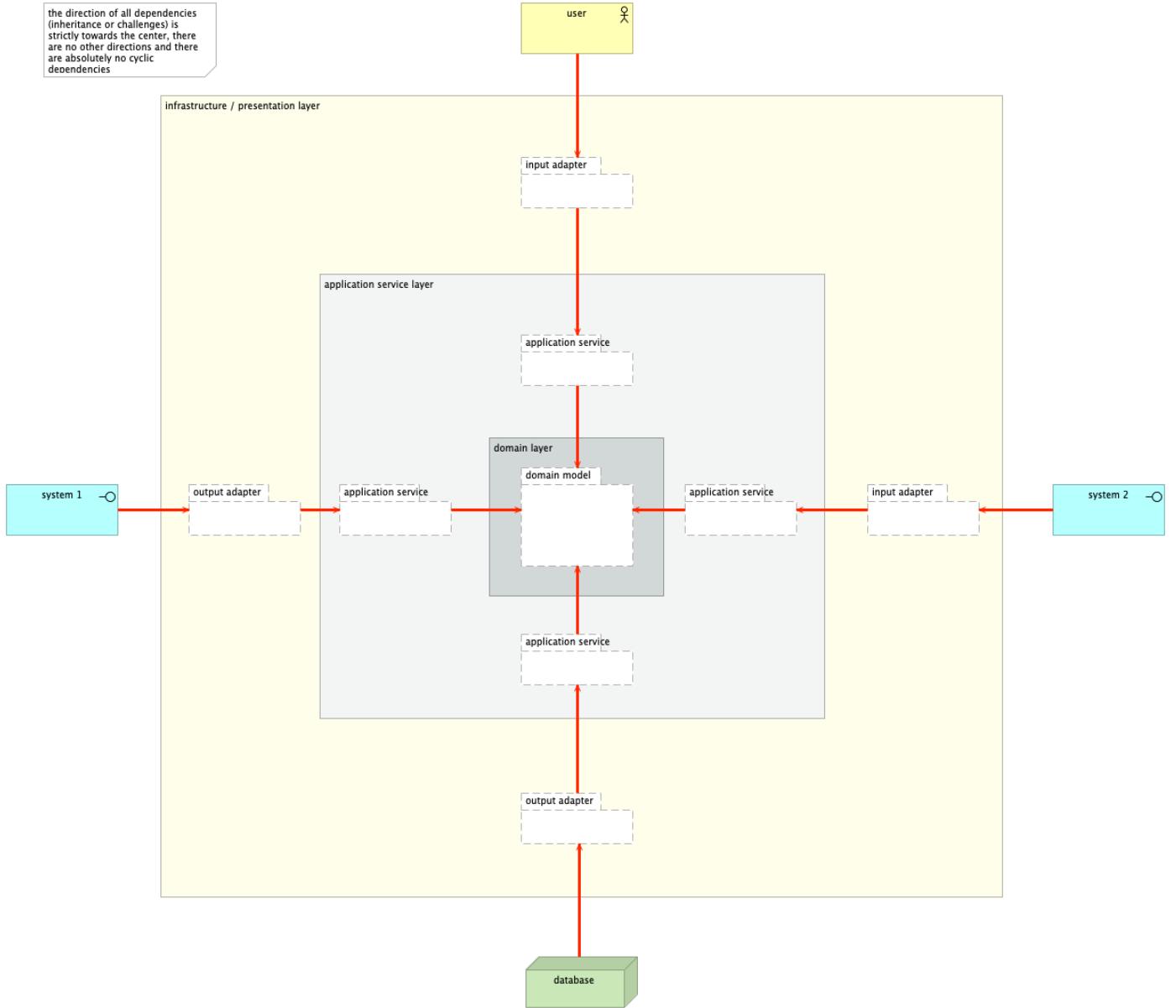
variant 2

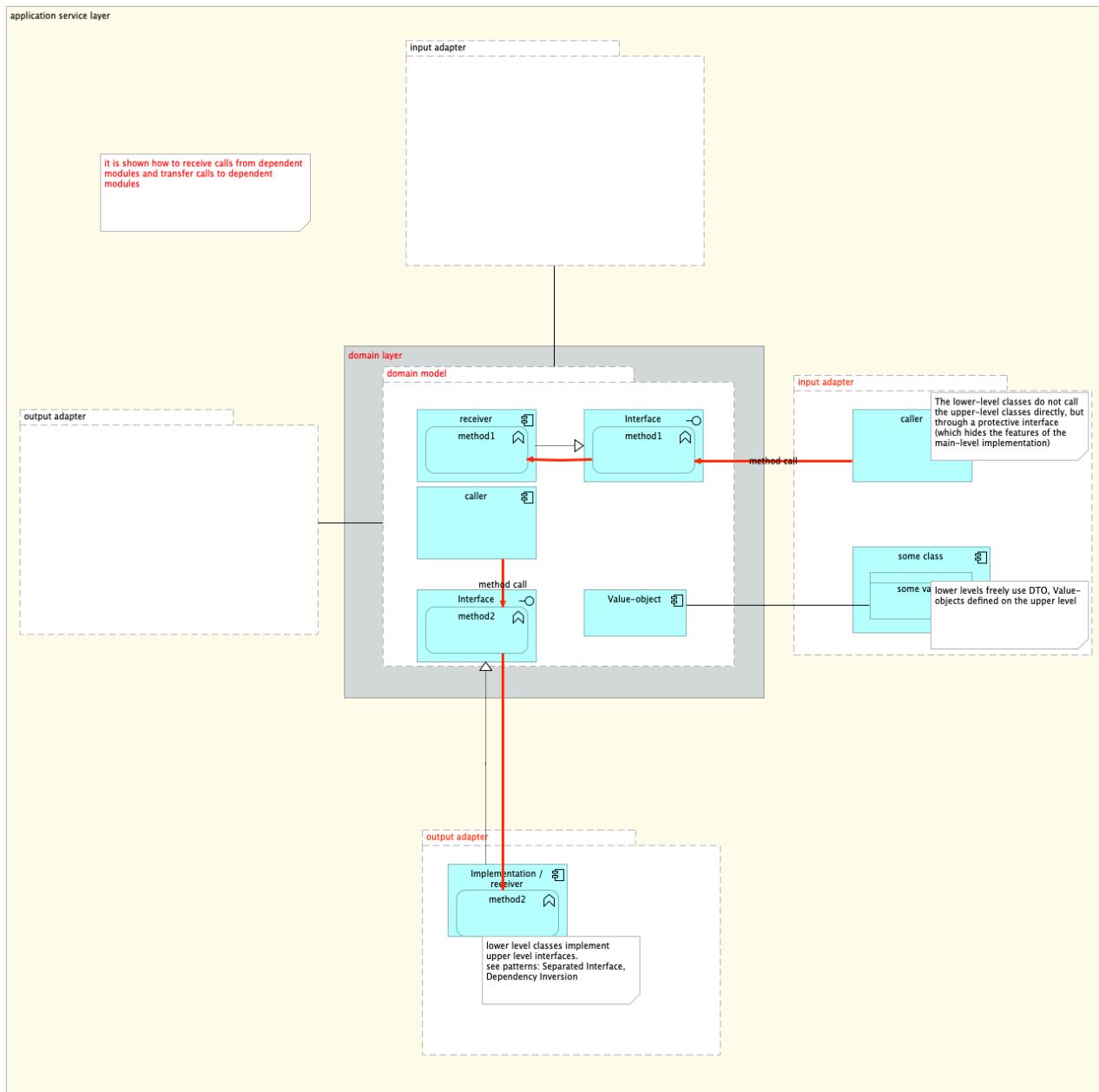


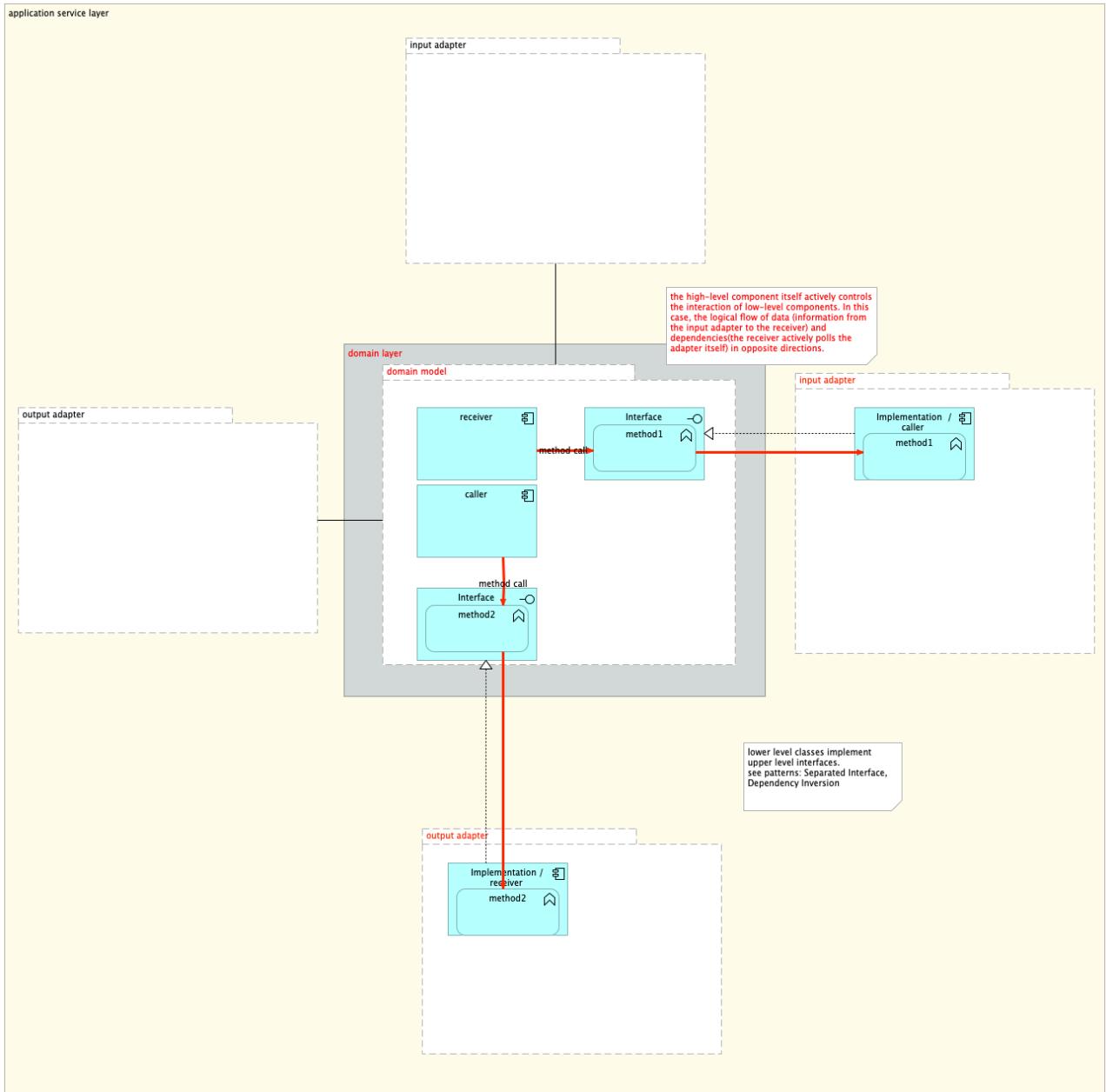
HEXAGONAL ARCHITECTURE (SYMMETRIC)



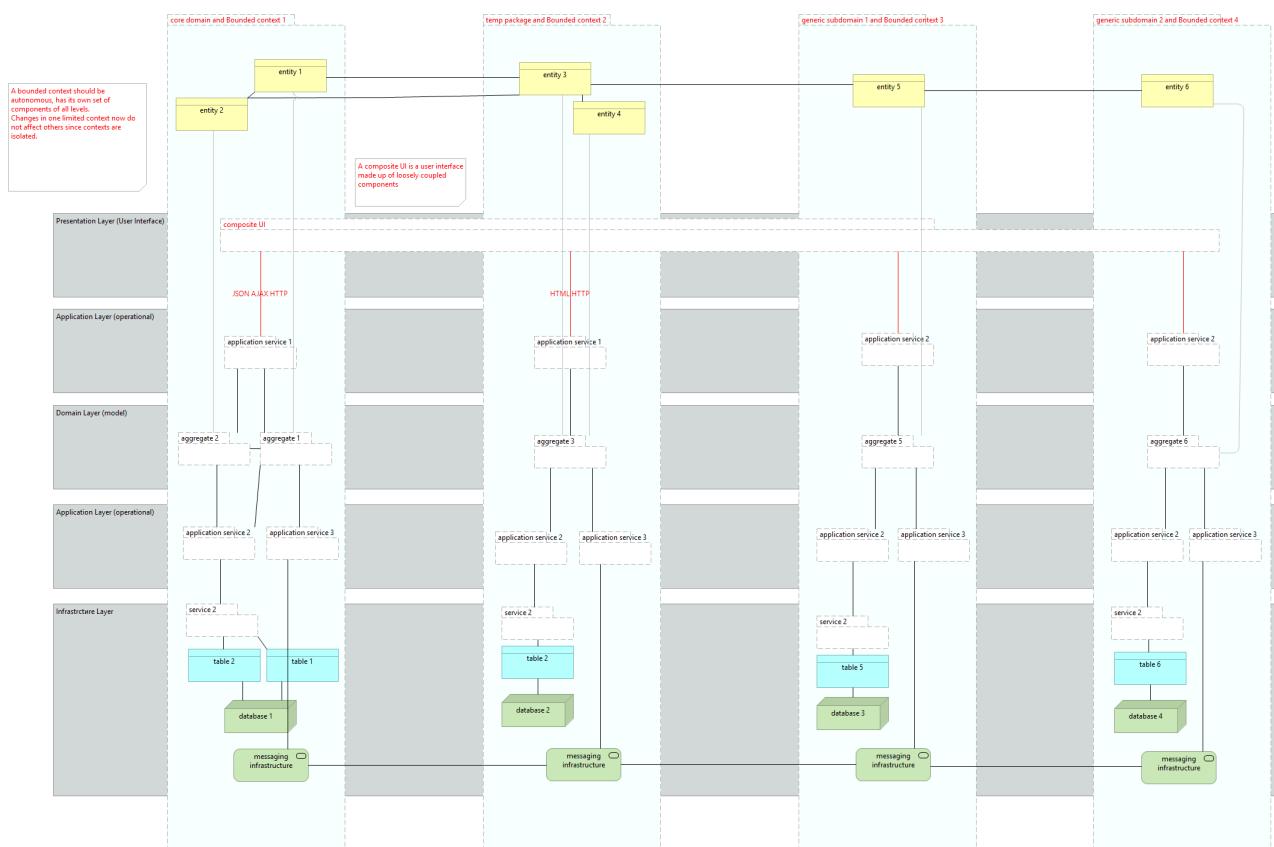
the direction of all dependencies (inheritance or challenges) is strictly towards the center, there are no other directions and there are absolutely no cyclic dependencies



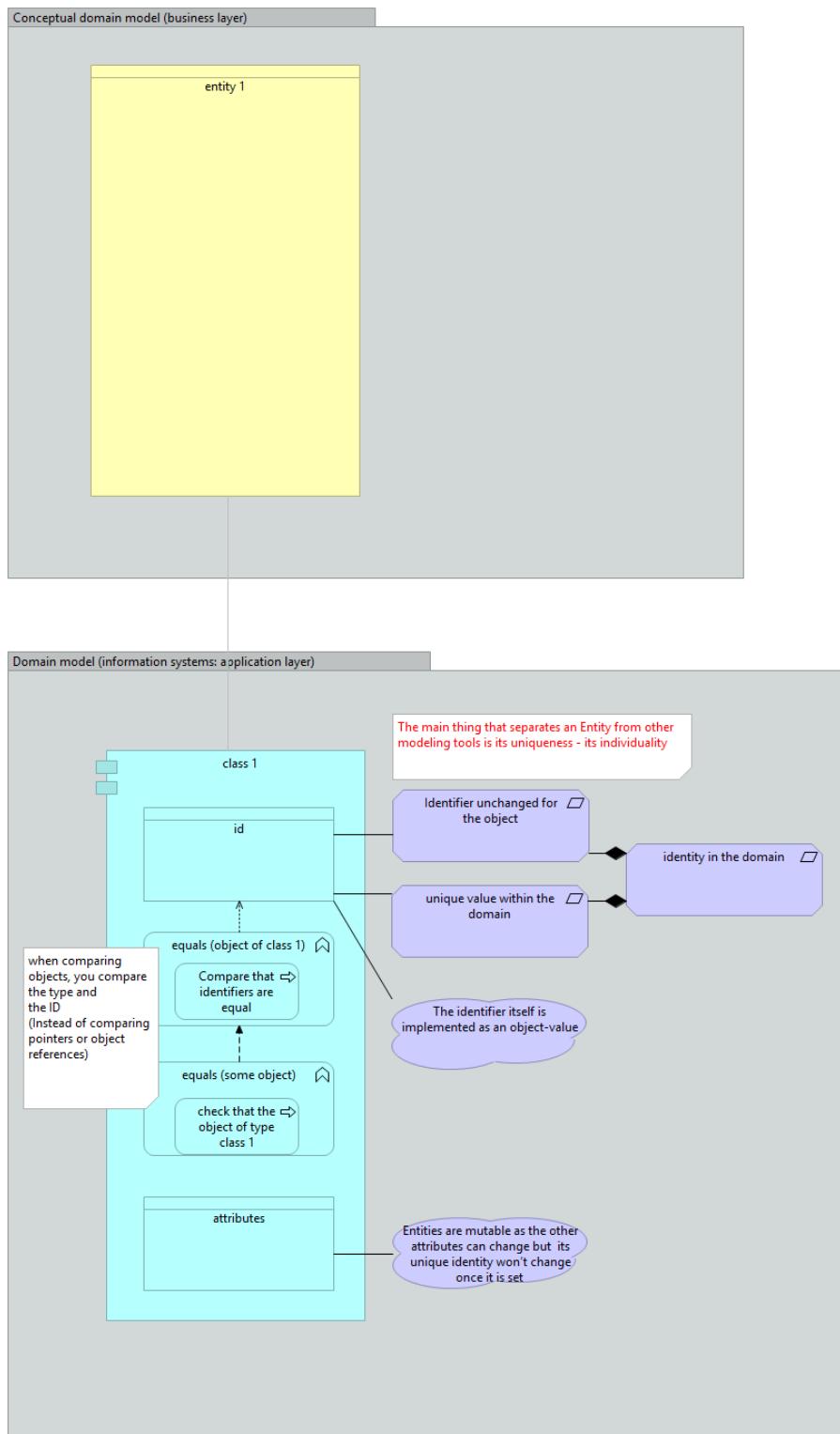


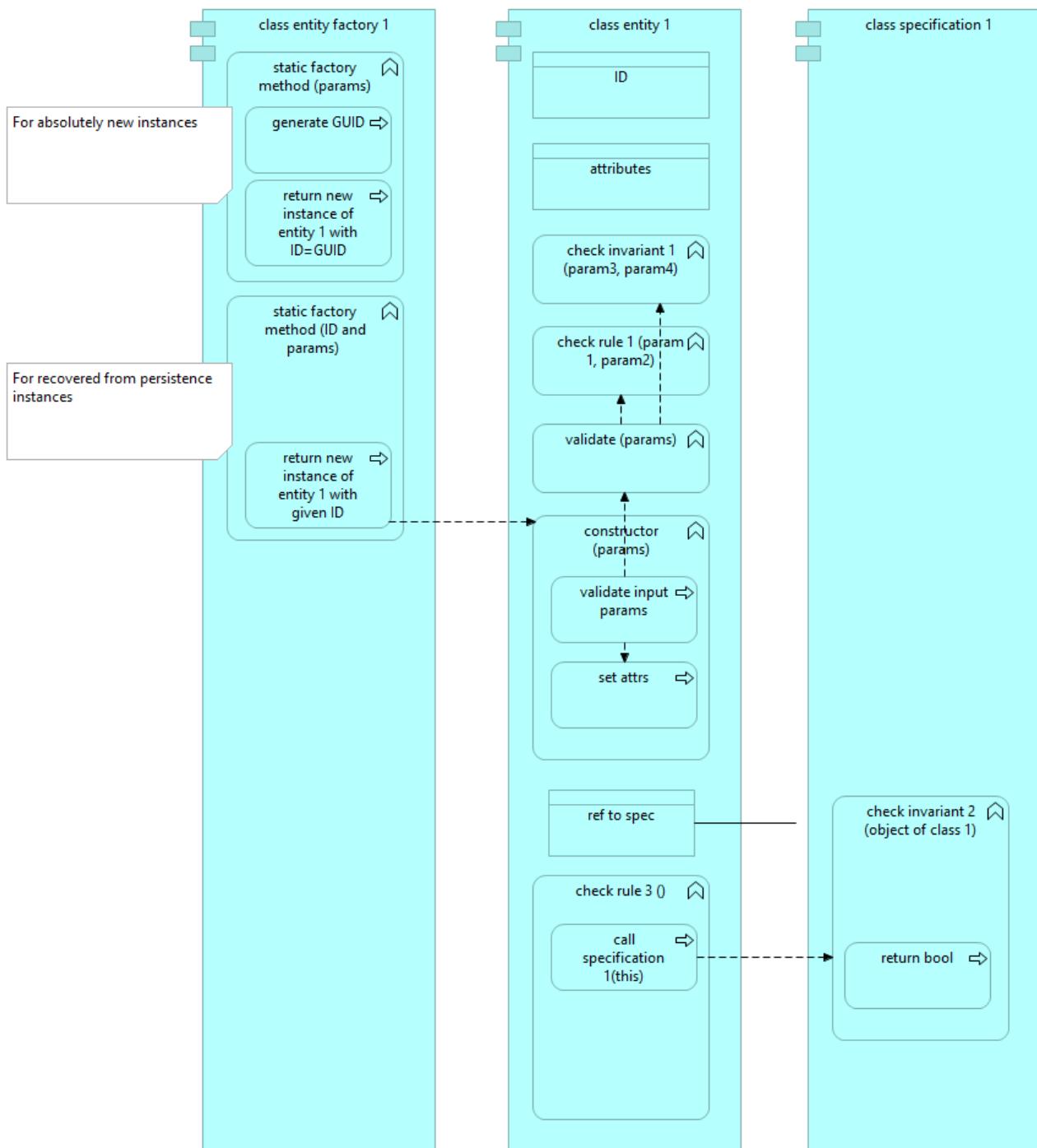


COMPOSITE UI

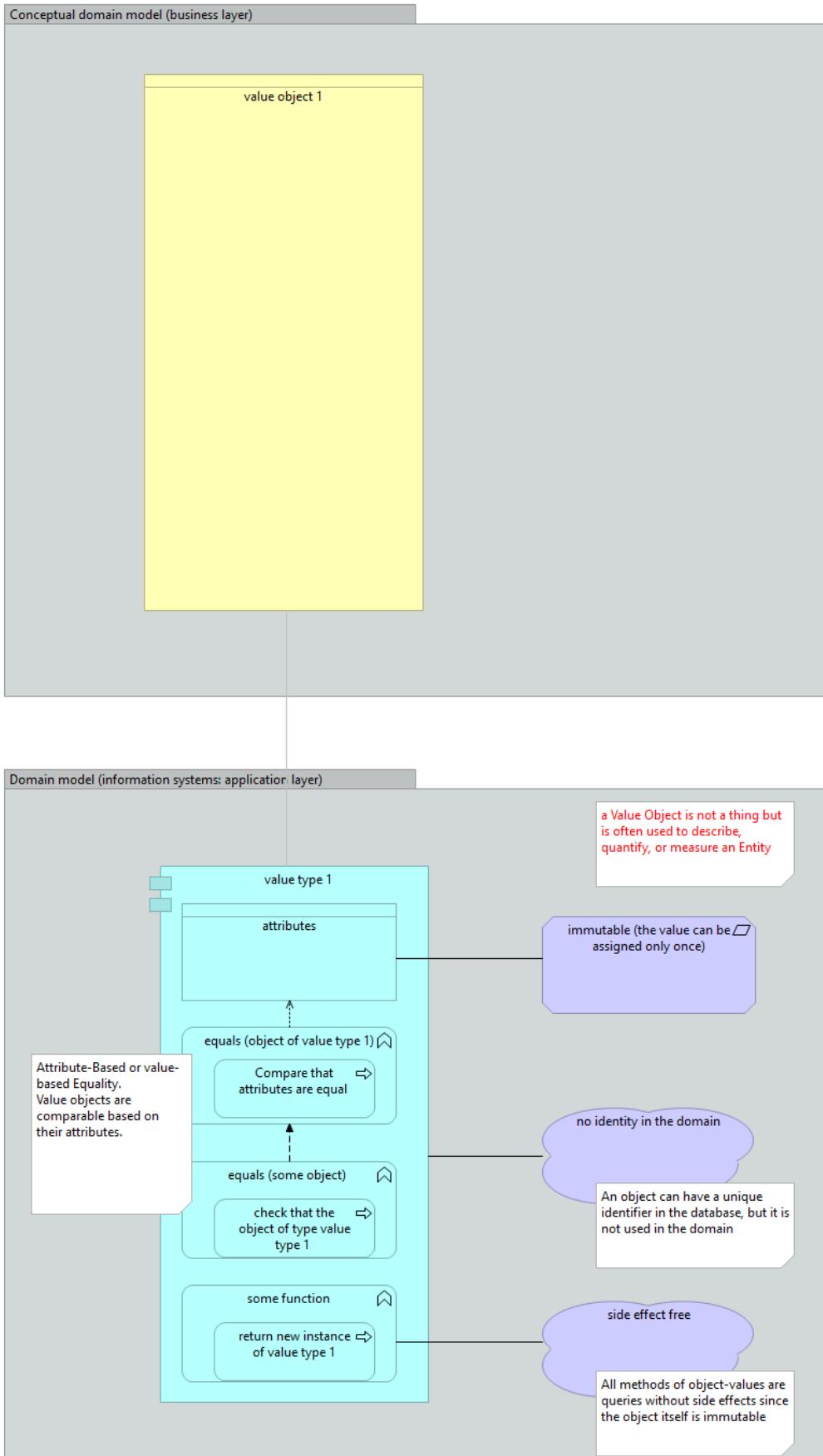


ENTITIES

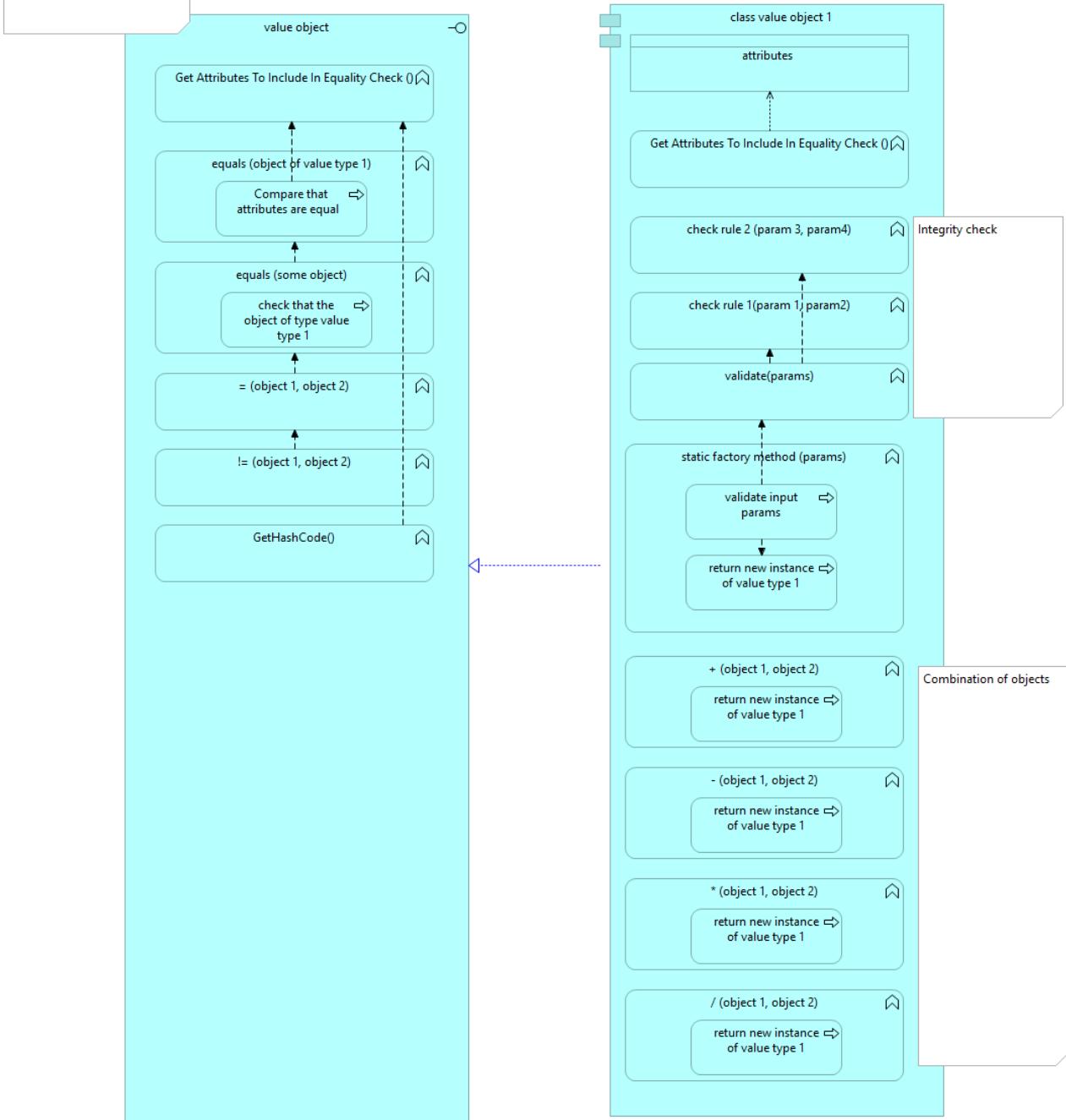




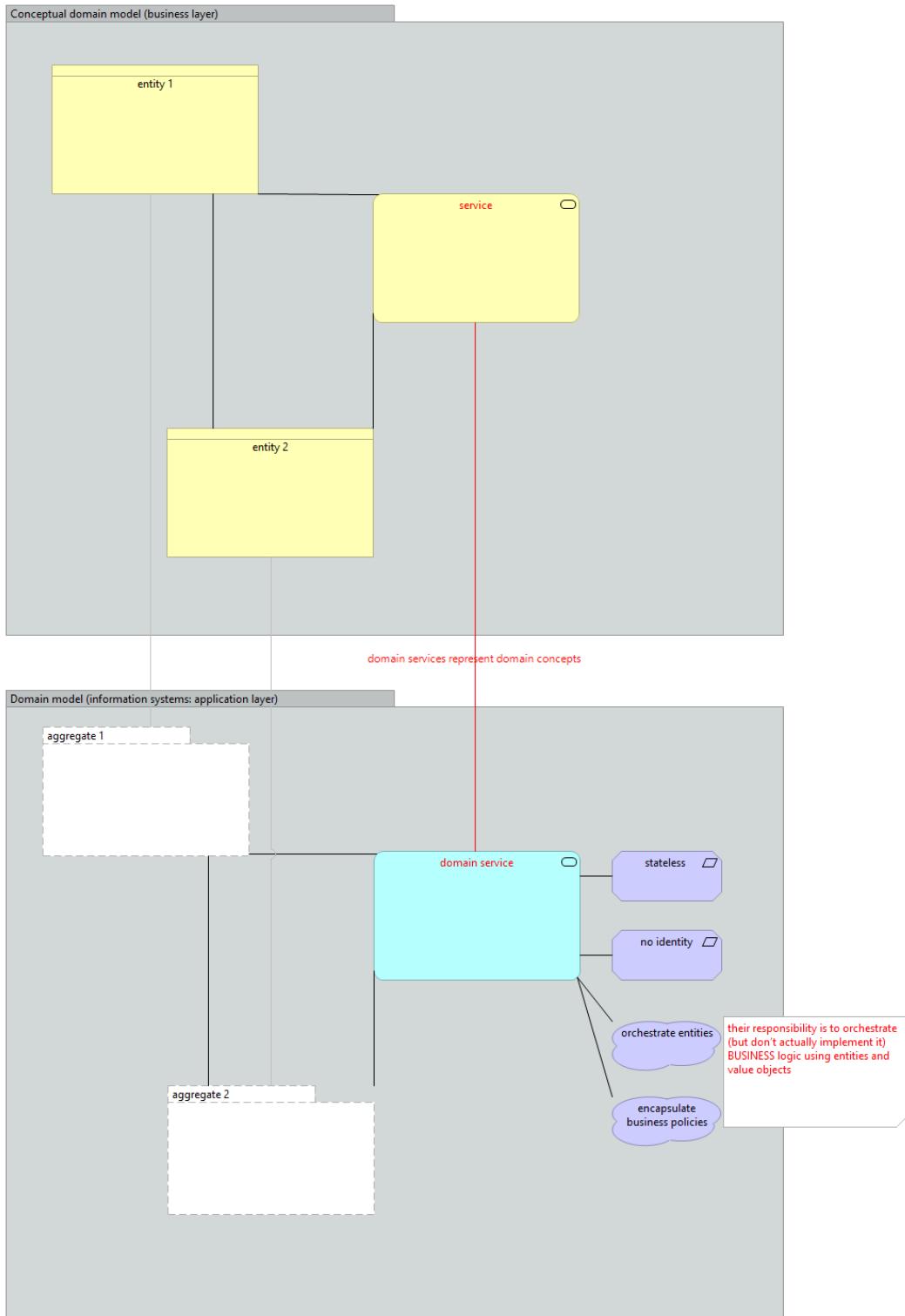
VALUE-OBJECTS



See example: Fowler's Money

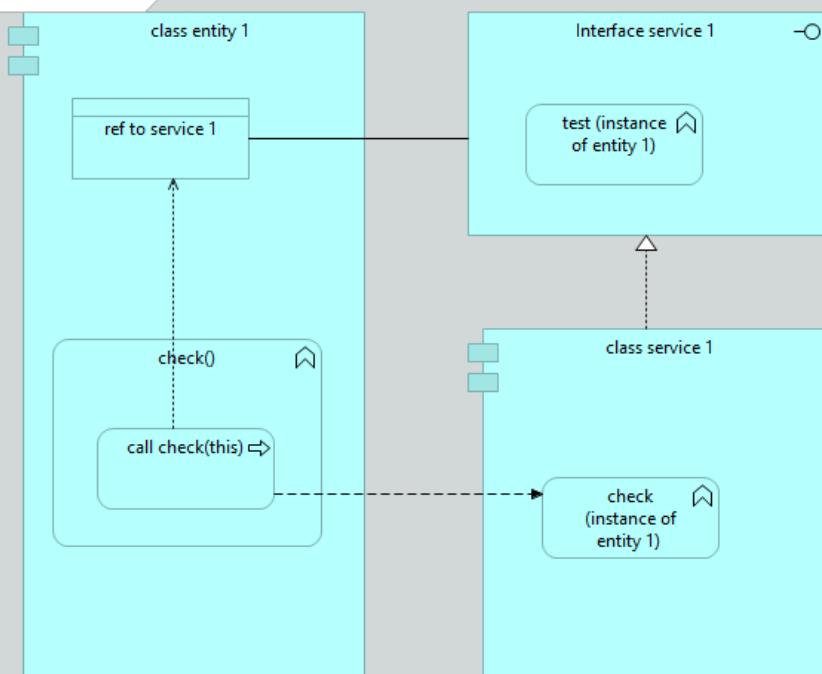


DOMAIN SERVICES

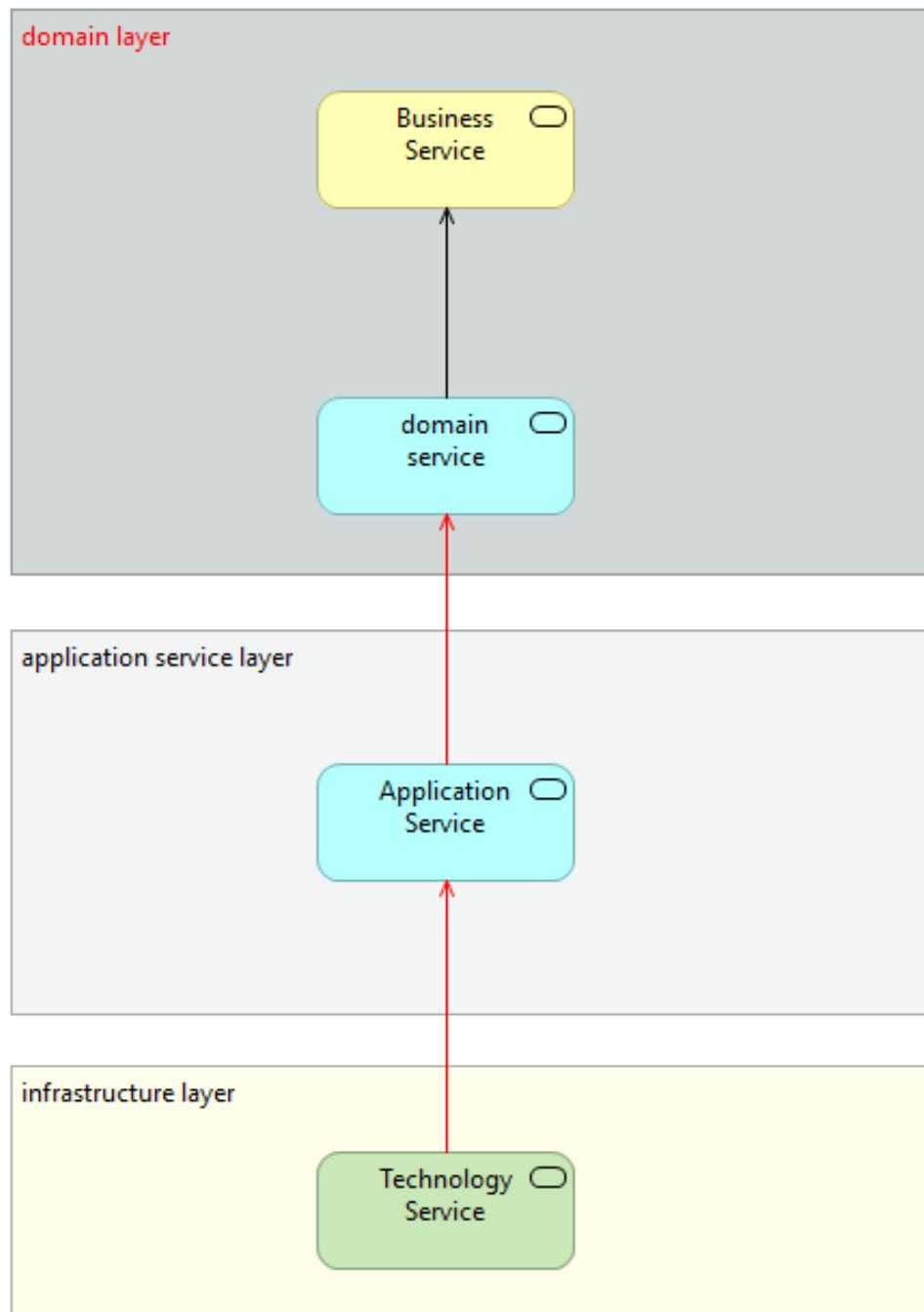


Domain model (information systems: application layer)

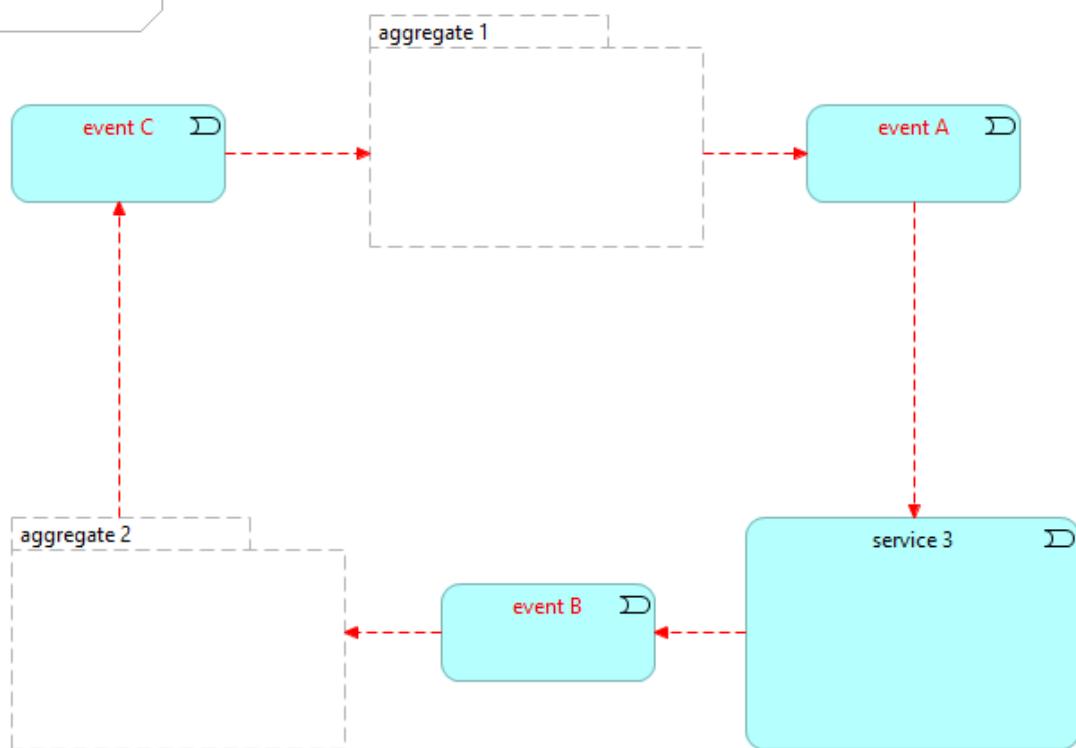
Example of communicating services with aggregates via direct service call

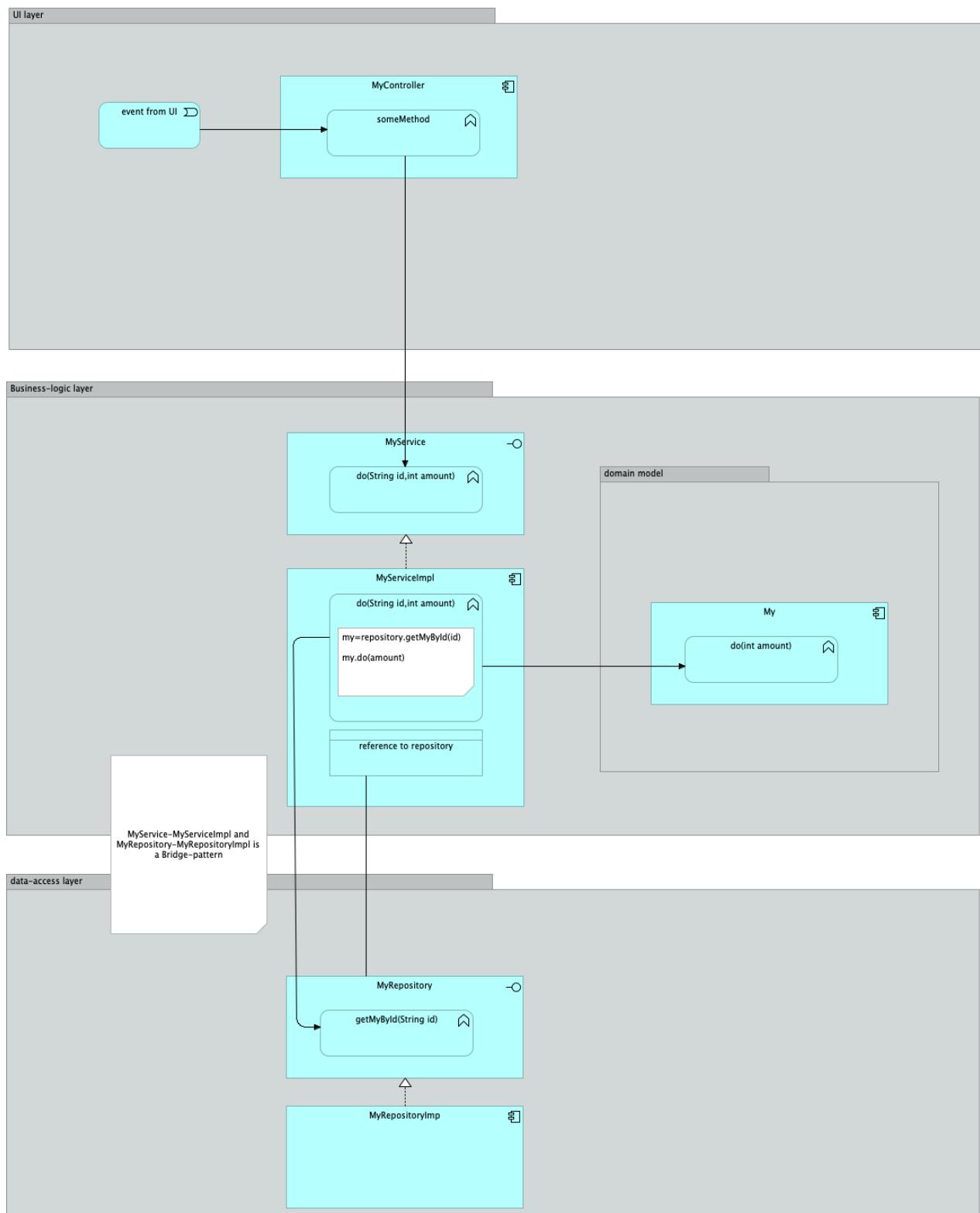


Services support each other

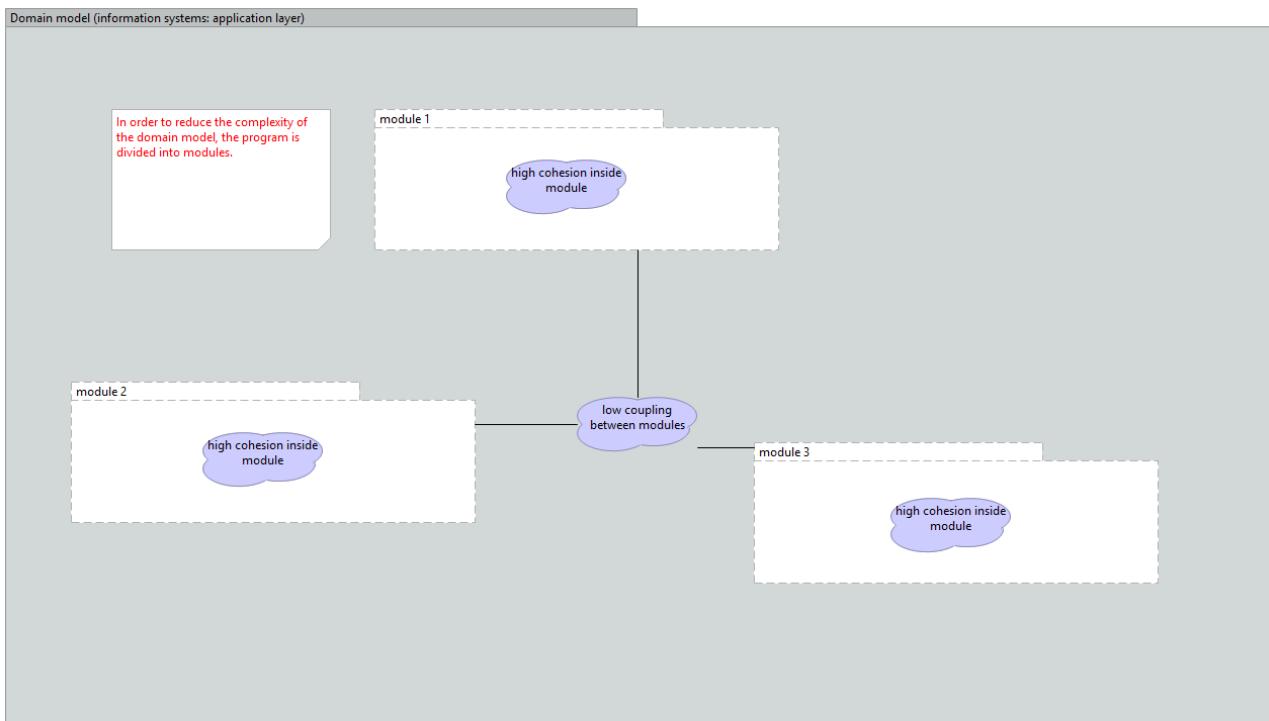


Example of communicating services with aggregates through events

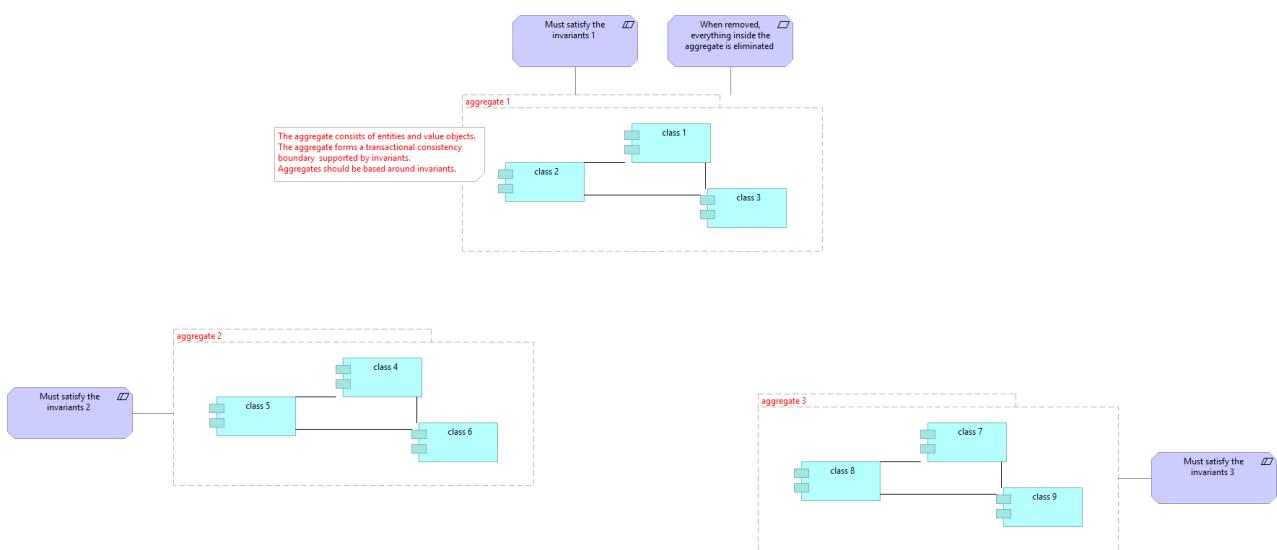




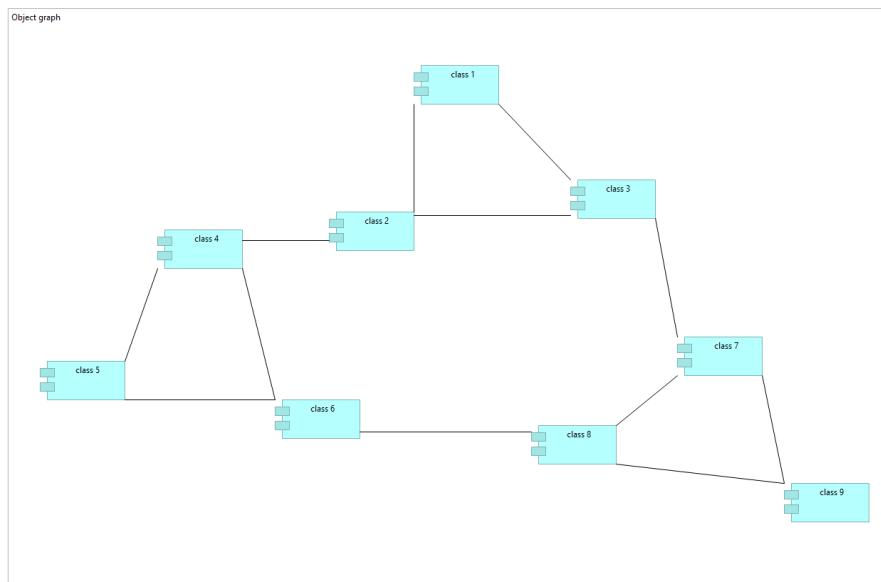
MODULES



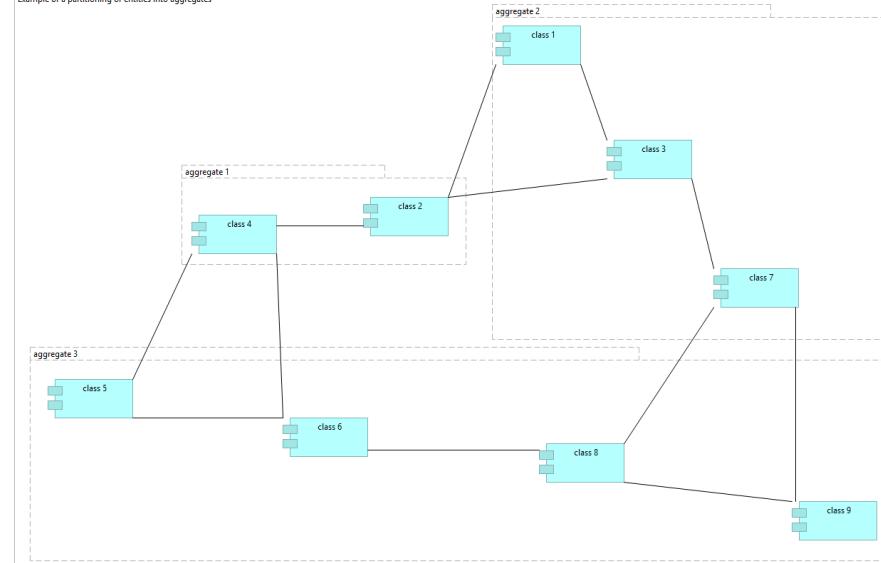
AGGREGATES



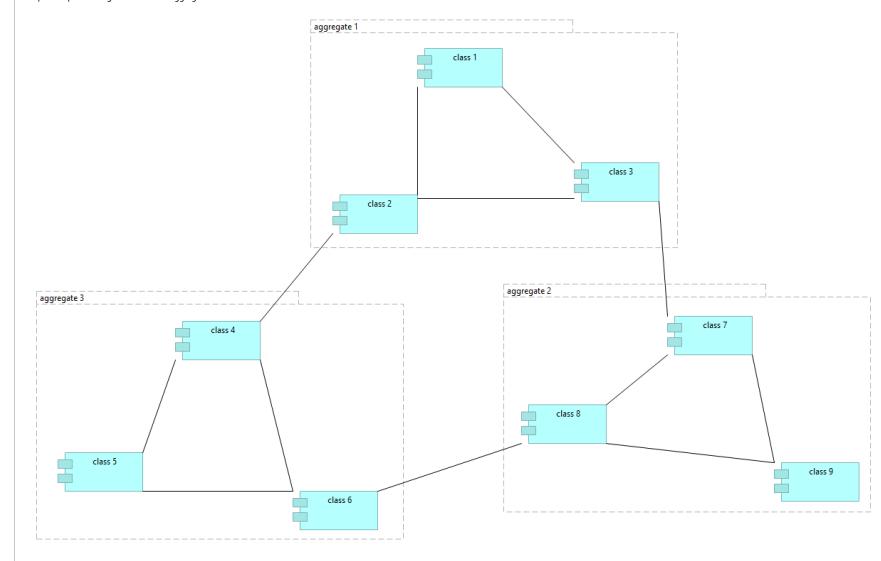
In the course of modeling, think about different ways of splitting the model into aggregates



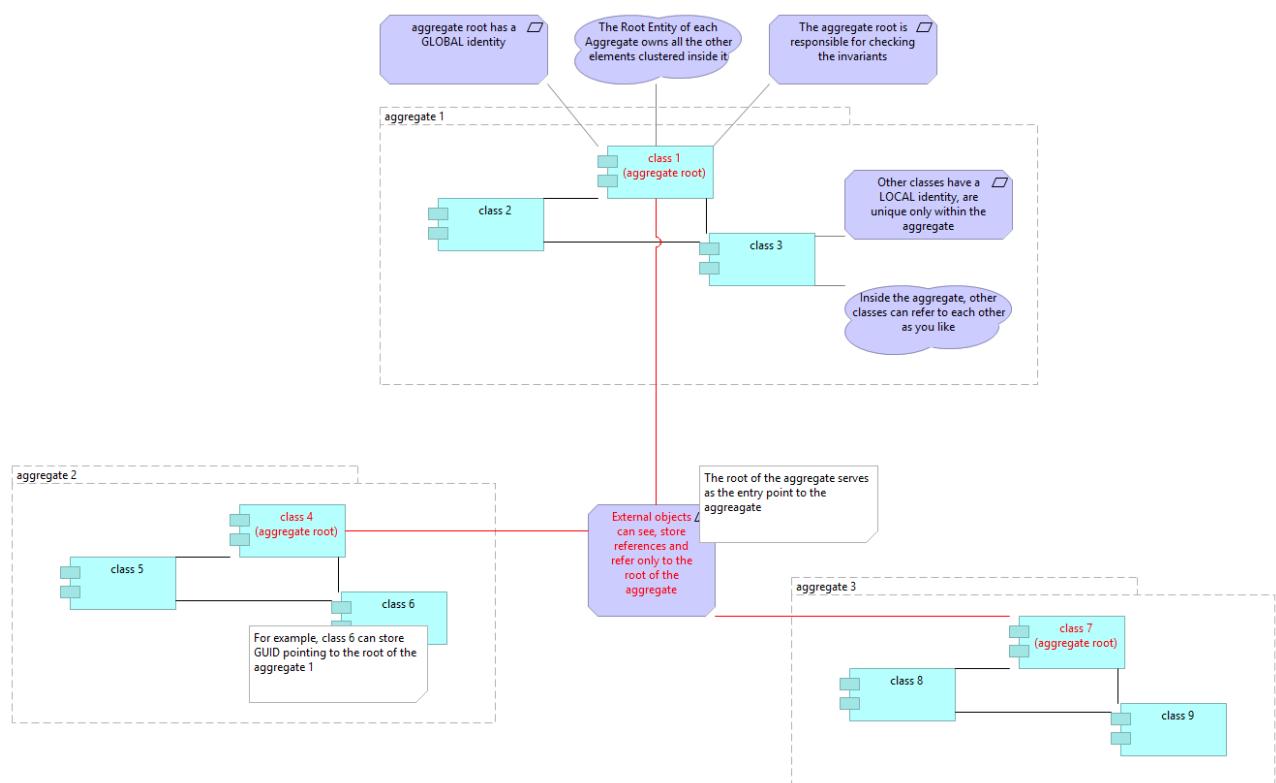
Example of a partitioning of entities into aggregates



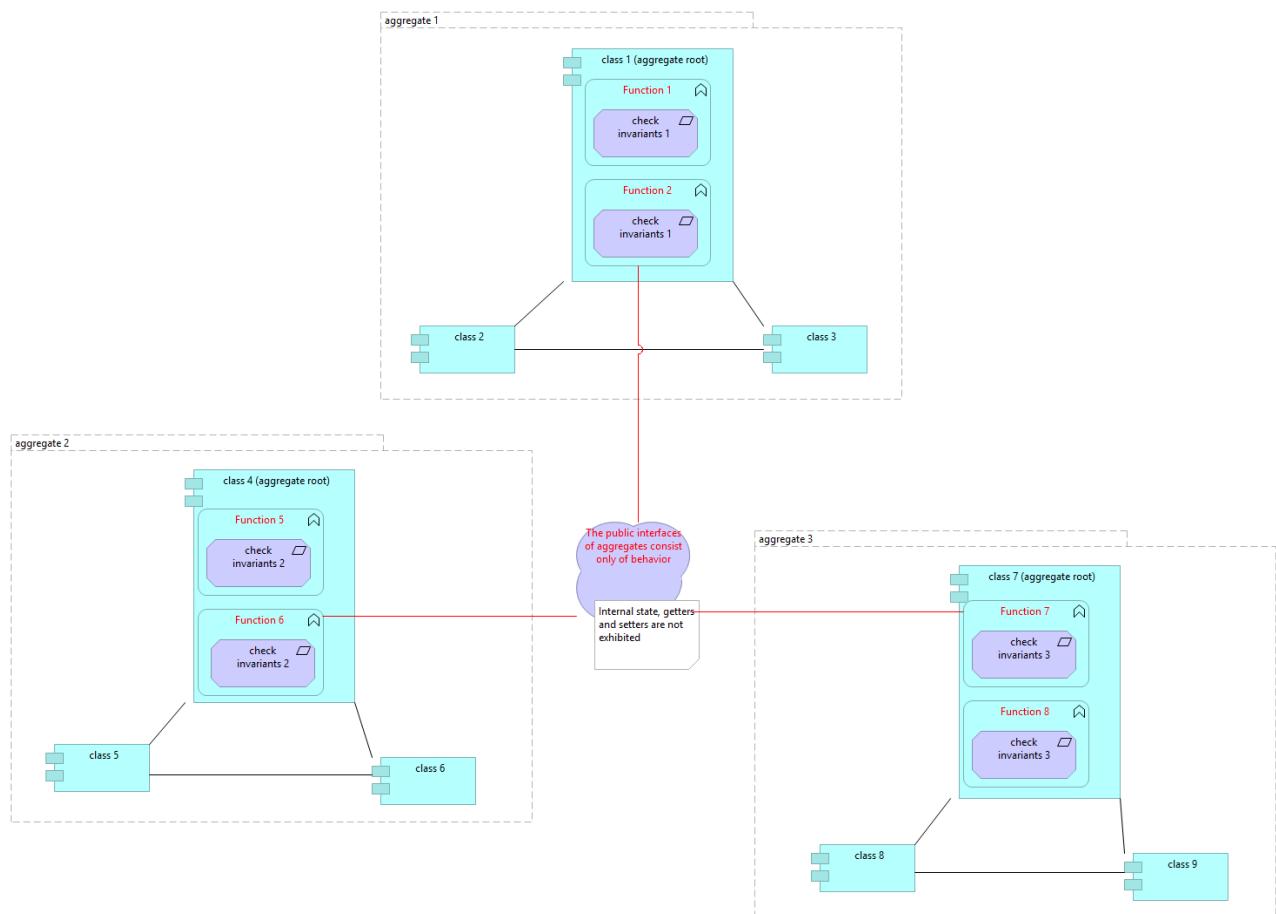
Example of a partitioning of entities into aggregates



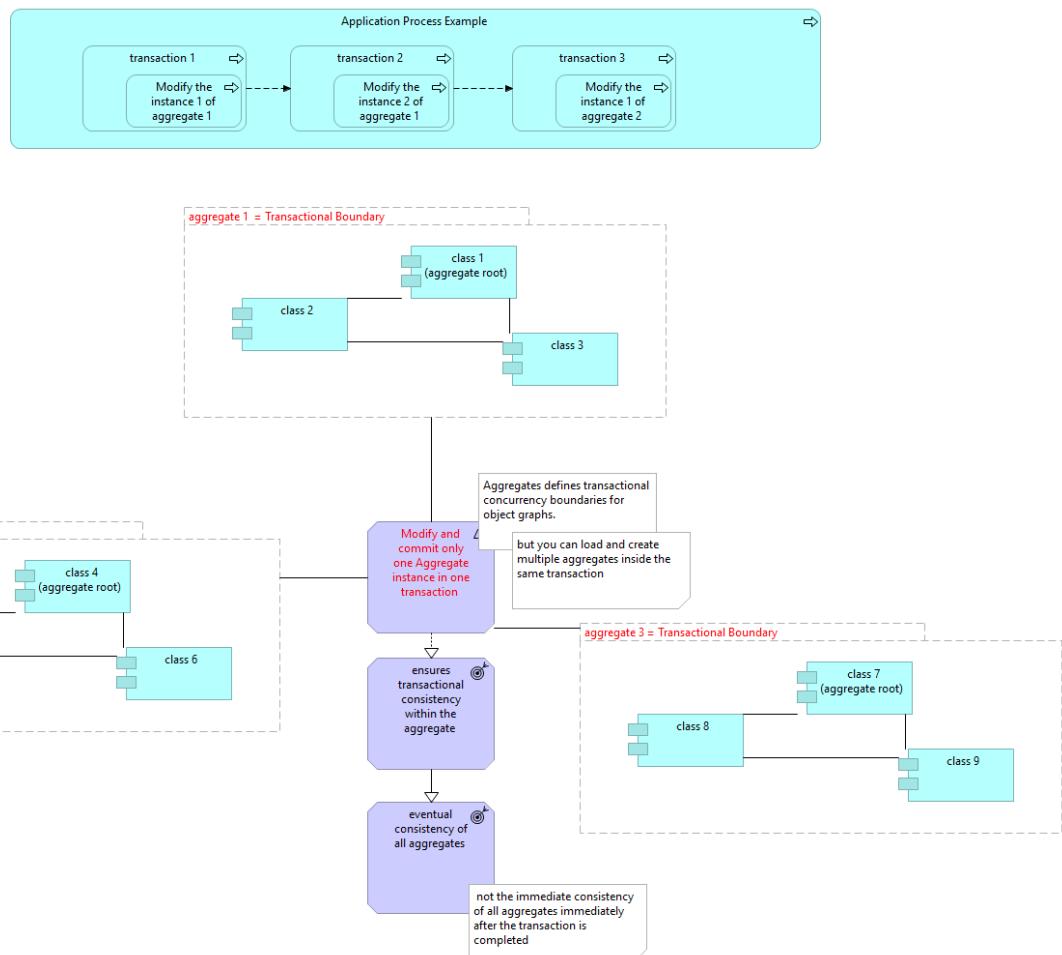
AGGREGATE ROOT



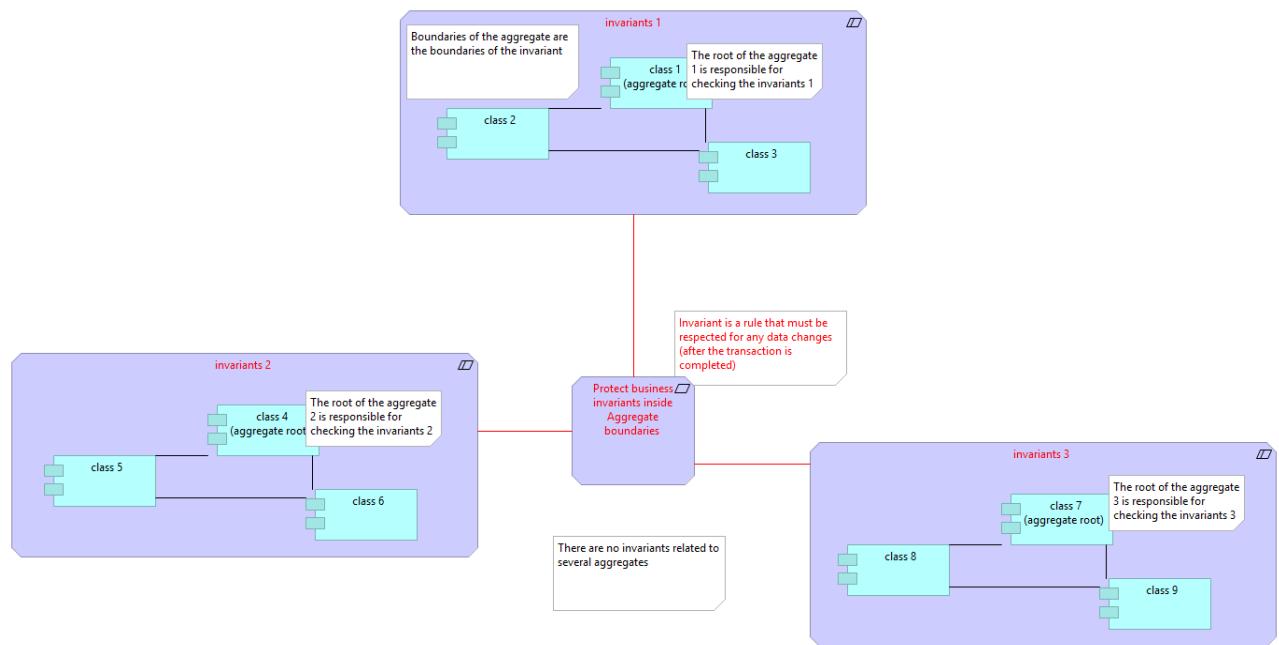
BEHAVIOR-FOCUSED AGGREGATE ROOT



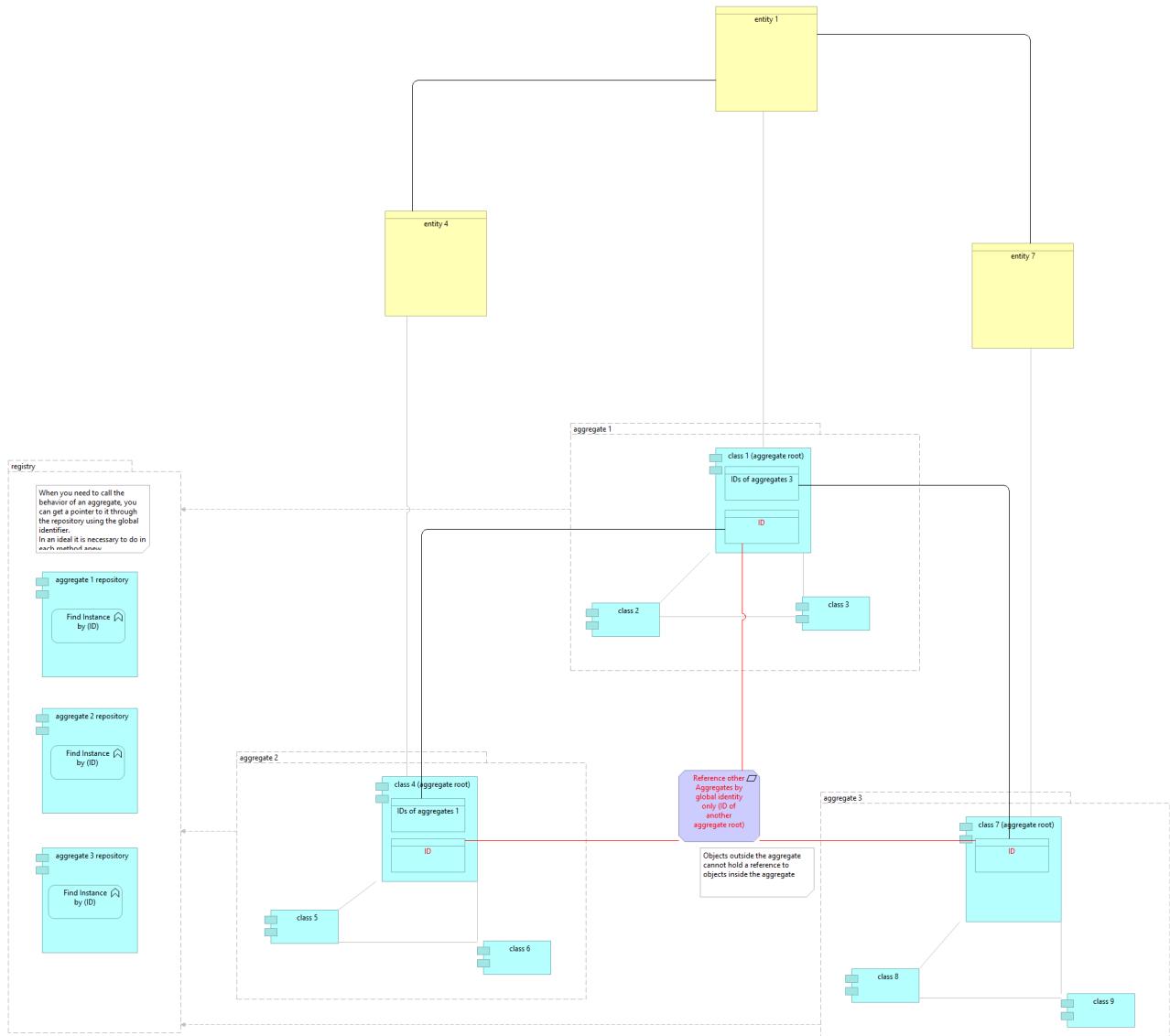
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION



PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES



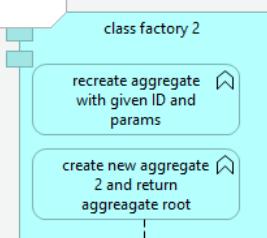
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY



FACTORIES

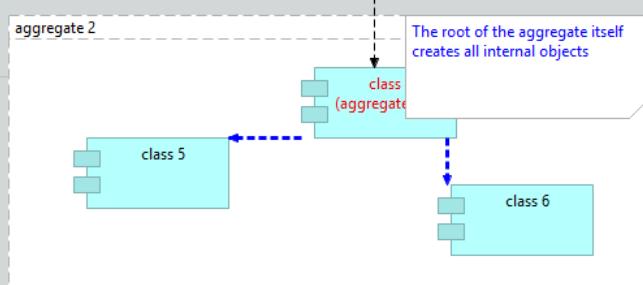
application service layer

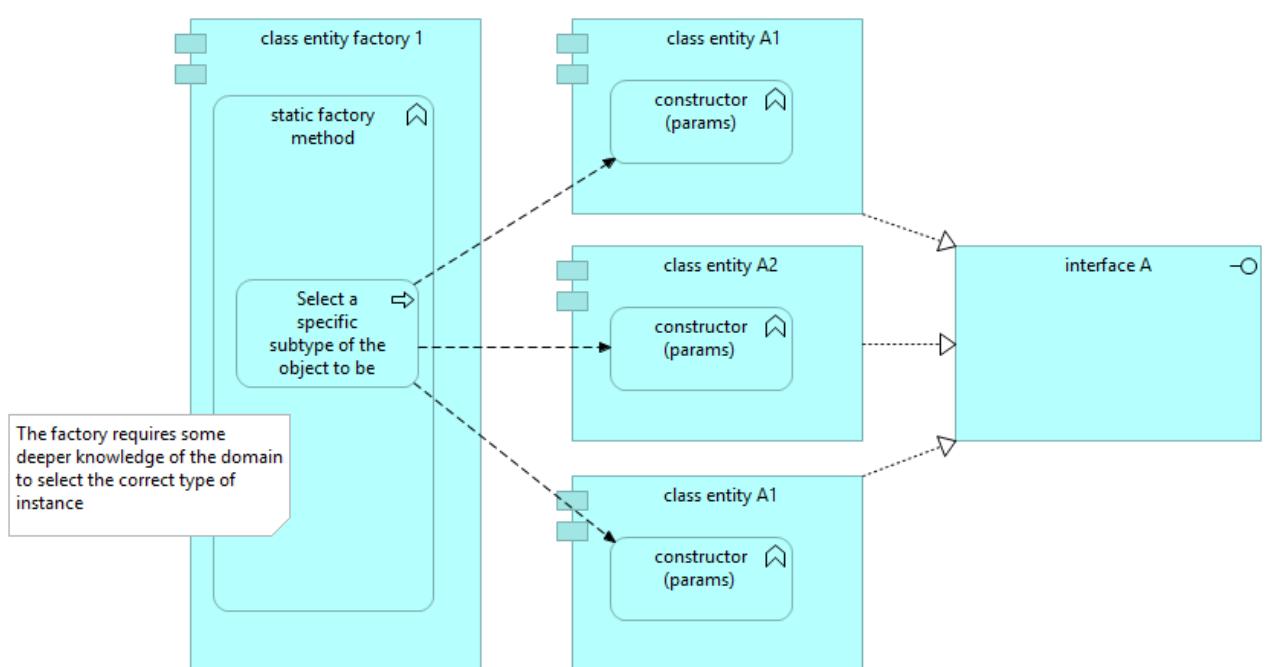
Use factory to create aggregates, complex entities and value objects.
Use factory to re-create domain objects from persistent storage.
The factory creates an aggregate entirely, with the satisfaction of all invariants
GoF pattern: factory



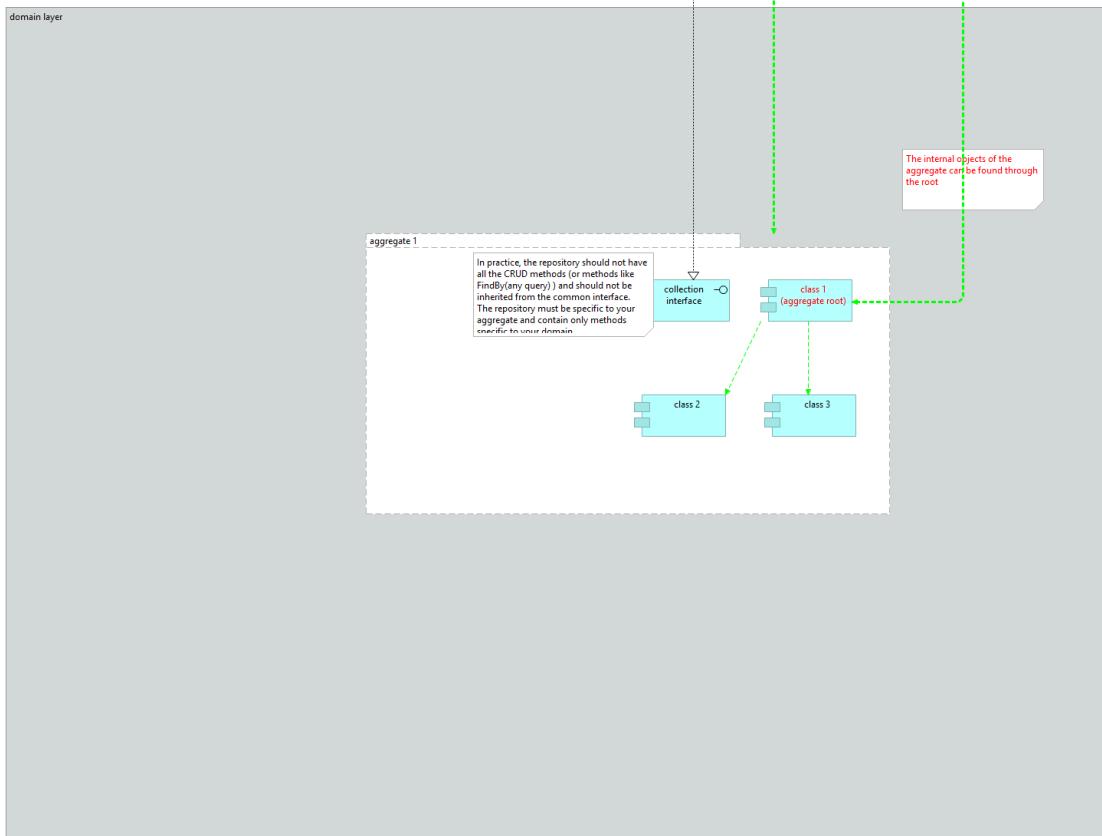
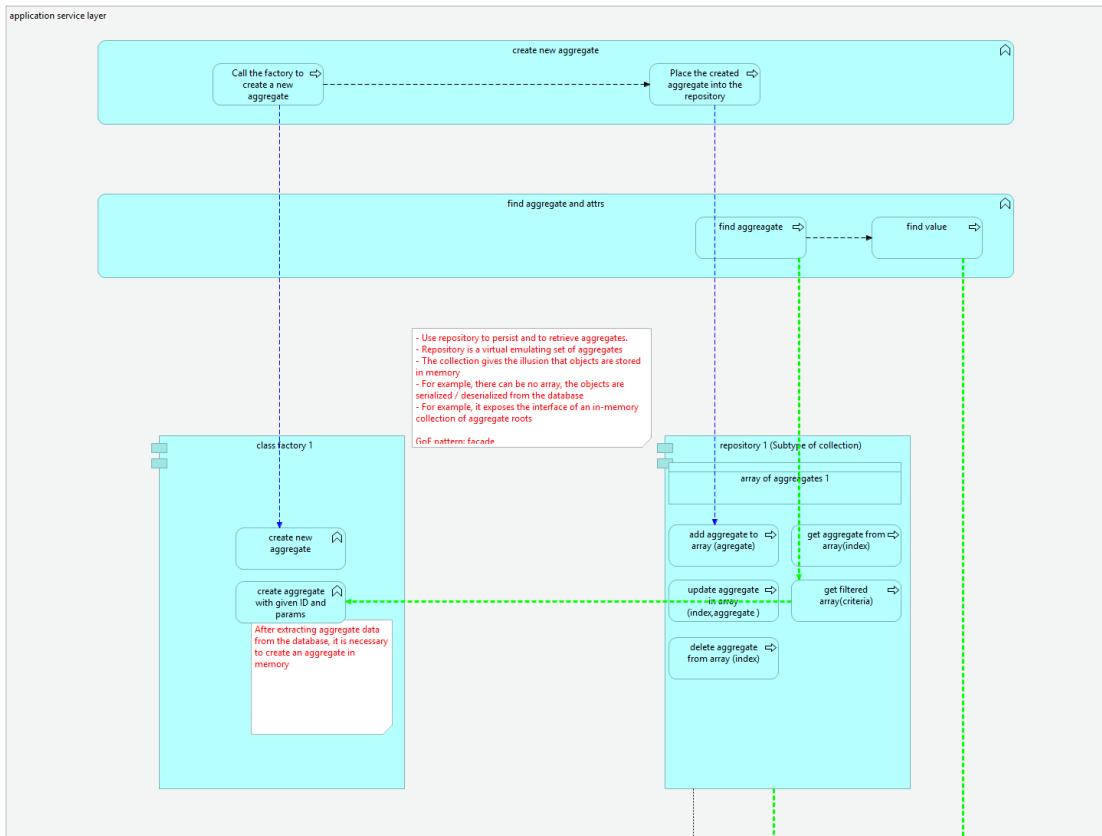
domain layer

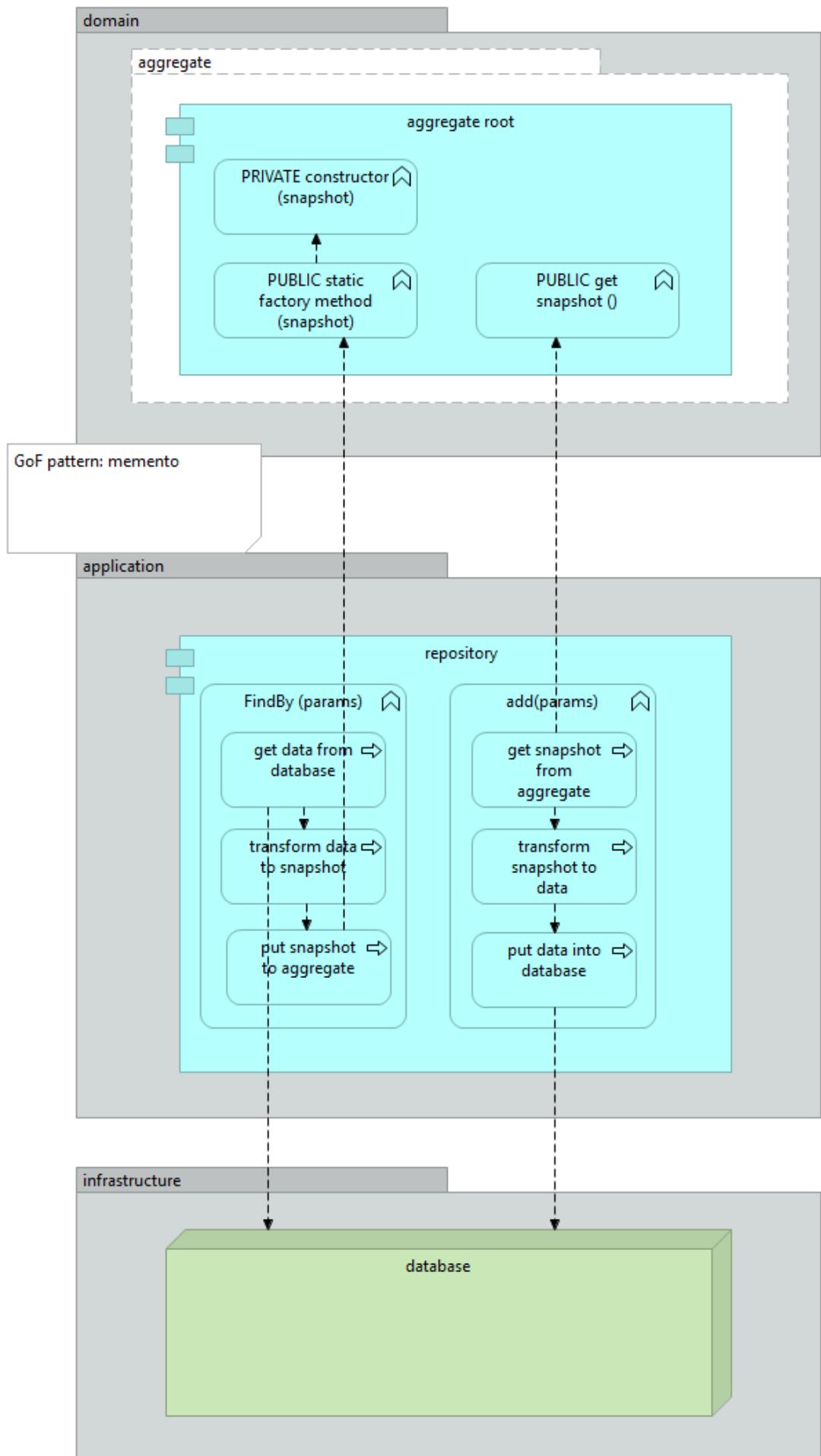
Must satisfy the invariants 2 after creation

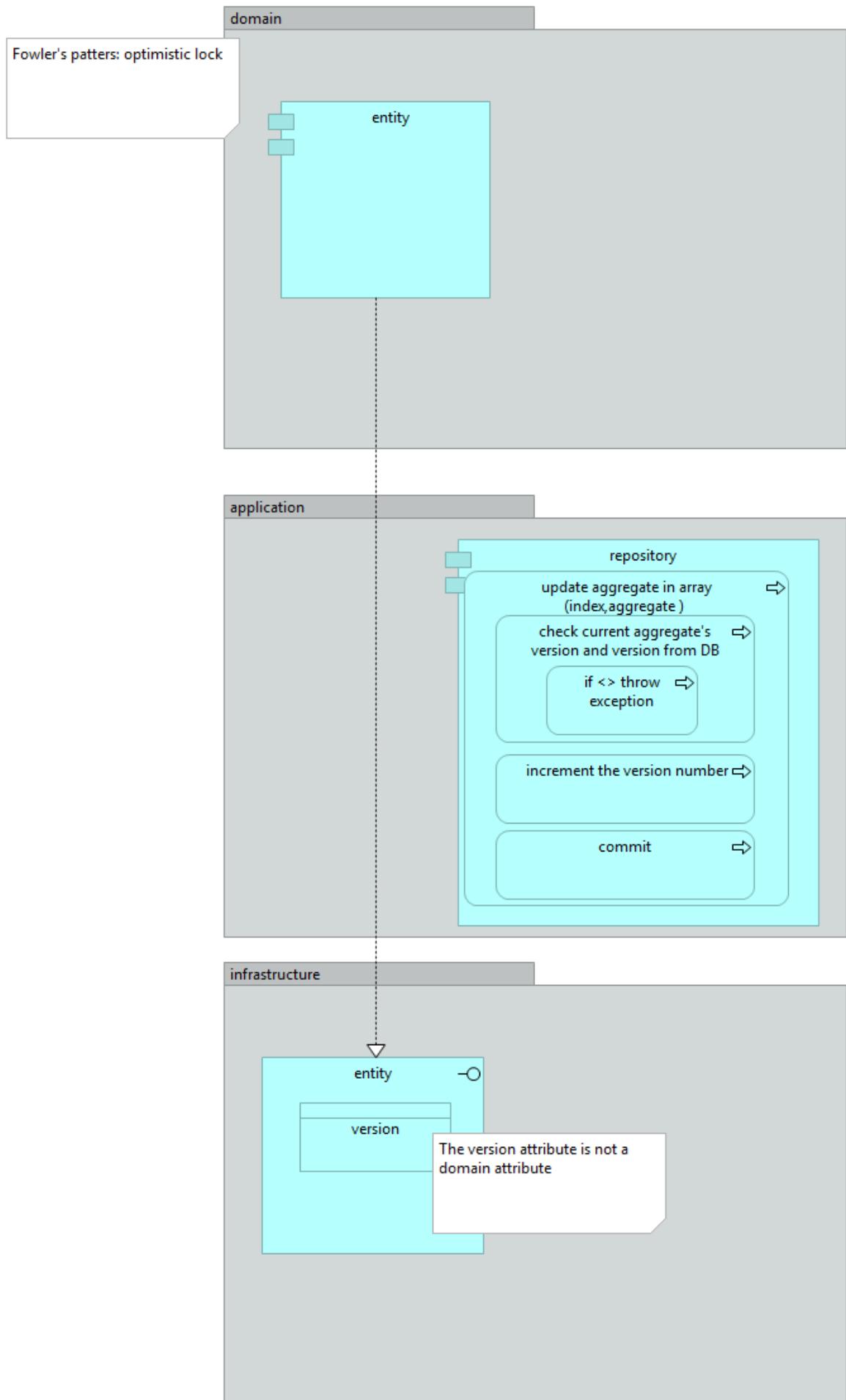


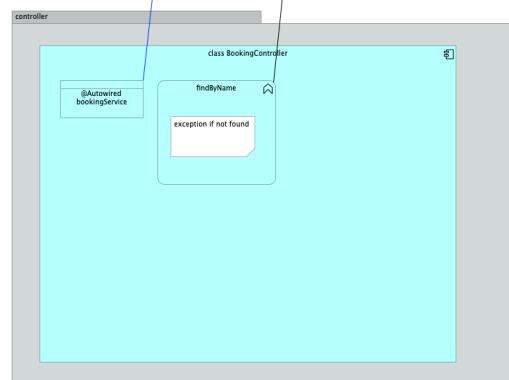
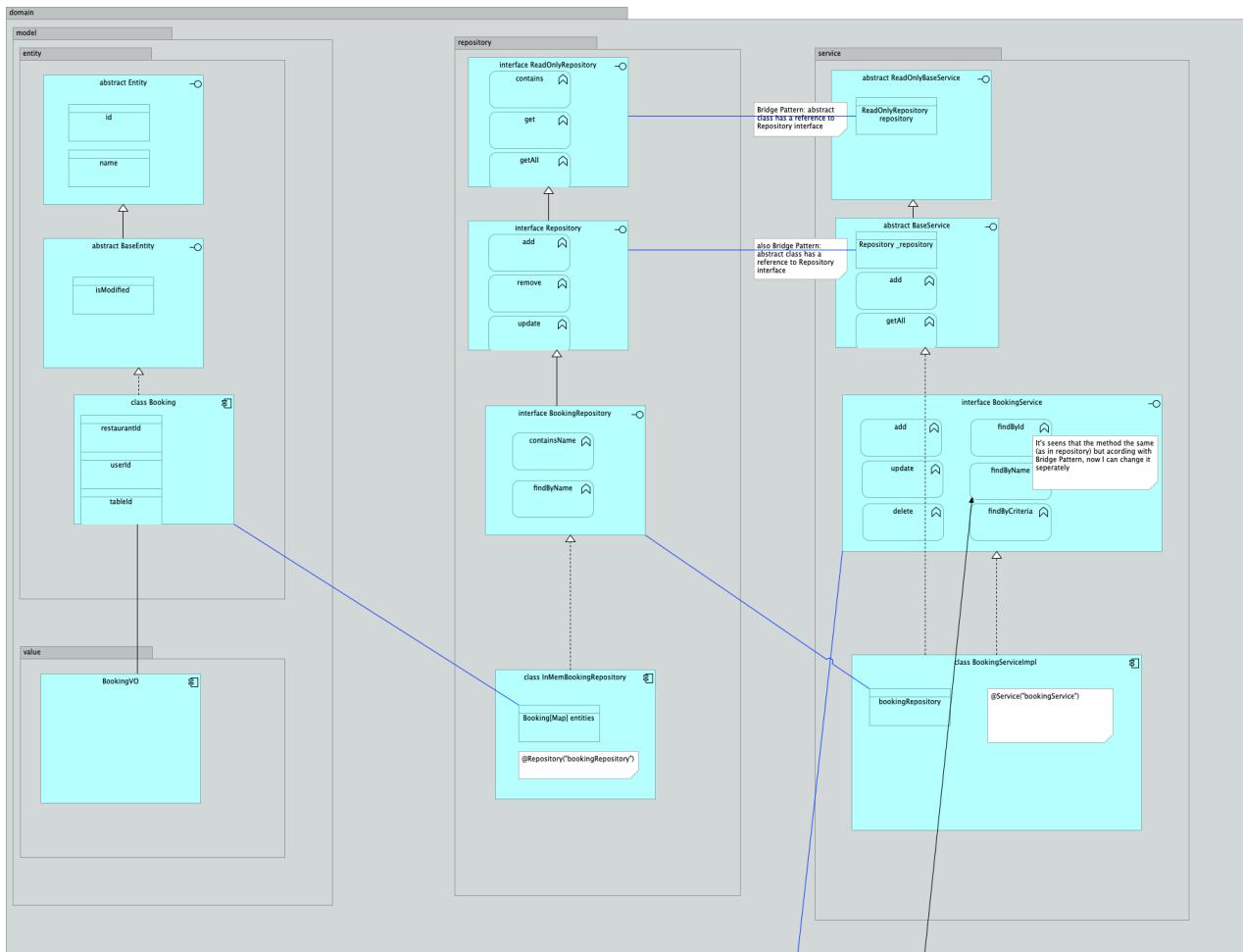


REPOSITORIES



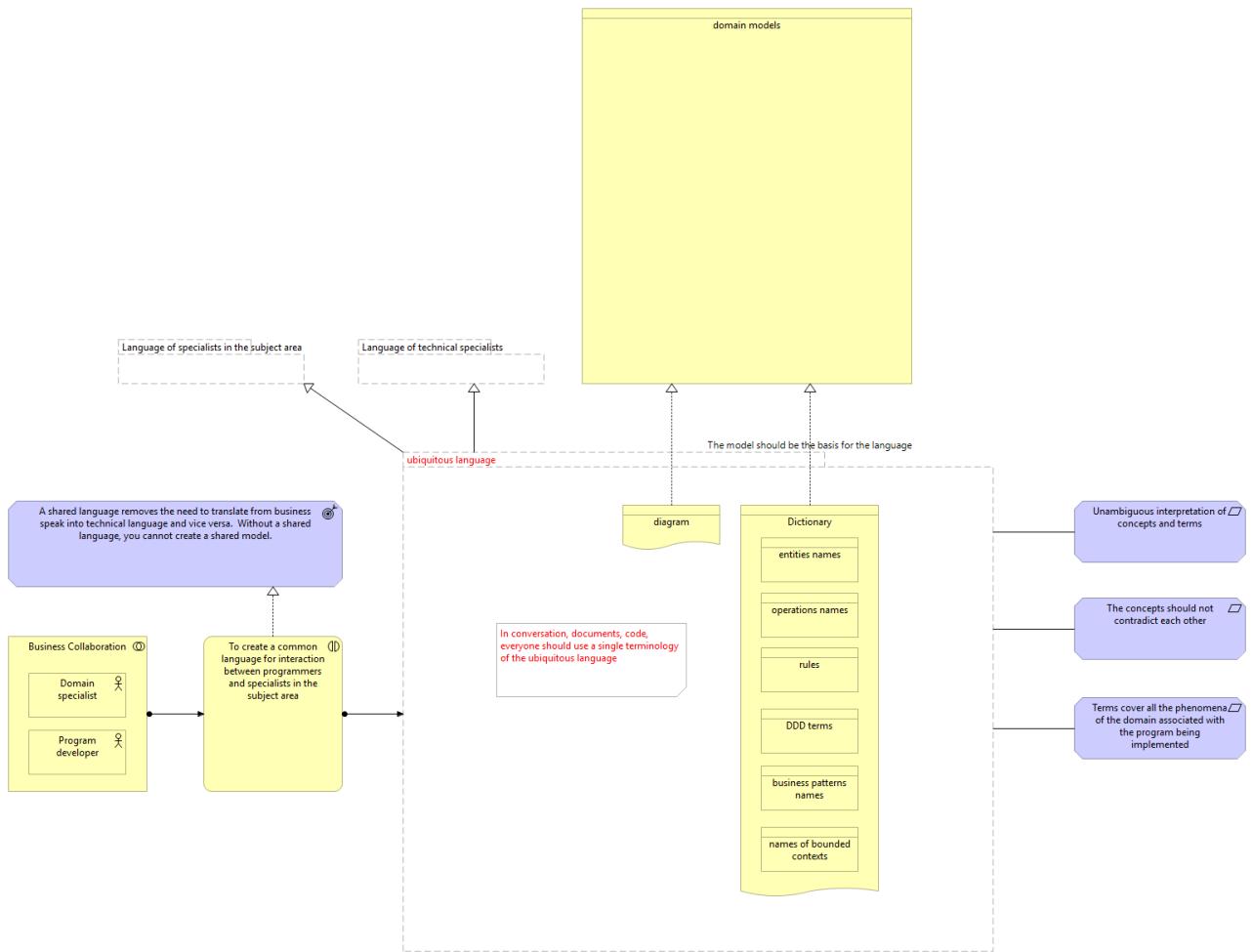




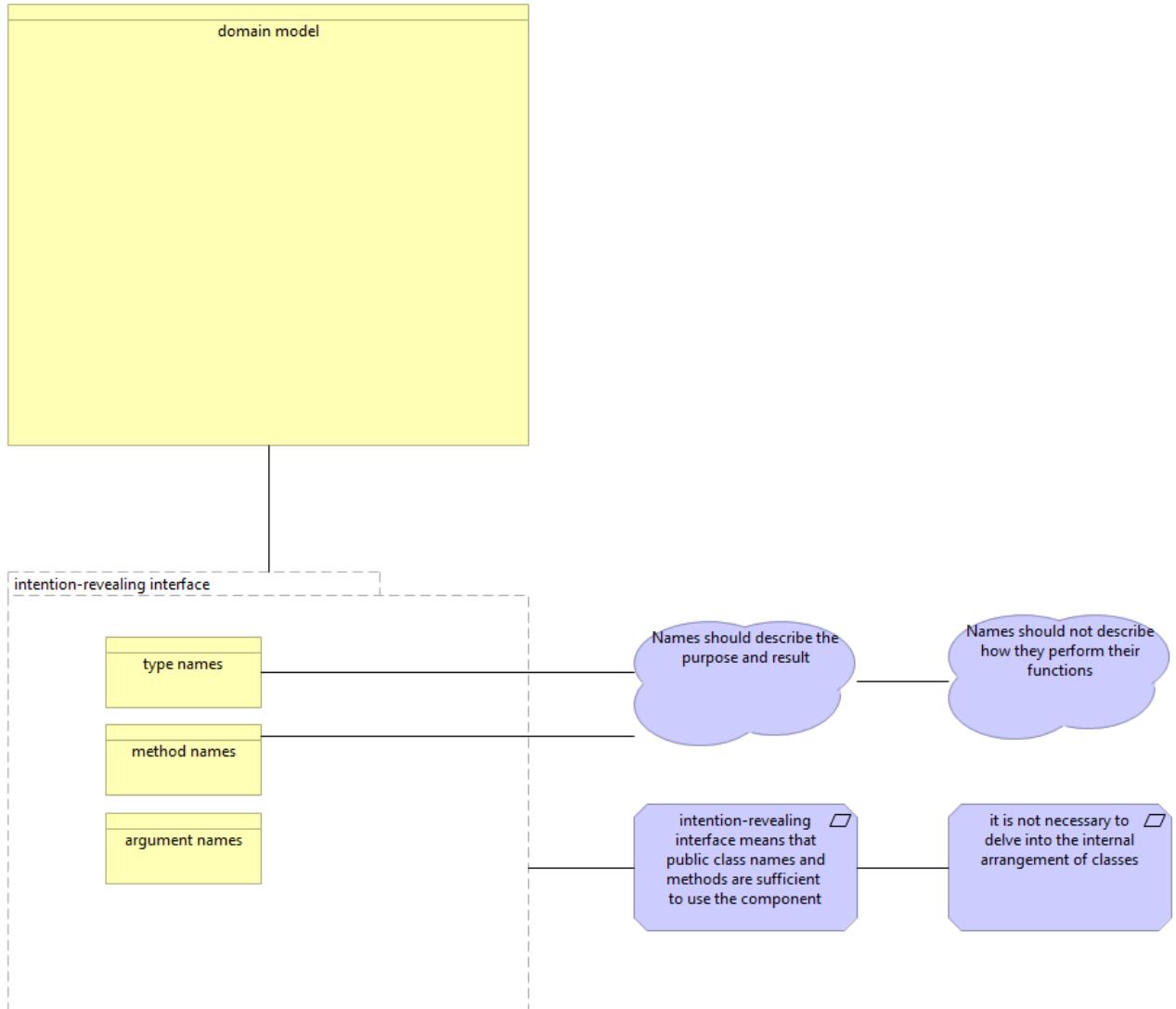


SUPPLE DESIGN

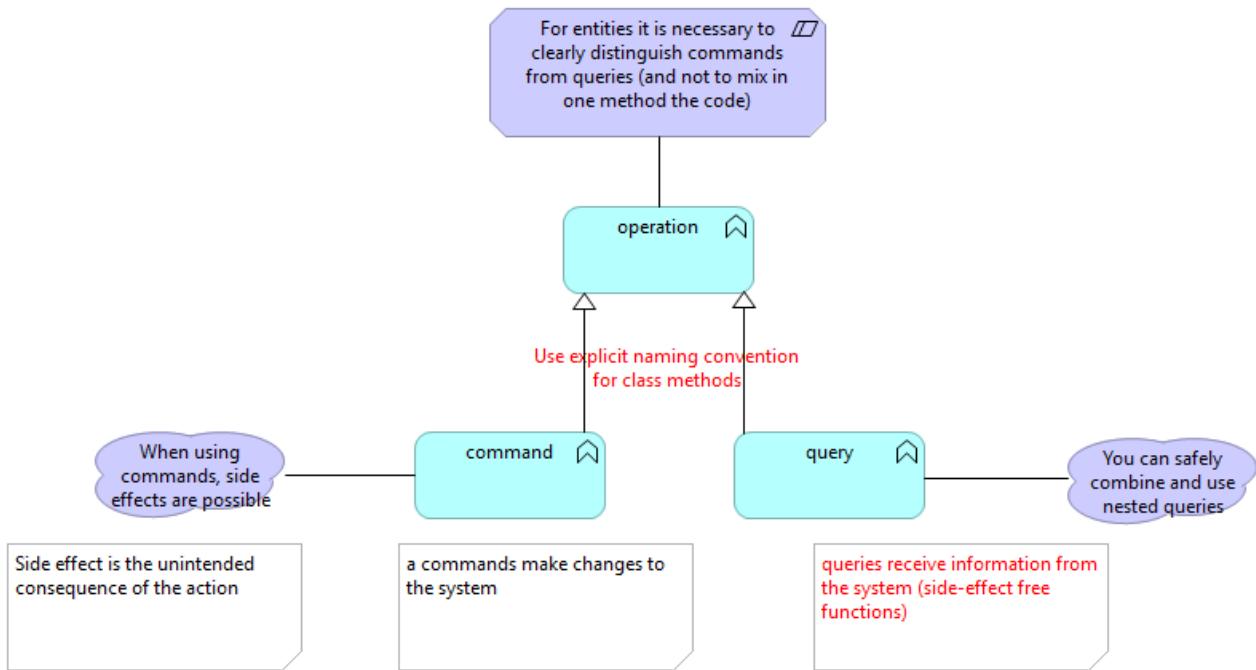
UBIQUITOUS LANGUAGE



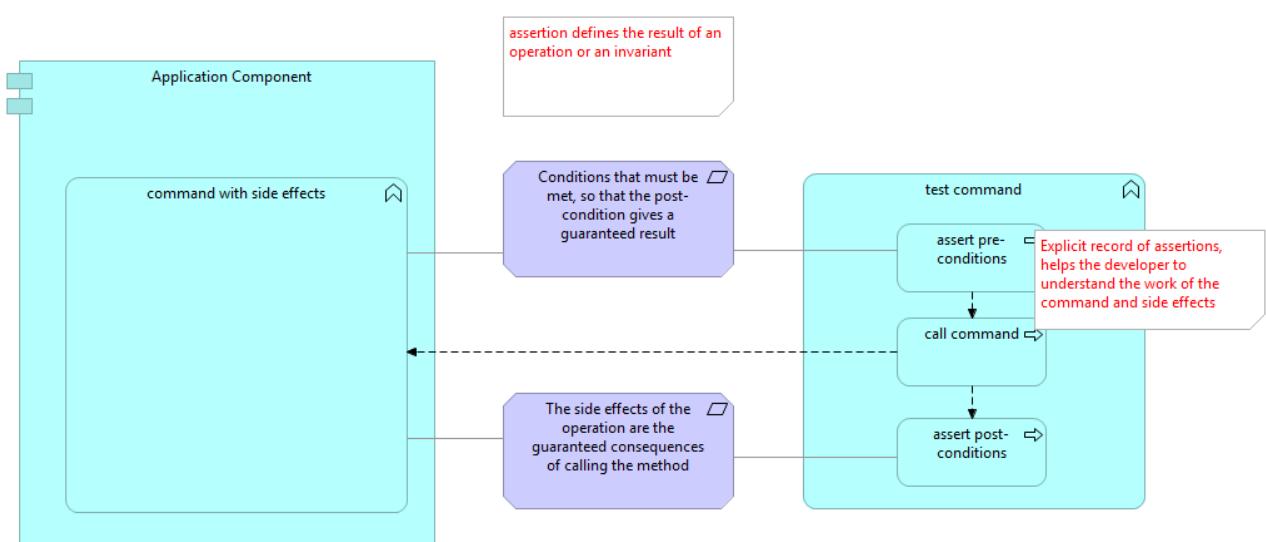
INTENTION-REVEALING INTERFACES



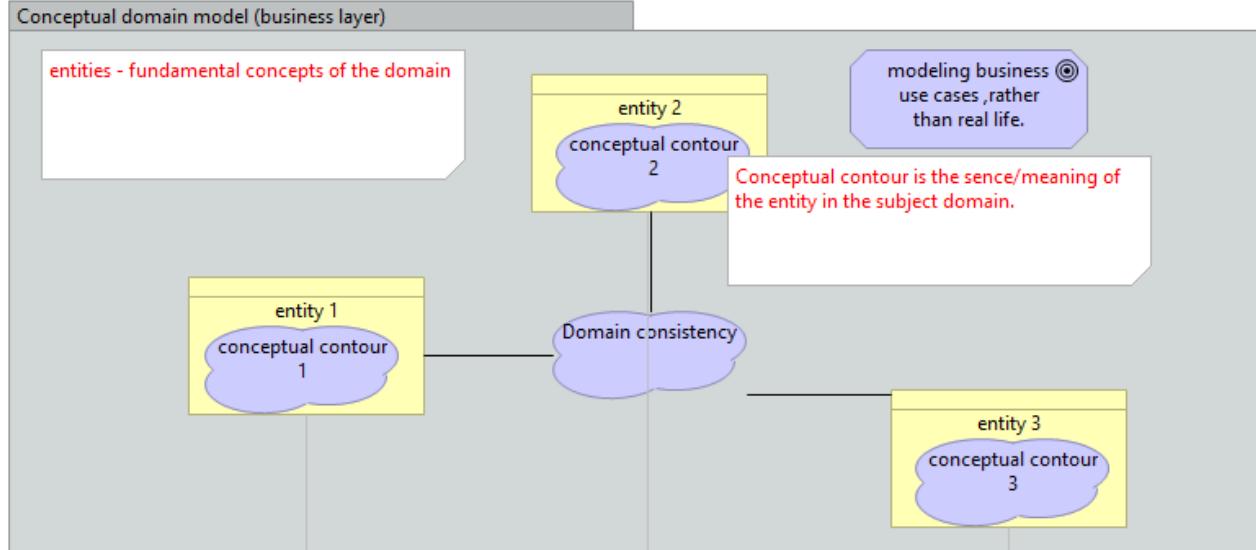
SIDE-EFFECT FREE FUNCTIONS



ASSERTIONS

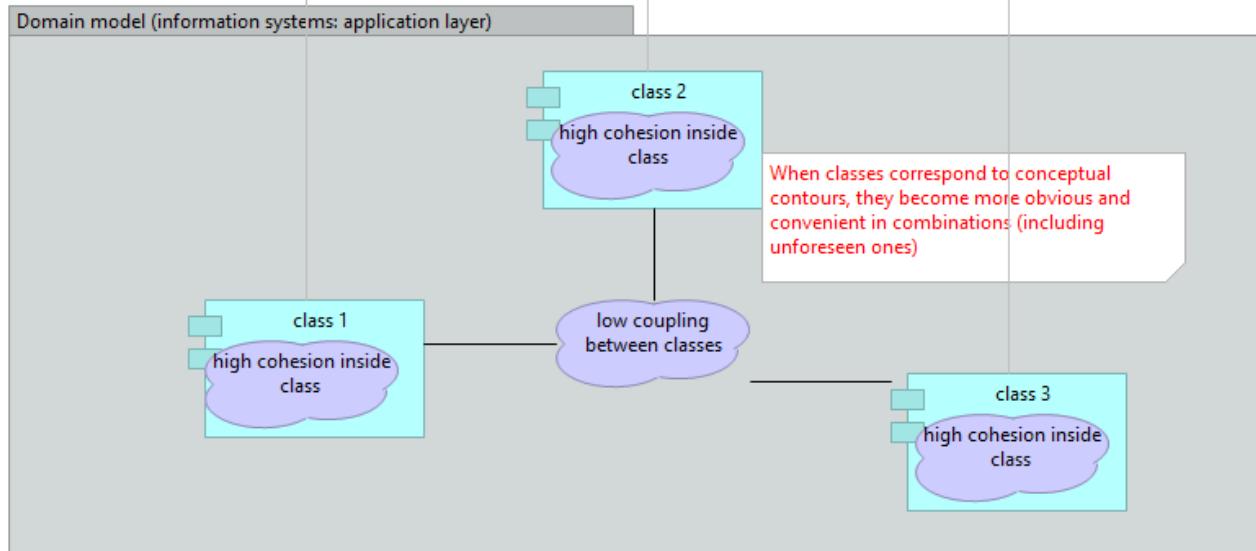


CONCEPTUAL CONTOURS

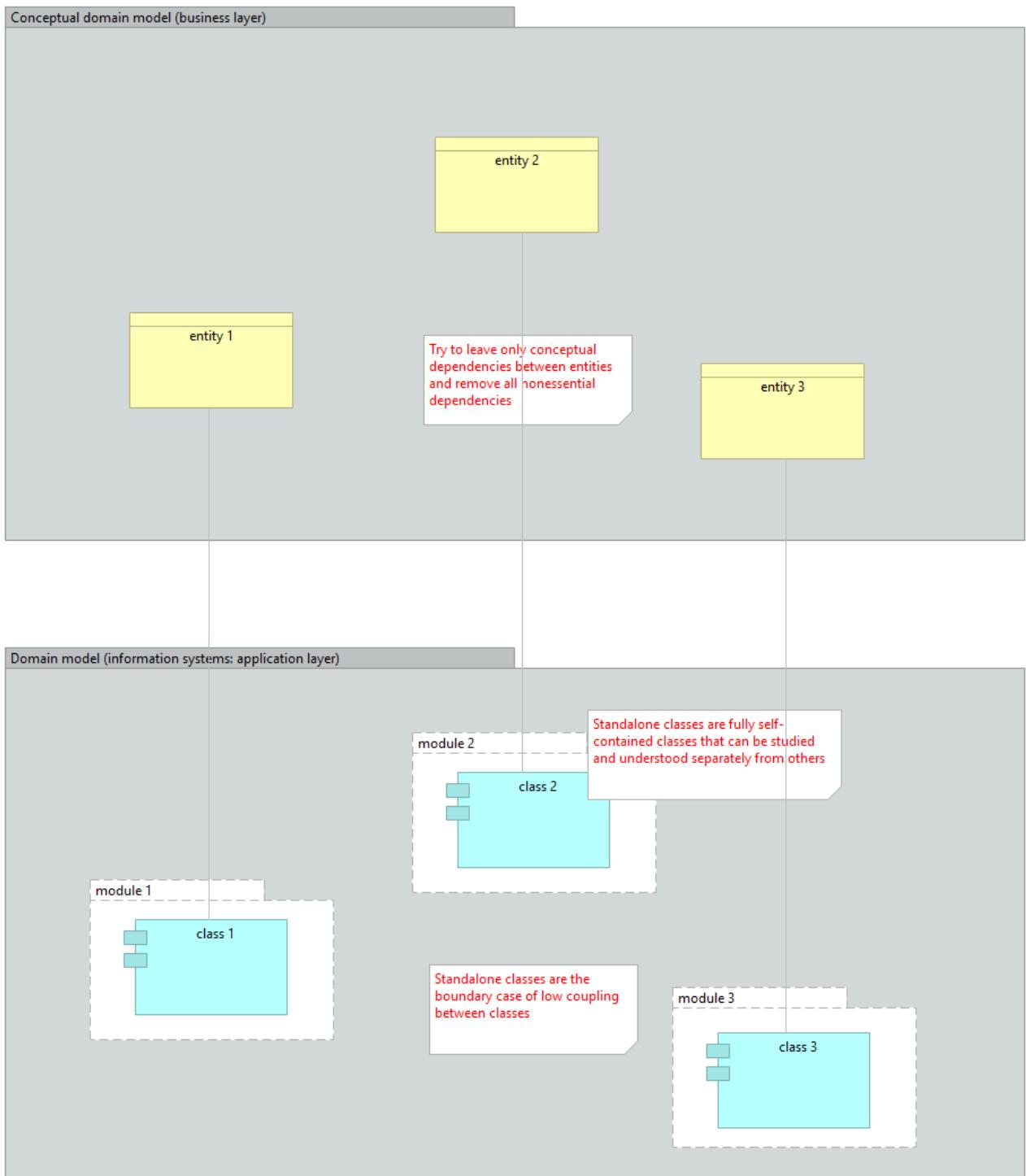


Classes are a reflection of the conceptual/semantic contours of the domain

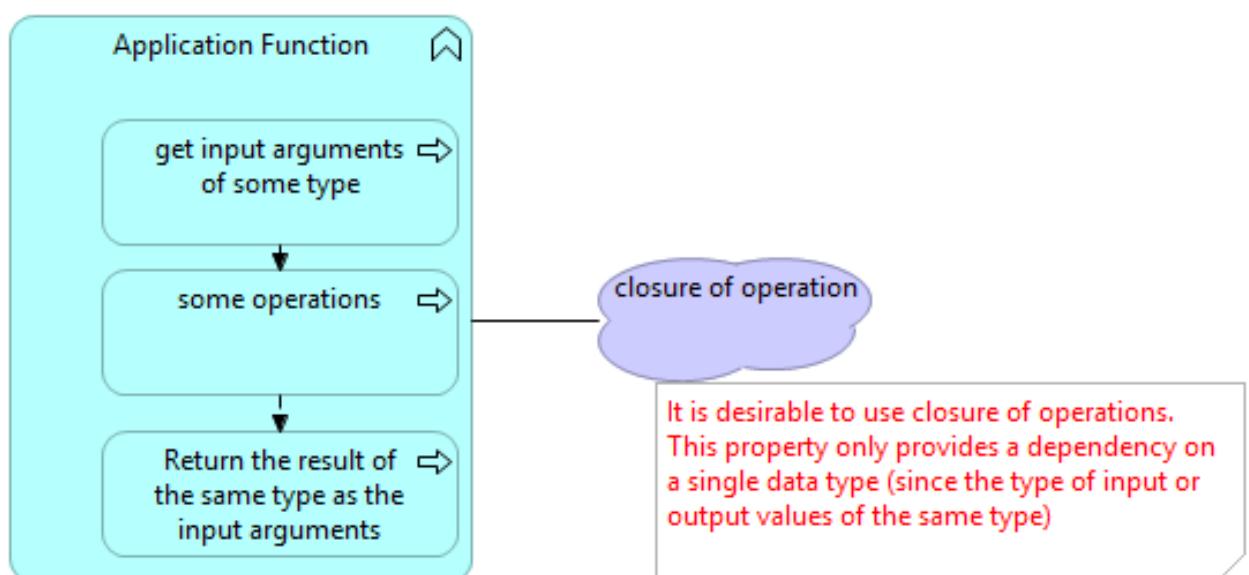
All entities of the domain and classes must correspond to each other and have meaning



STANDALONE CLASSES

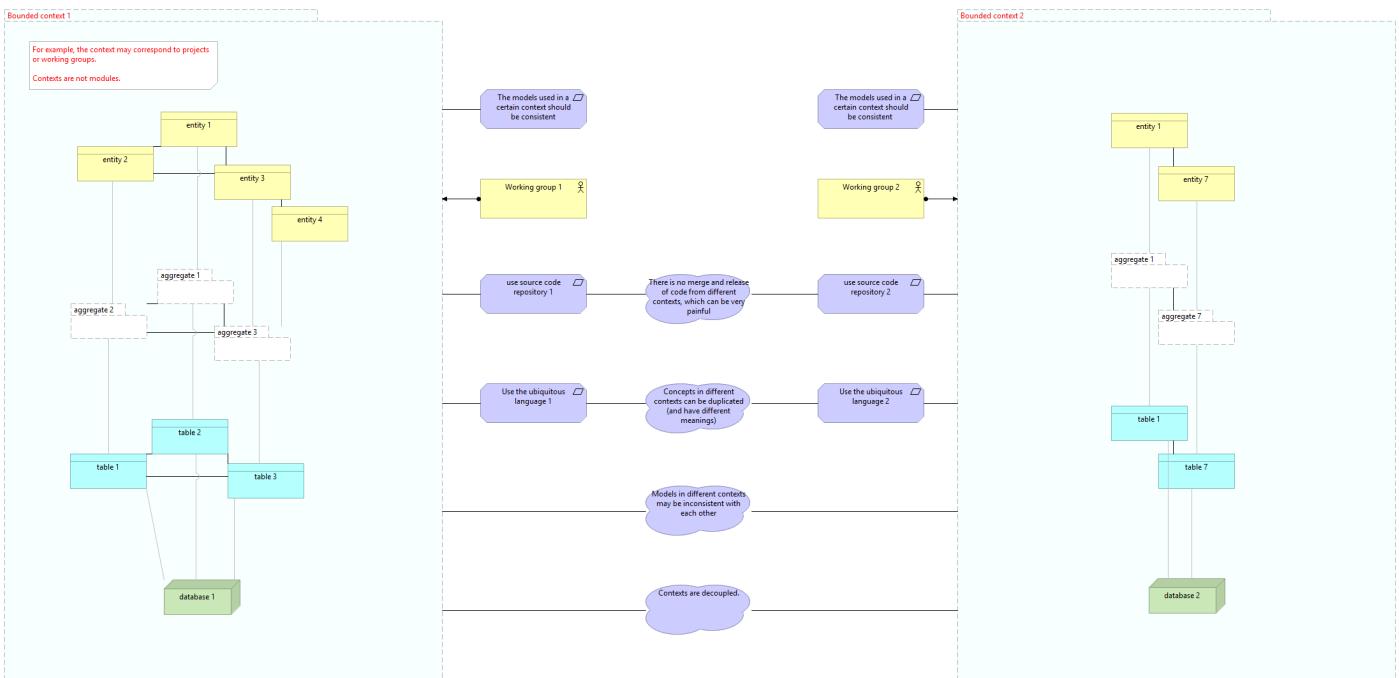


CLOSURE OF OPERATIONS

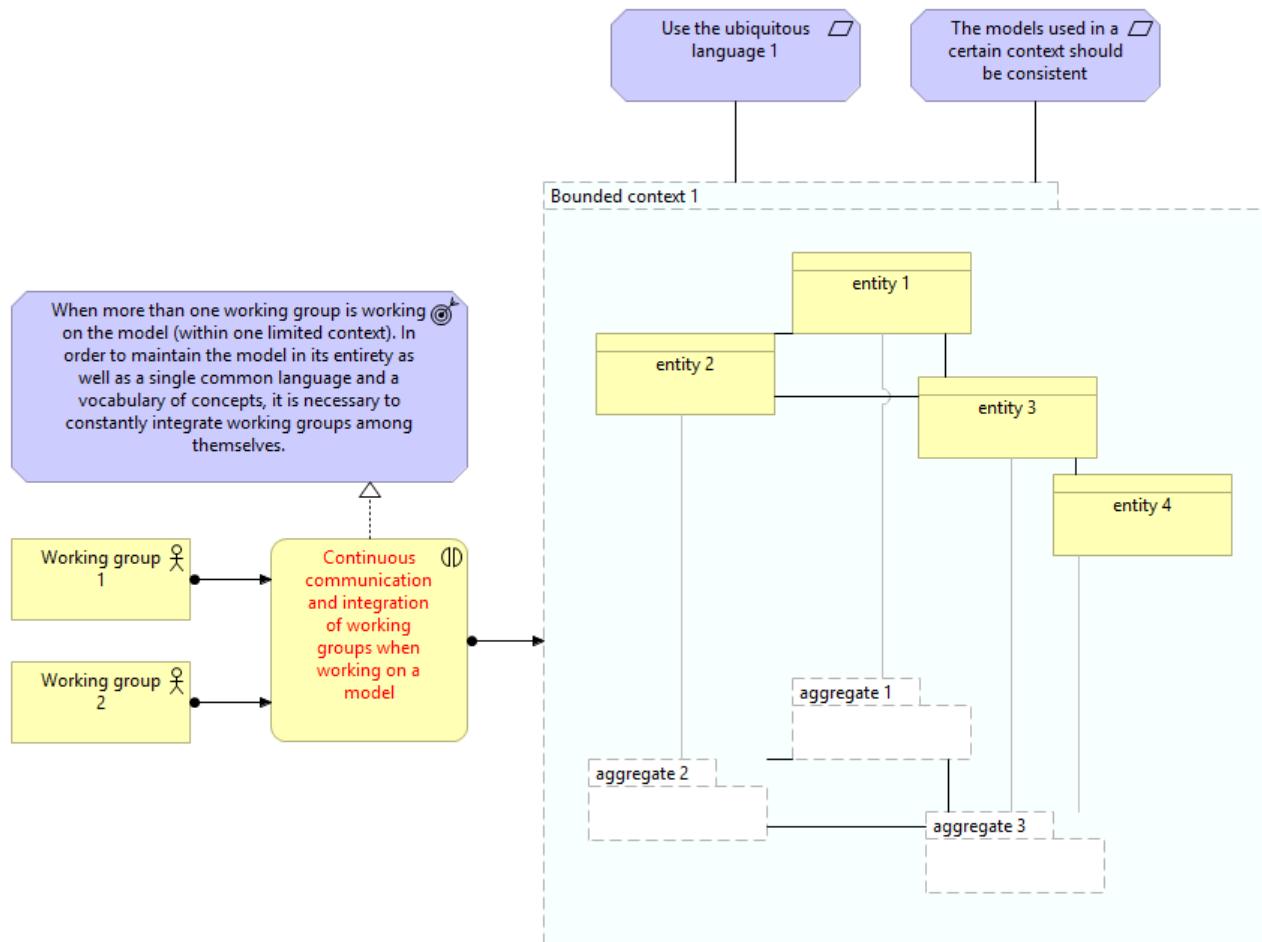


MODEL INTEGRITY AND CONTEXT

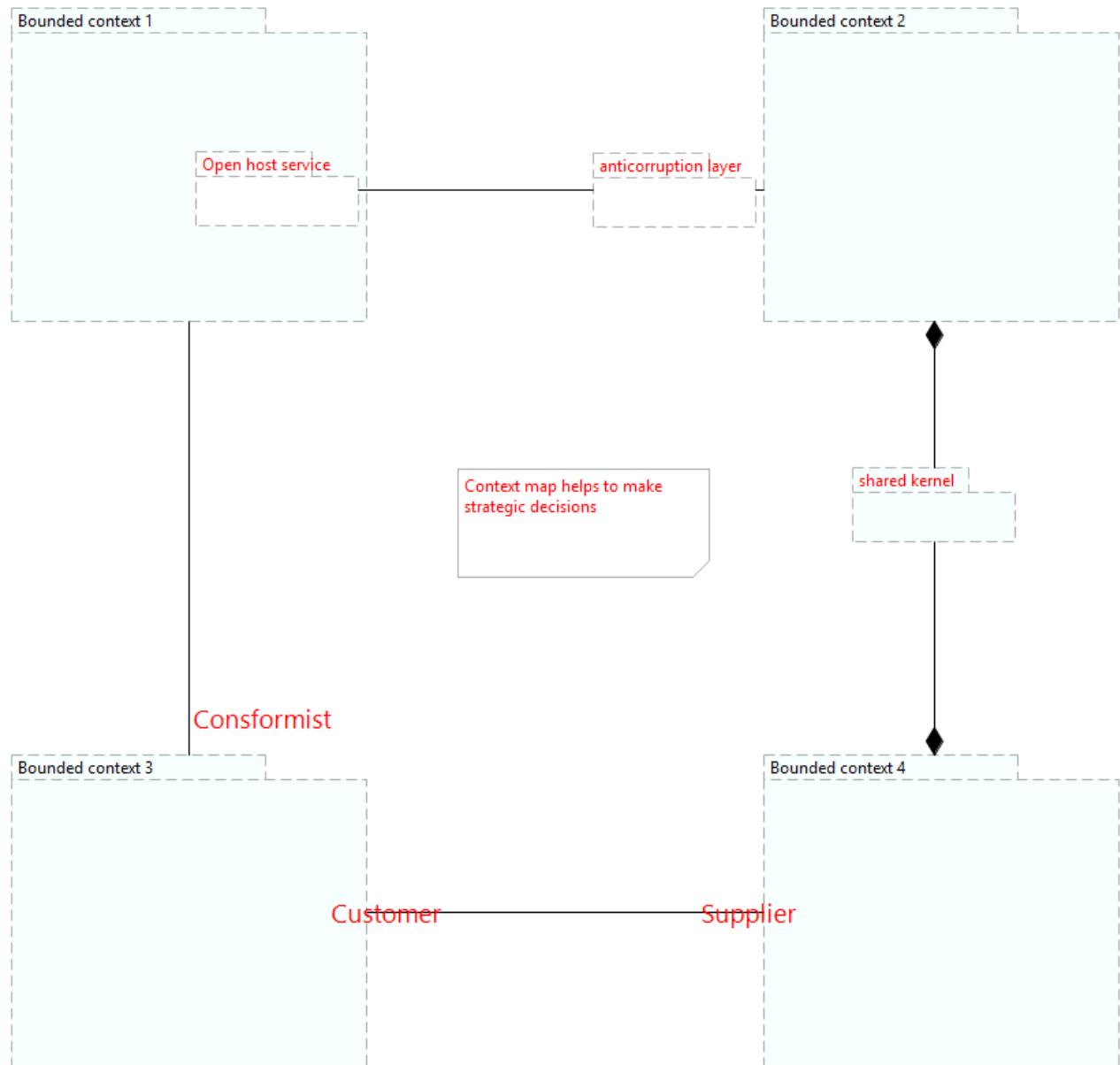
BOUNDED CONTEXT



CONTINUOUS INTEGRATION

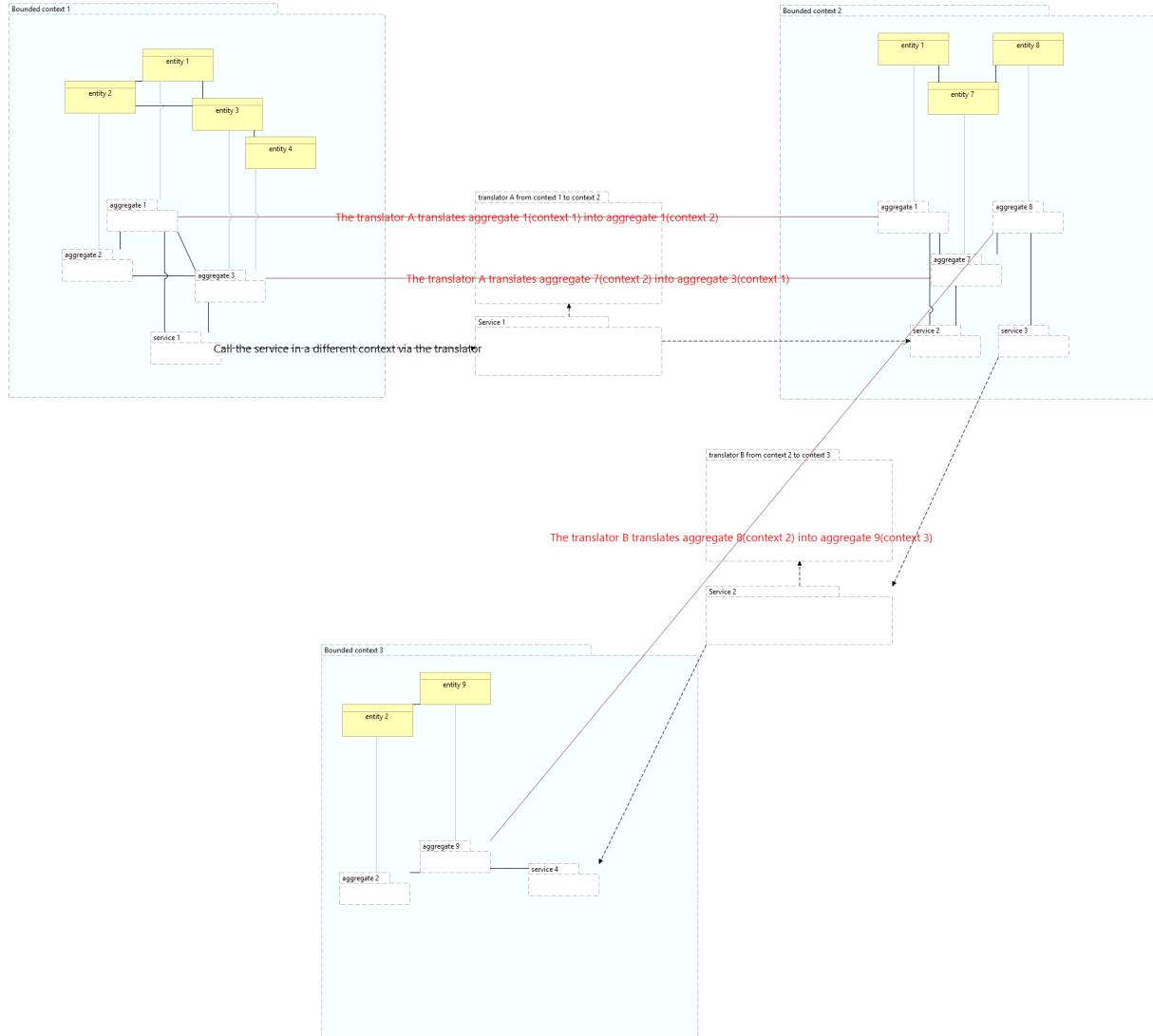


STRATEGIC CONTEXT MAP

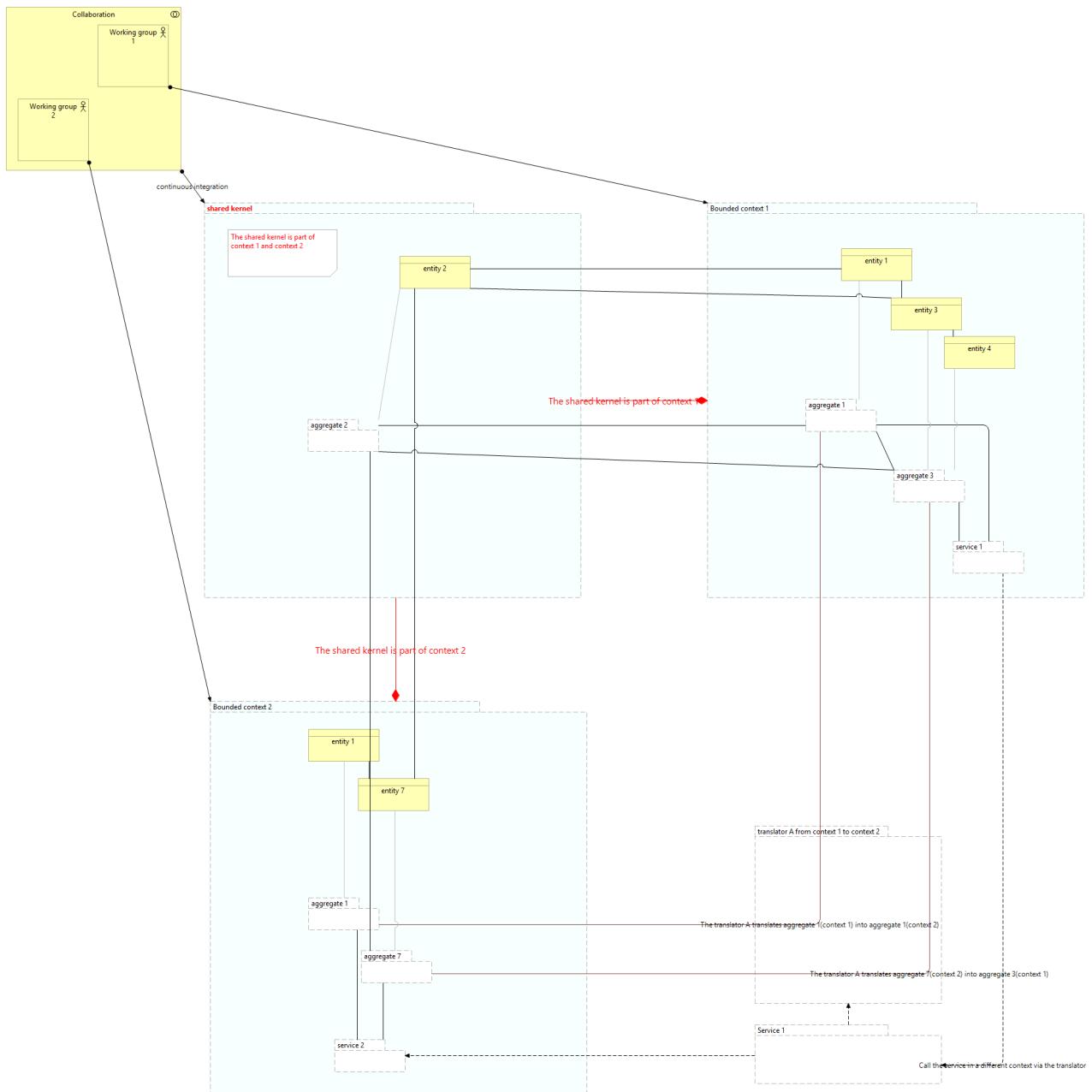


CONTEXTUAL MAP

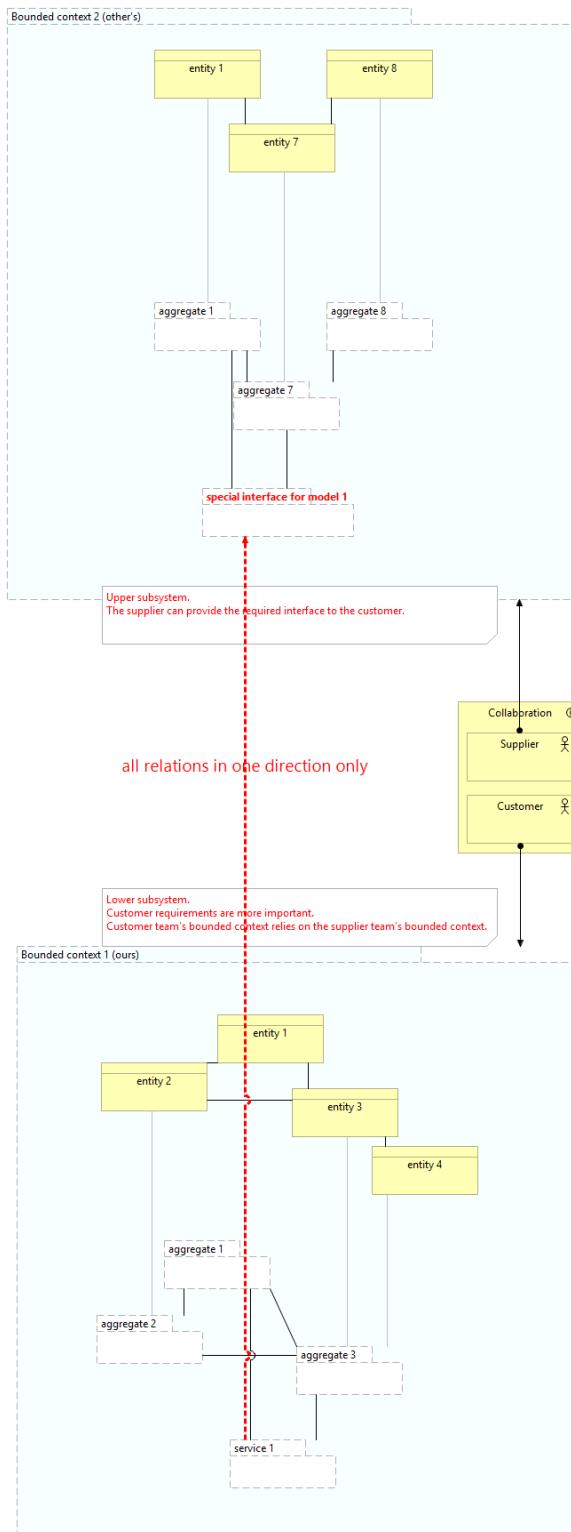
A single map of all contexts.
Integration of all bounded contexts.



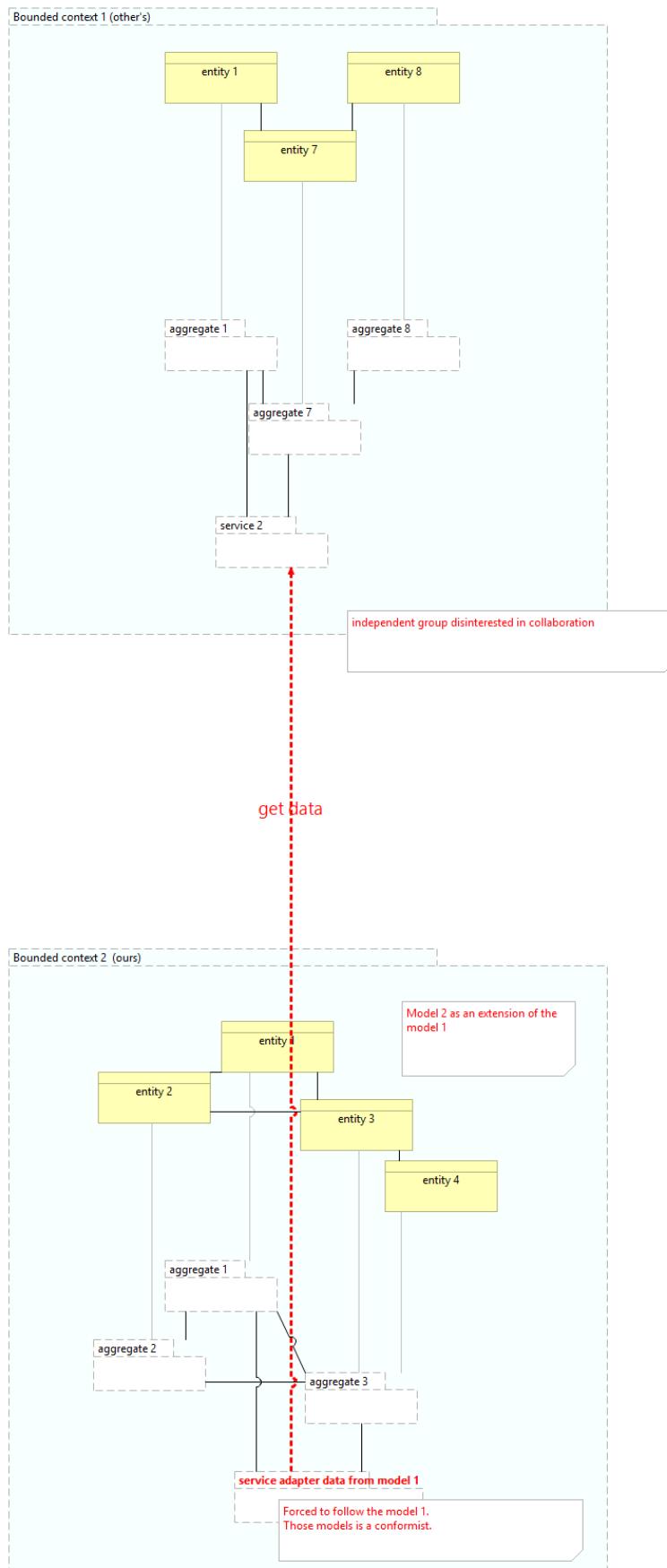
SHARED KERNEL



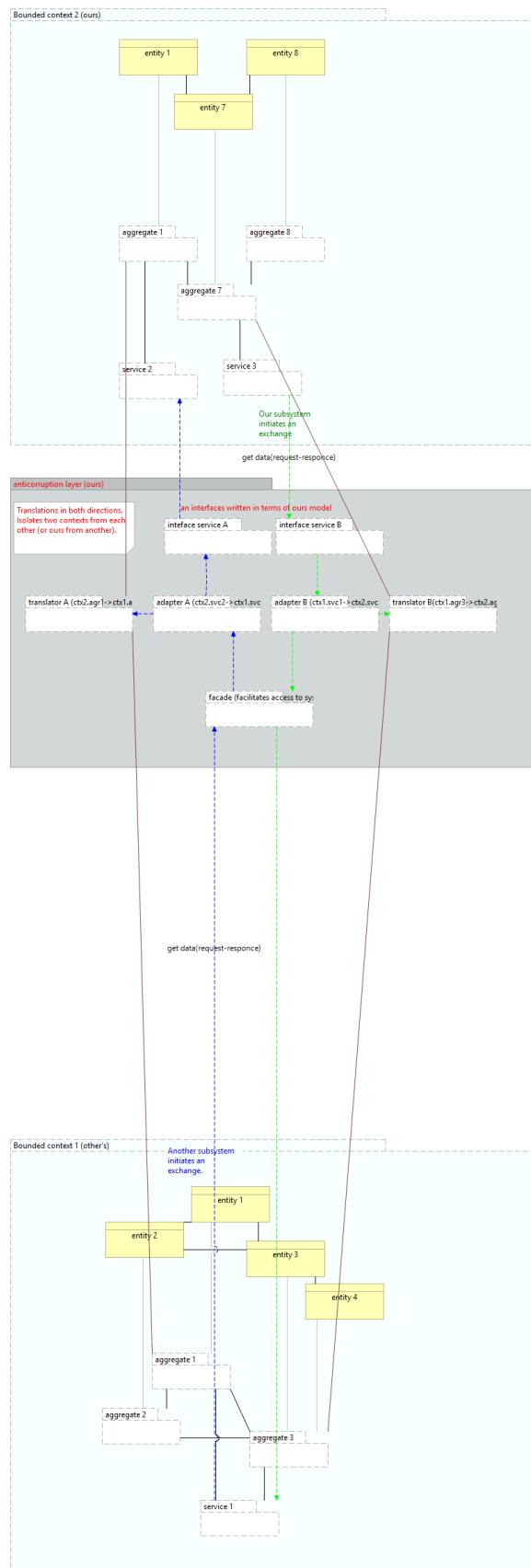
CUSTOMER-SUPPLIER TEAMS



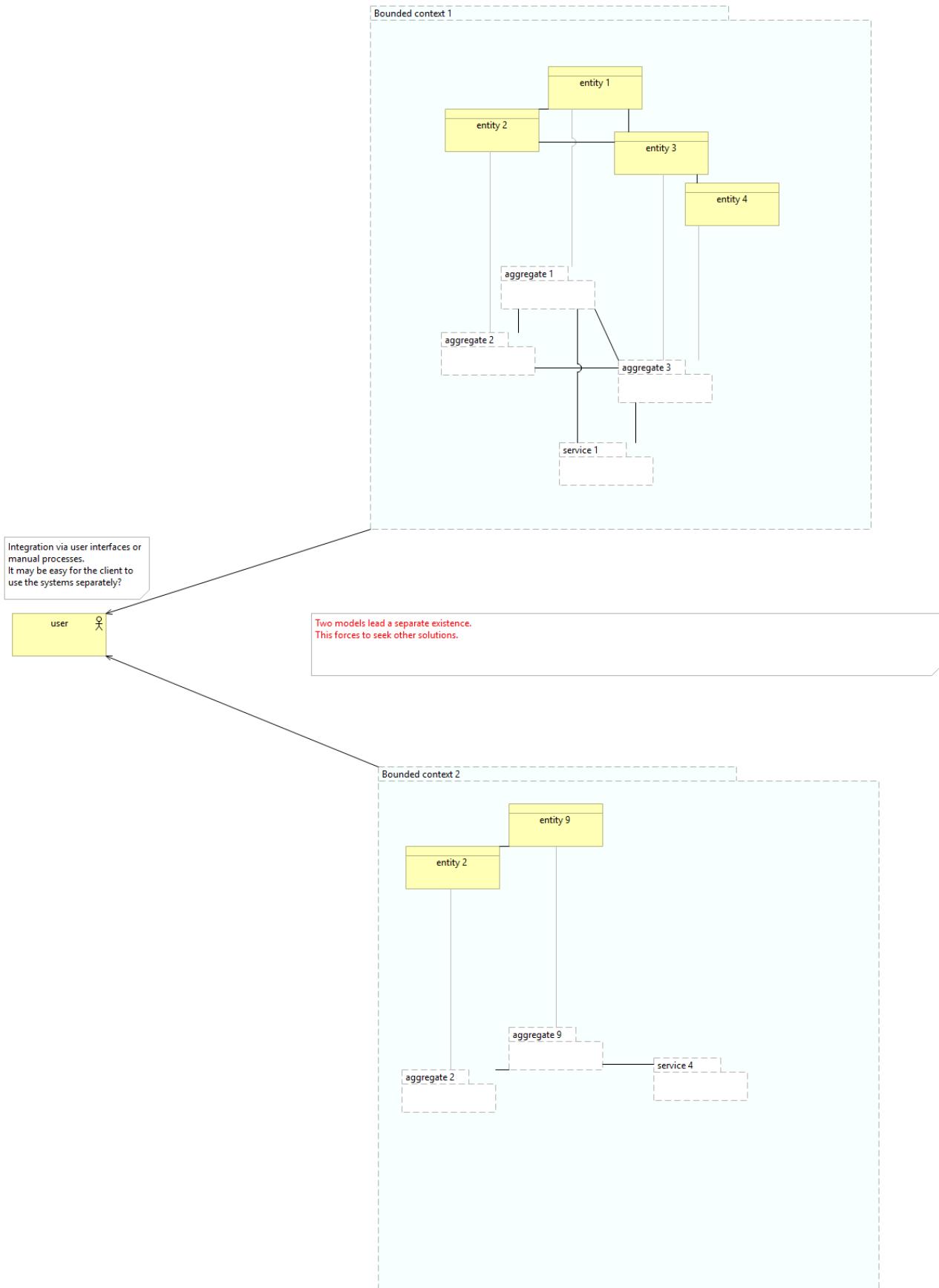
CONFORMIST



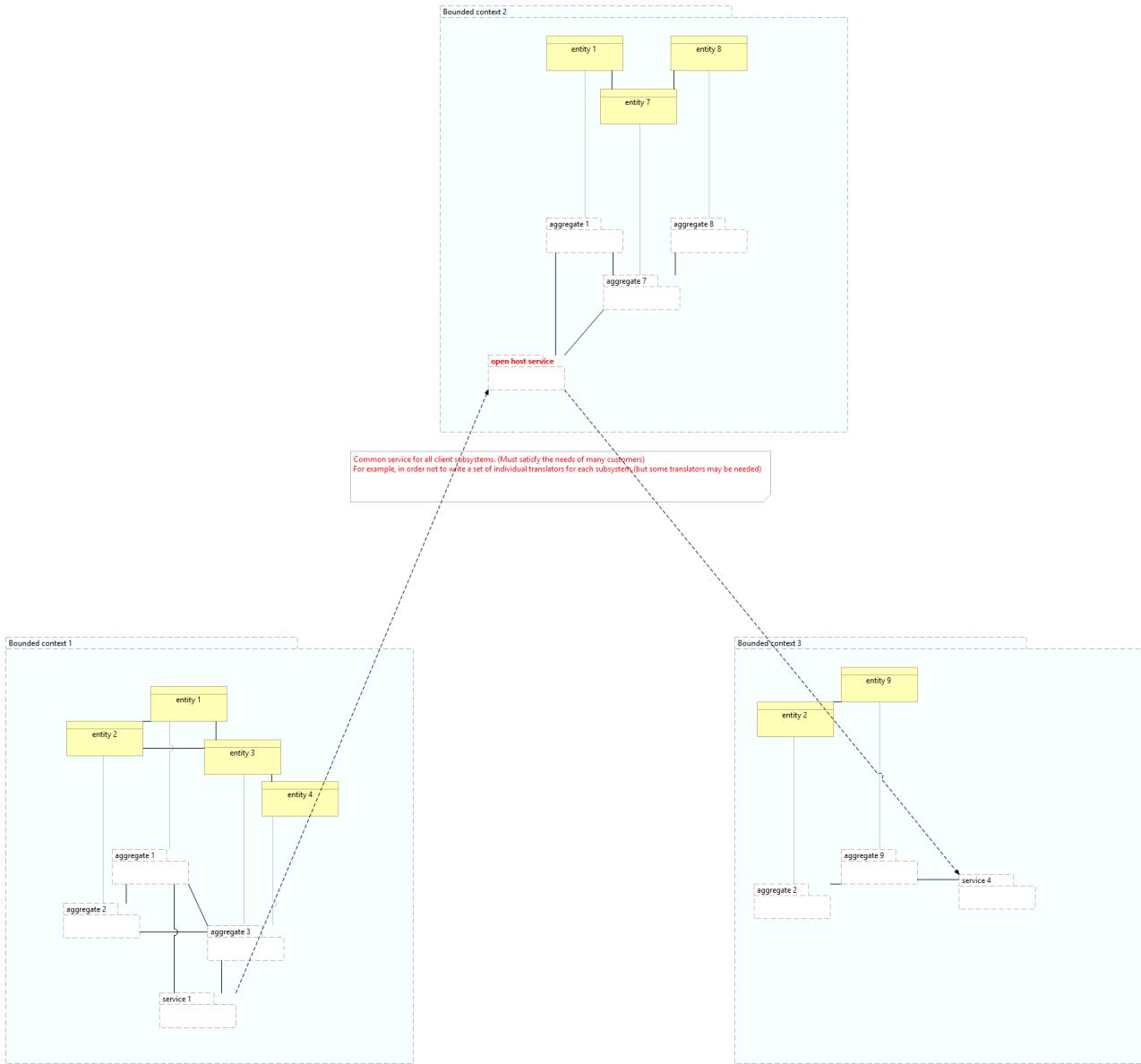
ANTICORRUPTION LAYER



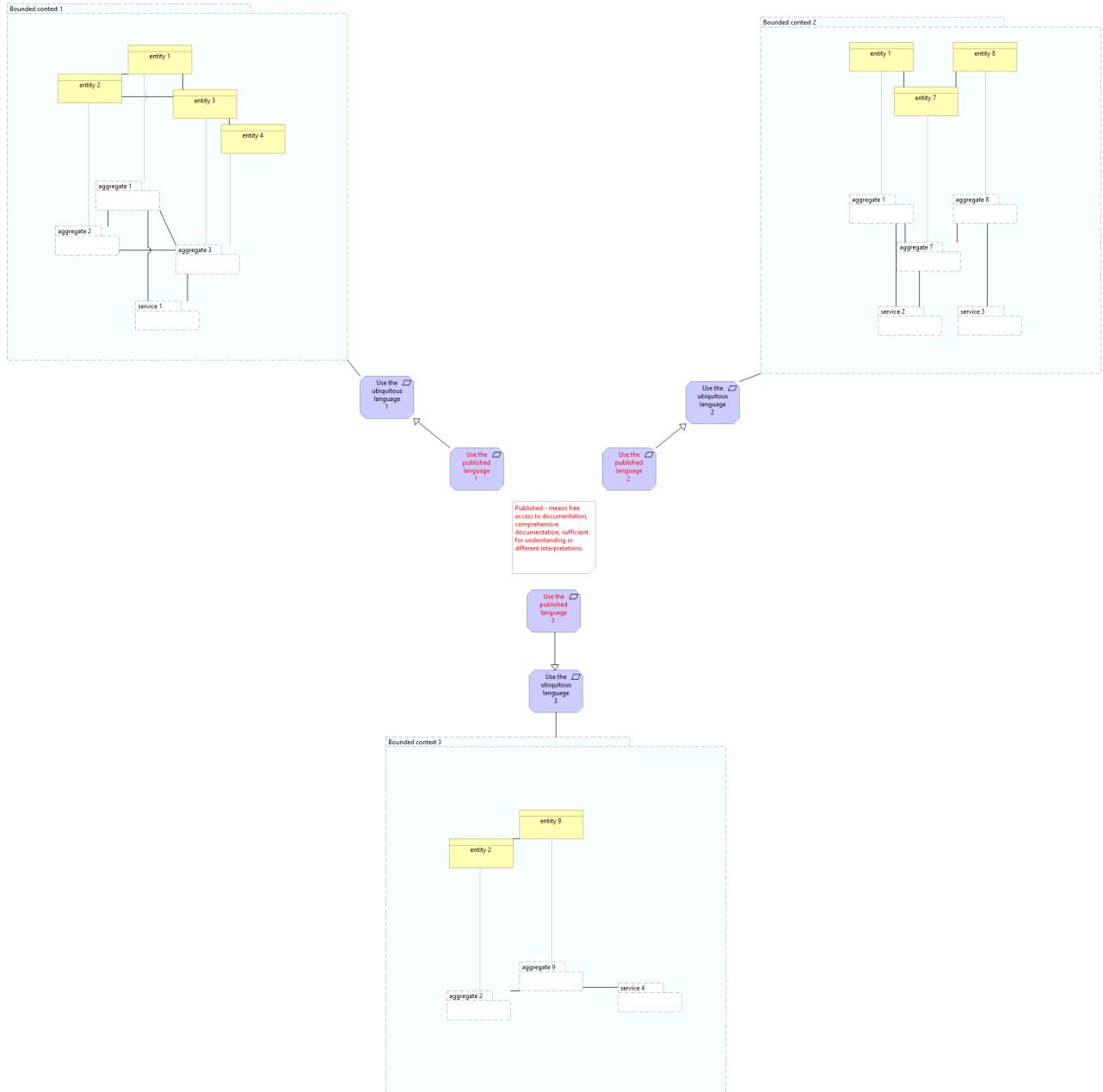
SEPARATE WAYS



OPEN HOST SERVICE

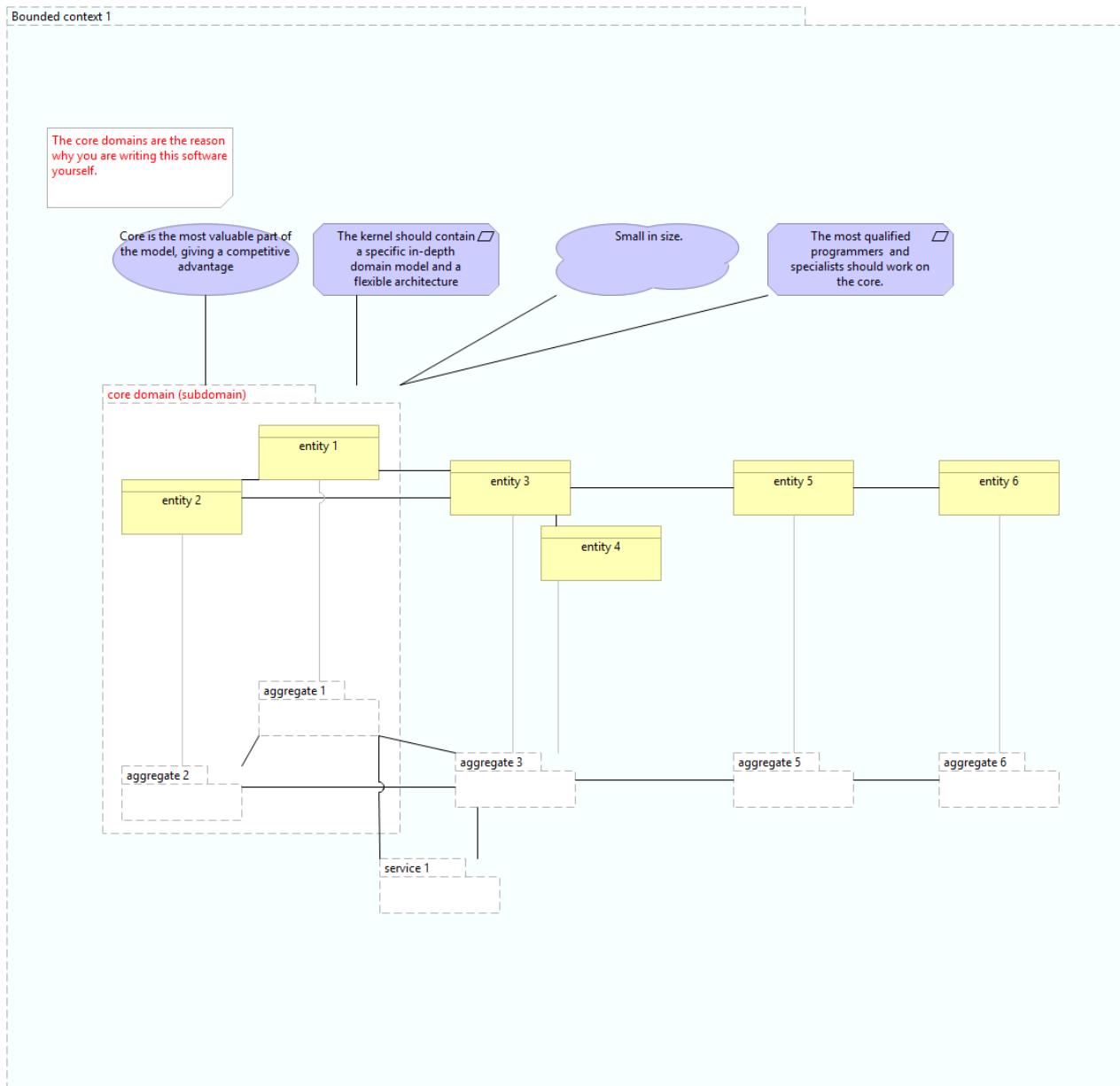


PUBLISHED LANGUAGE

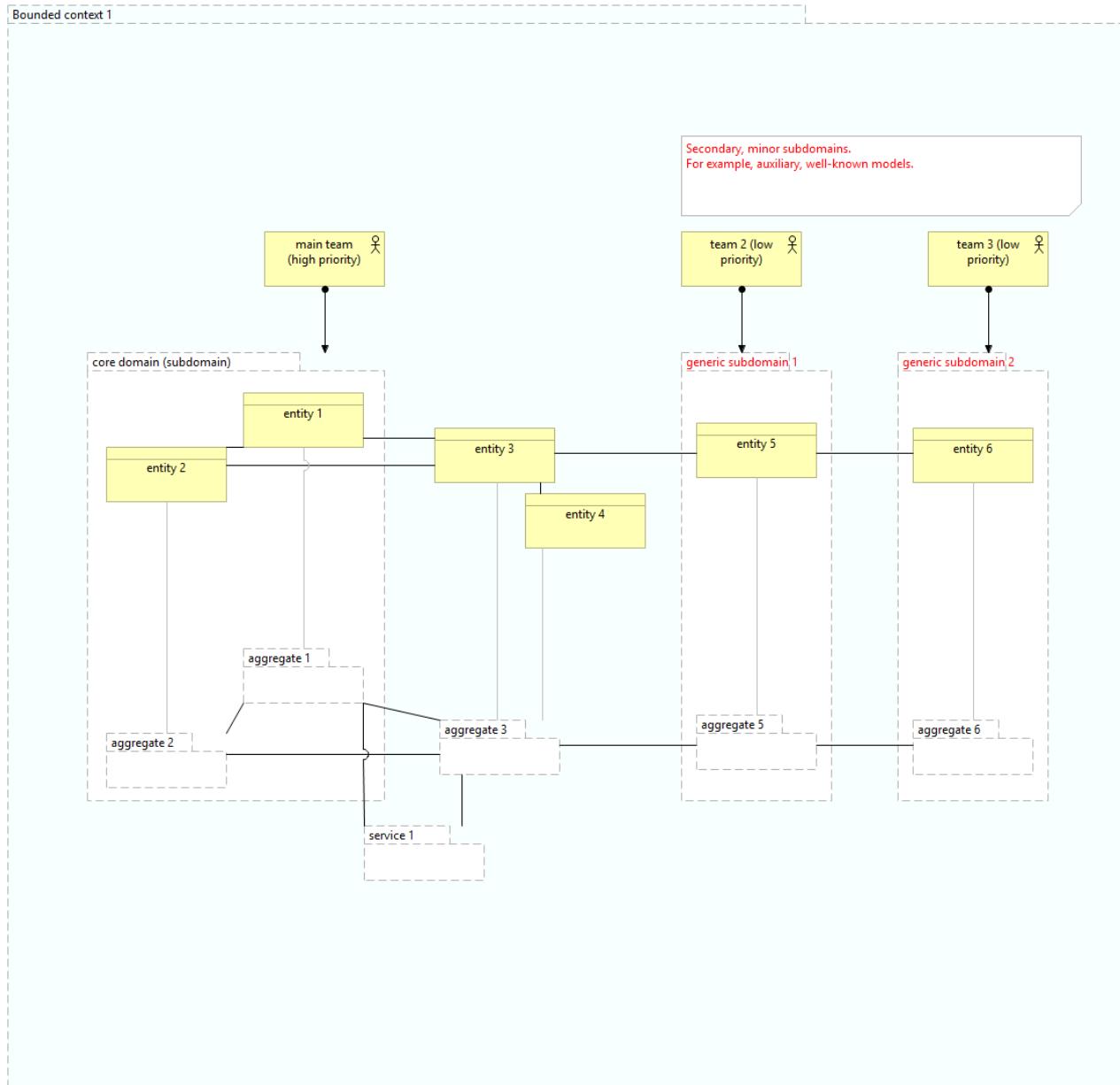


DISTILLATION

CORE DOMAIN



GENERIC SUBDOMAINS



DOMAIN VISION STATEMENT

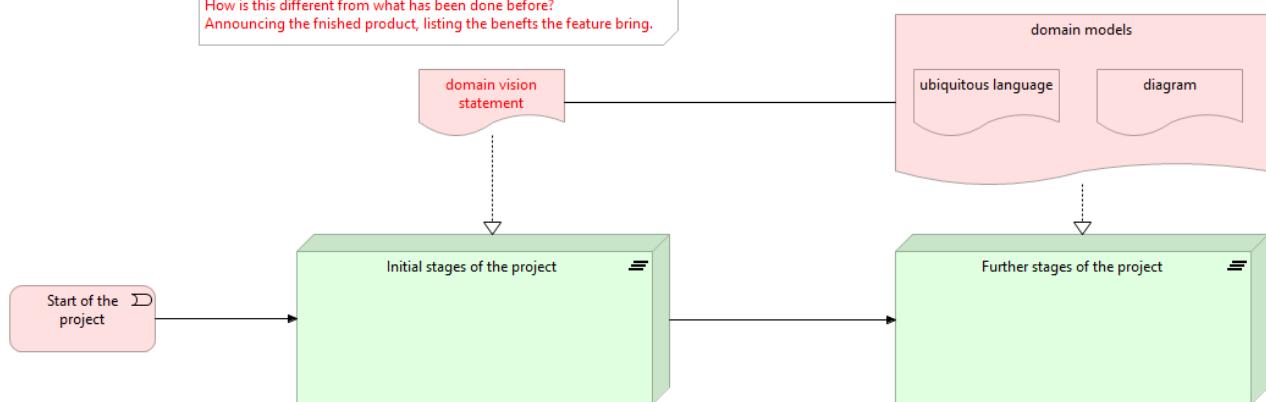
This document is a guideline that sets the direction before the development of the model.

Defines the core domain in the most general terms.

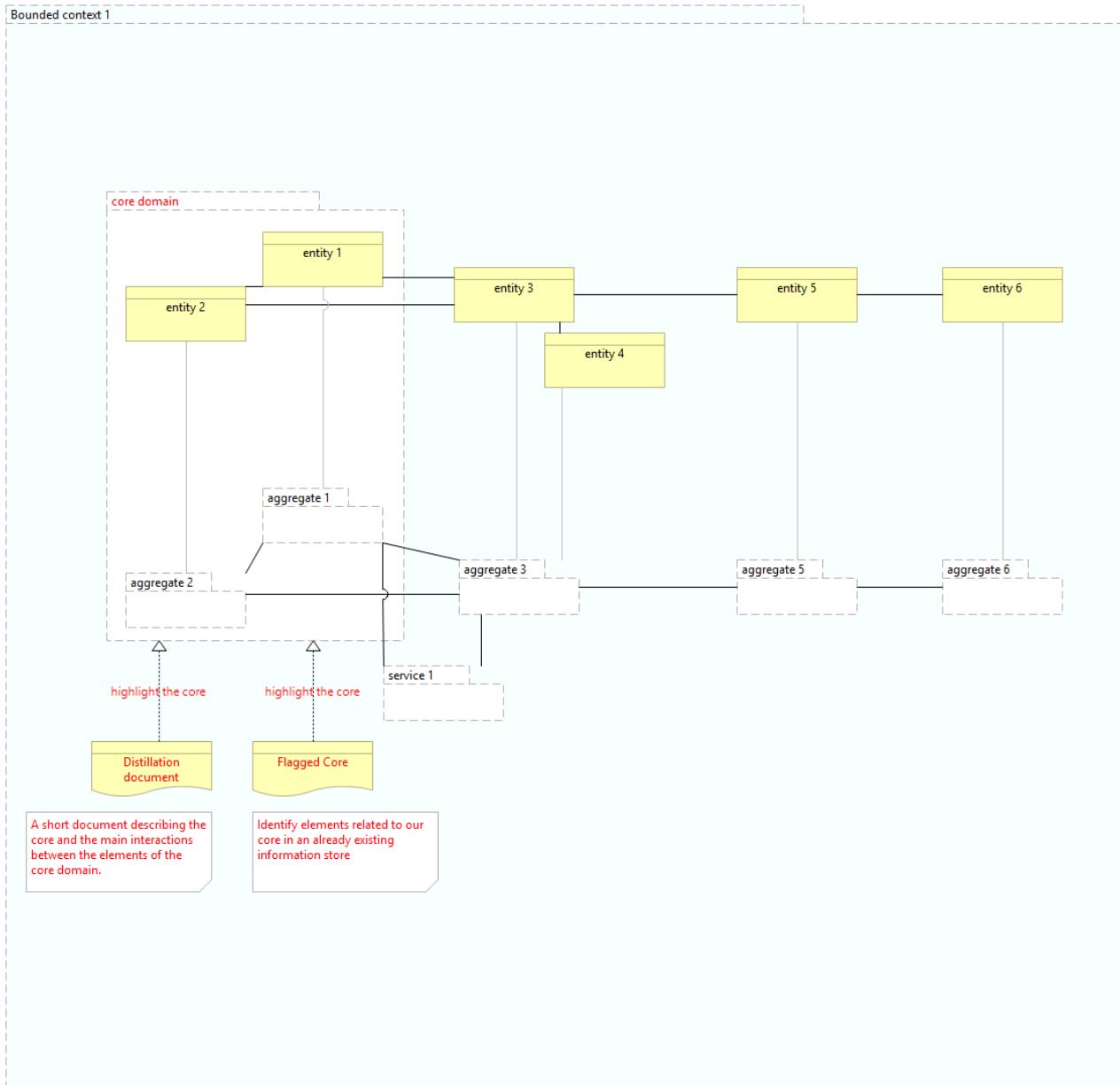
Draw out the reason you are opting to build rather than buy a product.

How is this different from what has been done before?

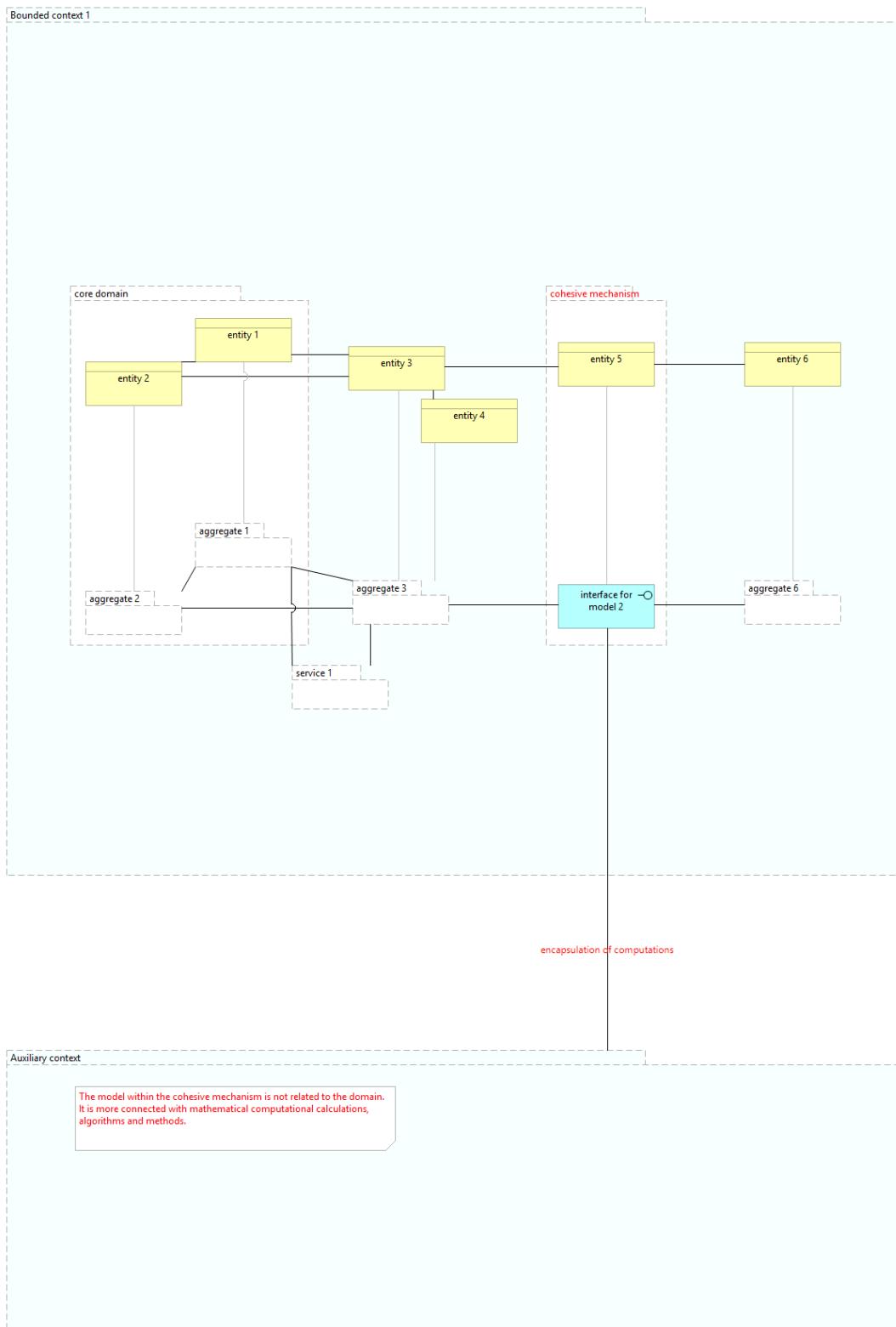
Announcing the finished product, listing the benefits the feature bring.



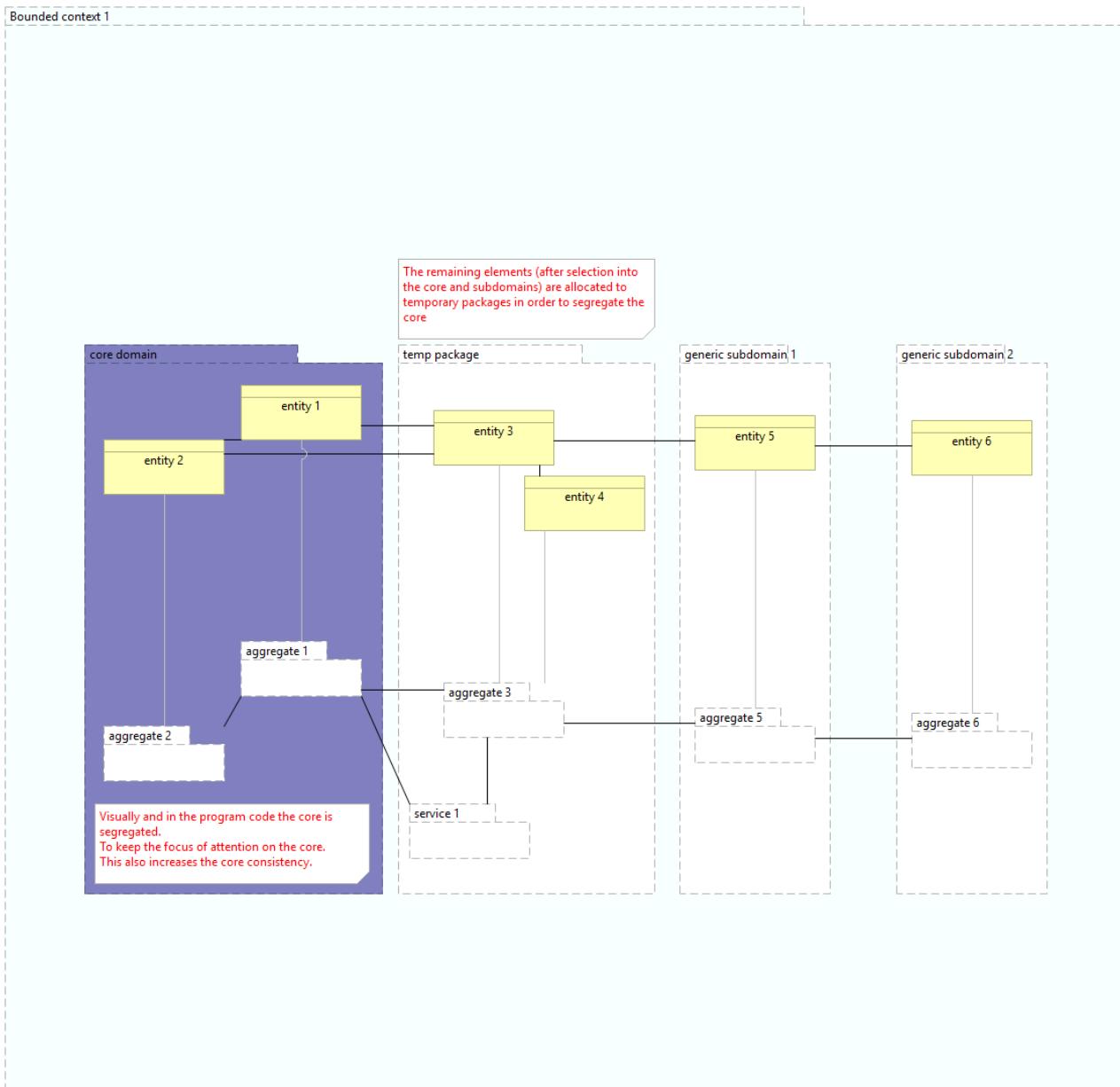
HIGHLIGHTED CORE



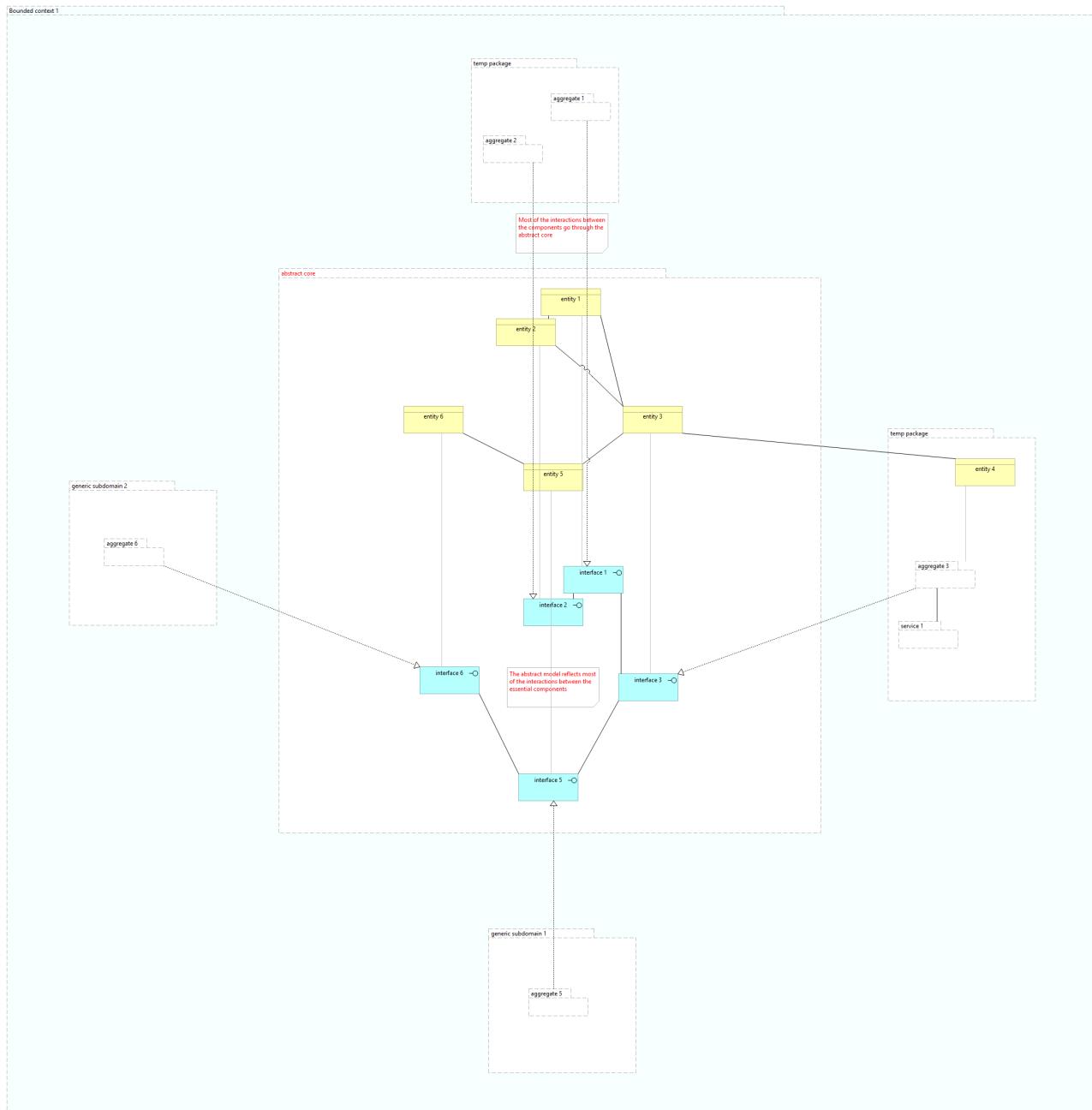
COHESIVE MECHANISMS



SEGREGATED CORE

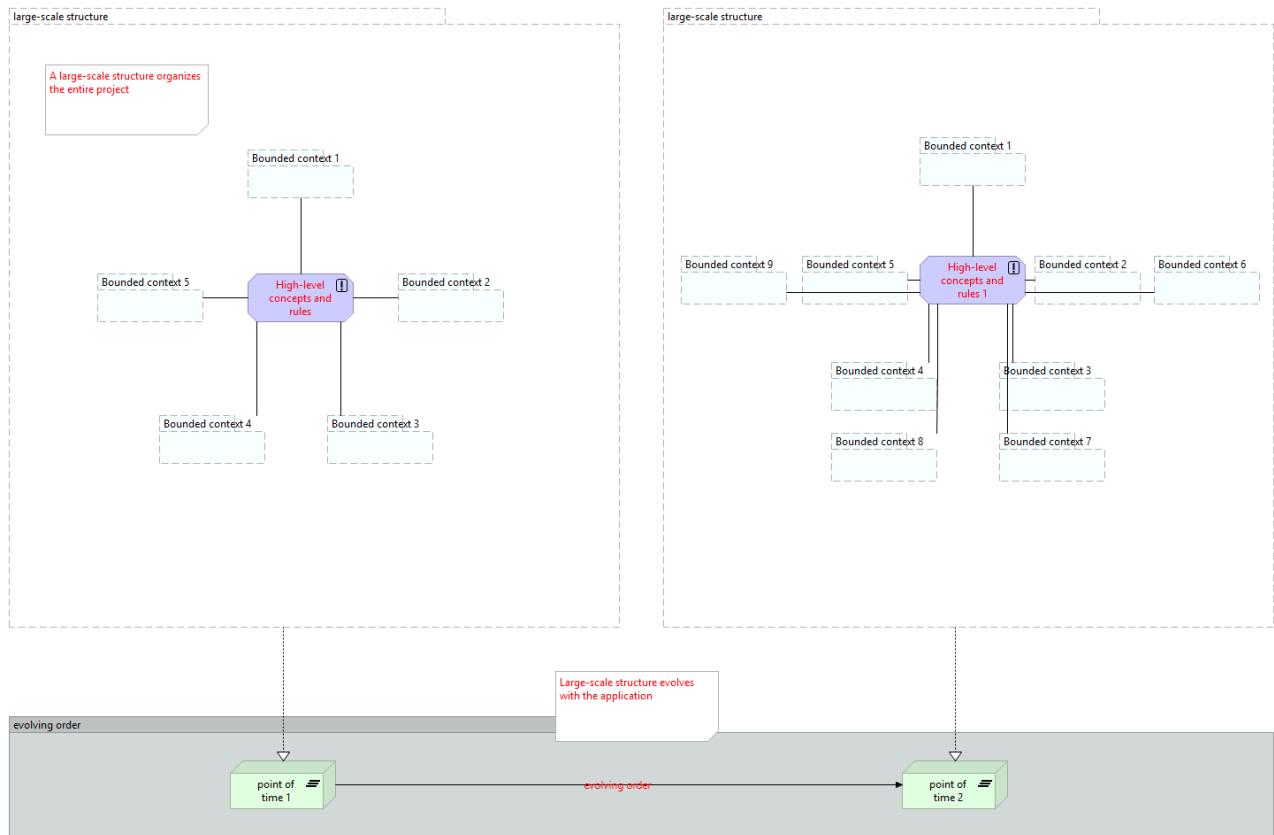


ABSTRACT CORE

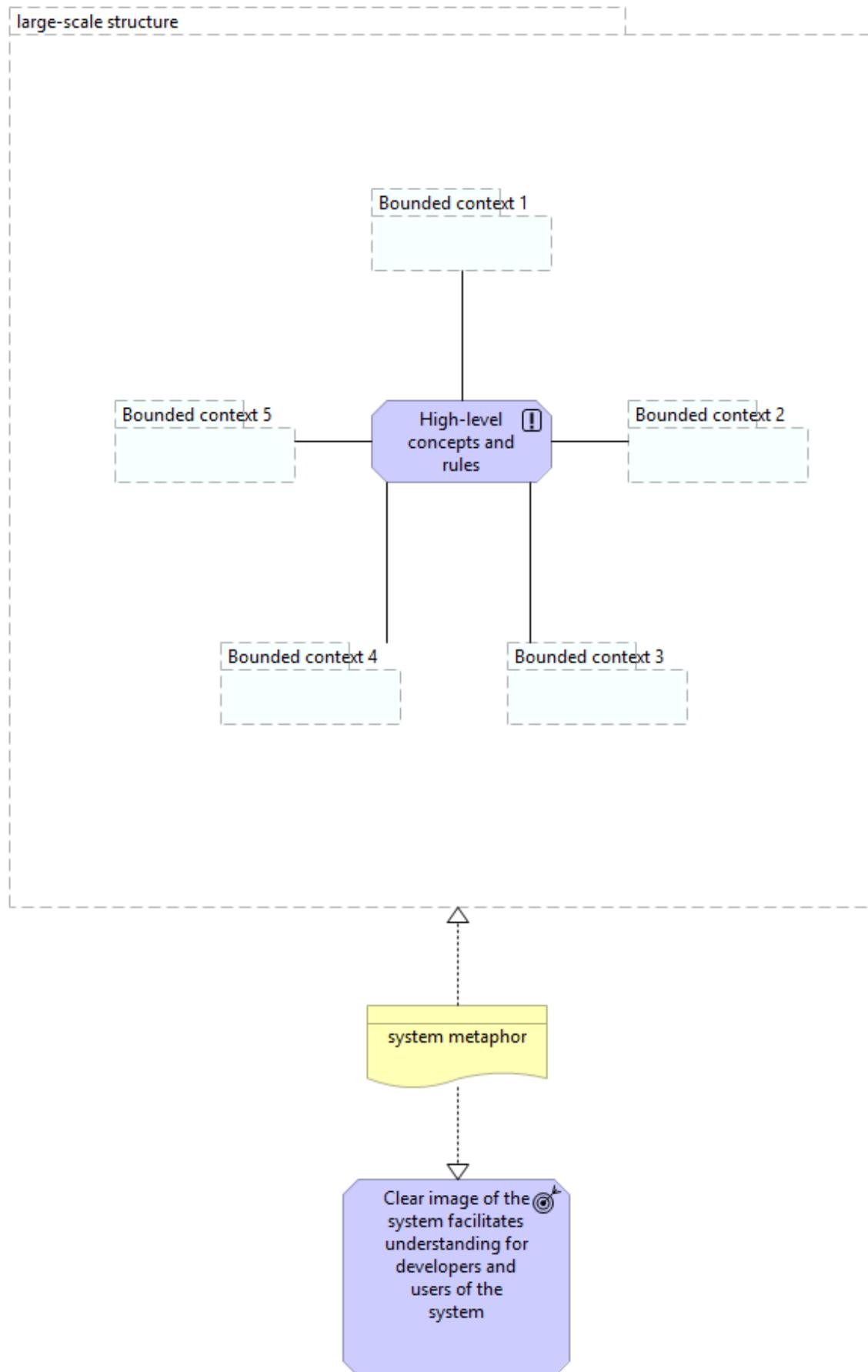


LARGE-SCALE STRUCTURE

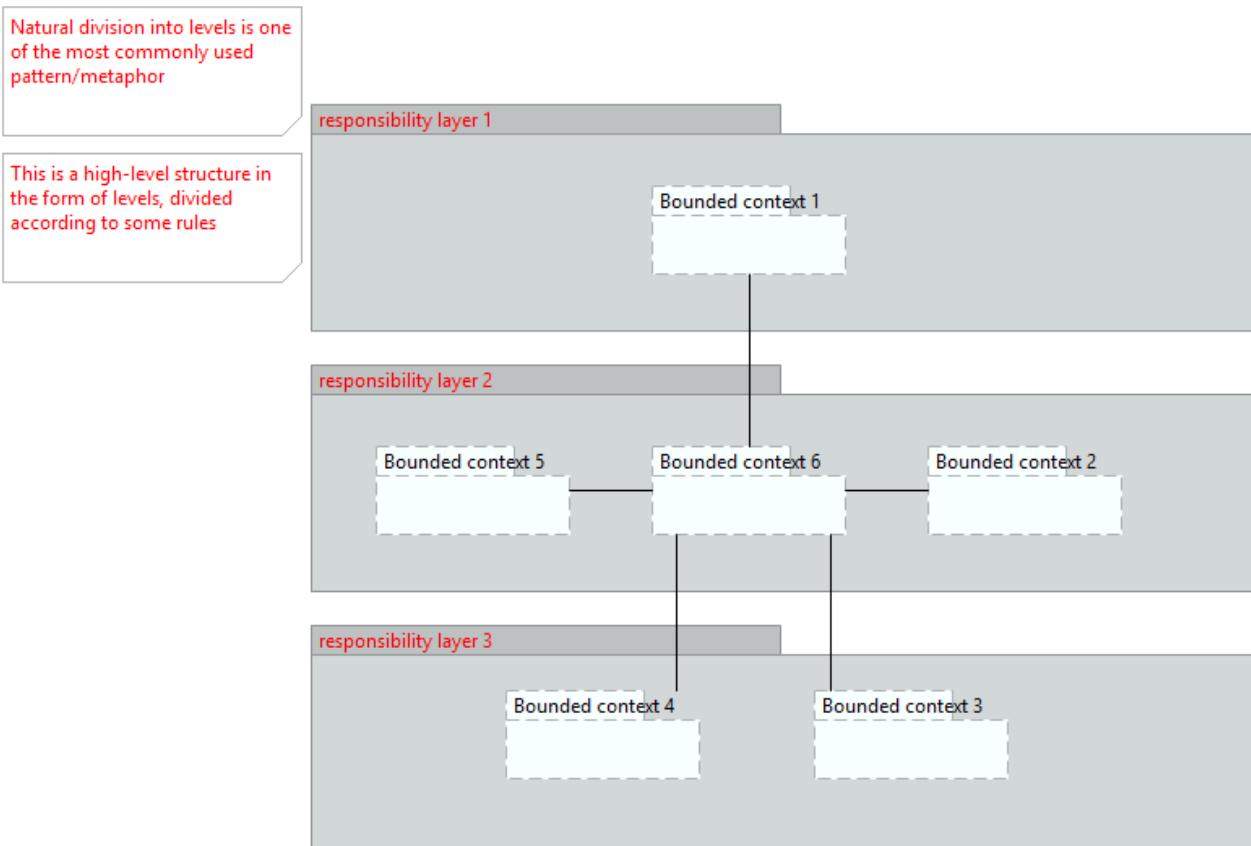
EVOLVING ORDER



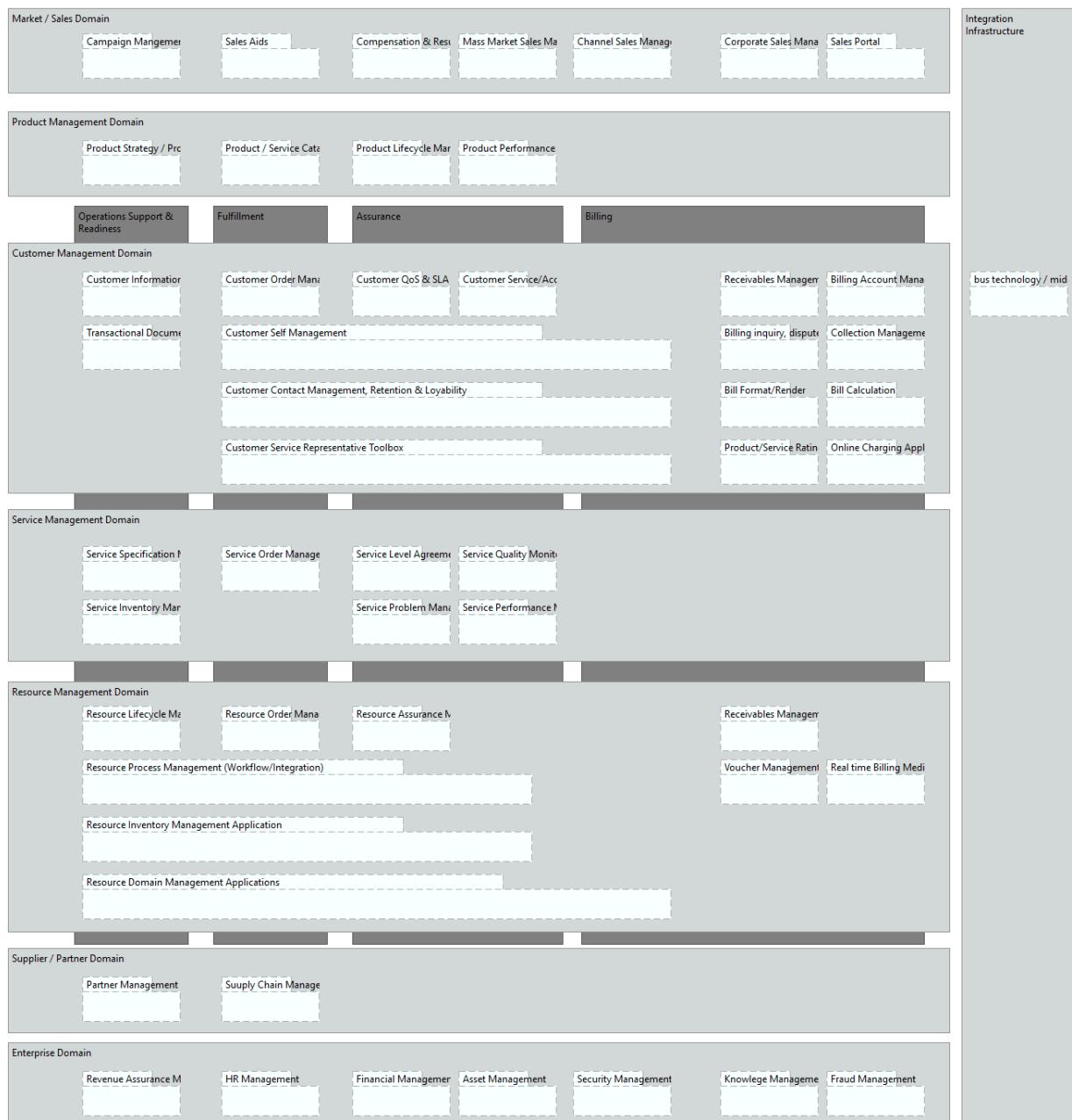
SYSTEM METAPHOR



RESPONSIBILITY LAYERS



Layers based on NGOSS
Framework Telecom Application
Map



Layers based on Porter's Value Chain

Firm infrastructure

Bounded context 1

Human Resources Management

Bounded context 5

Bounded context 6

Bounded context 2

Technological Development

Bounded context 4

Bounded context 3

Procurement

Bounded context 5

Bounded context 6

Inbound Logistics

Bounded context 7

Operations

Bounded context 8

Outbound Logistics

Bounded context 9

Marketing & Sales

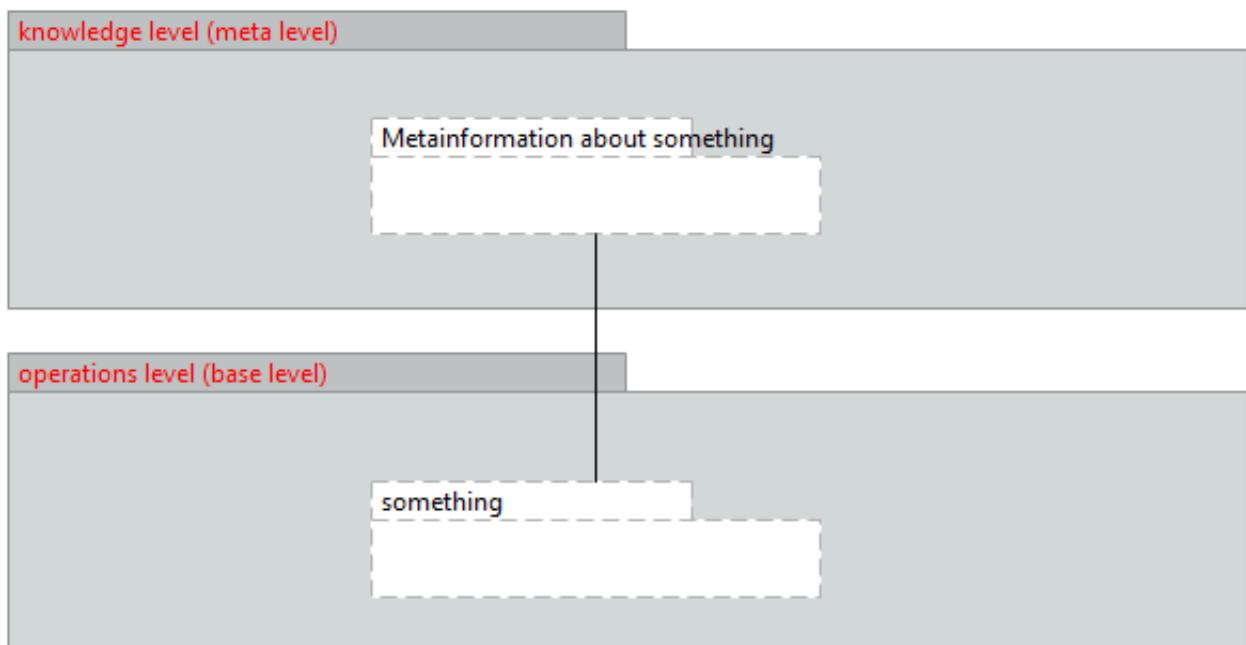
Bounded context 10

Service

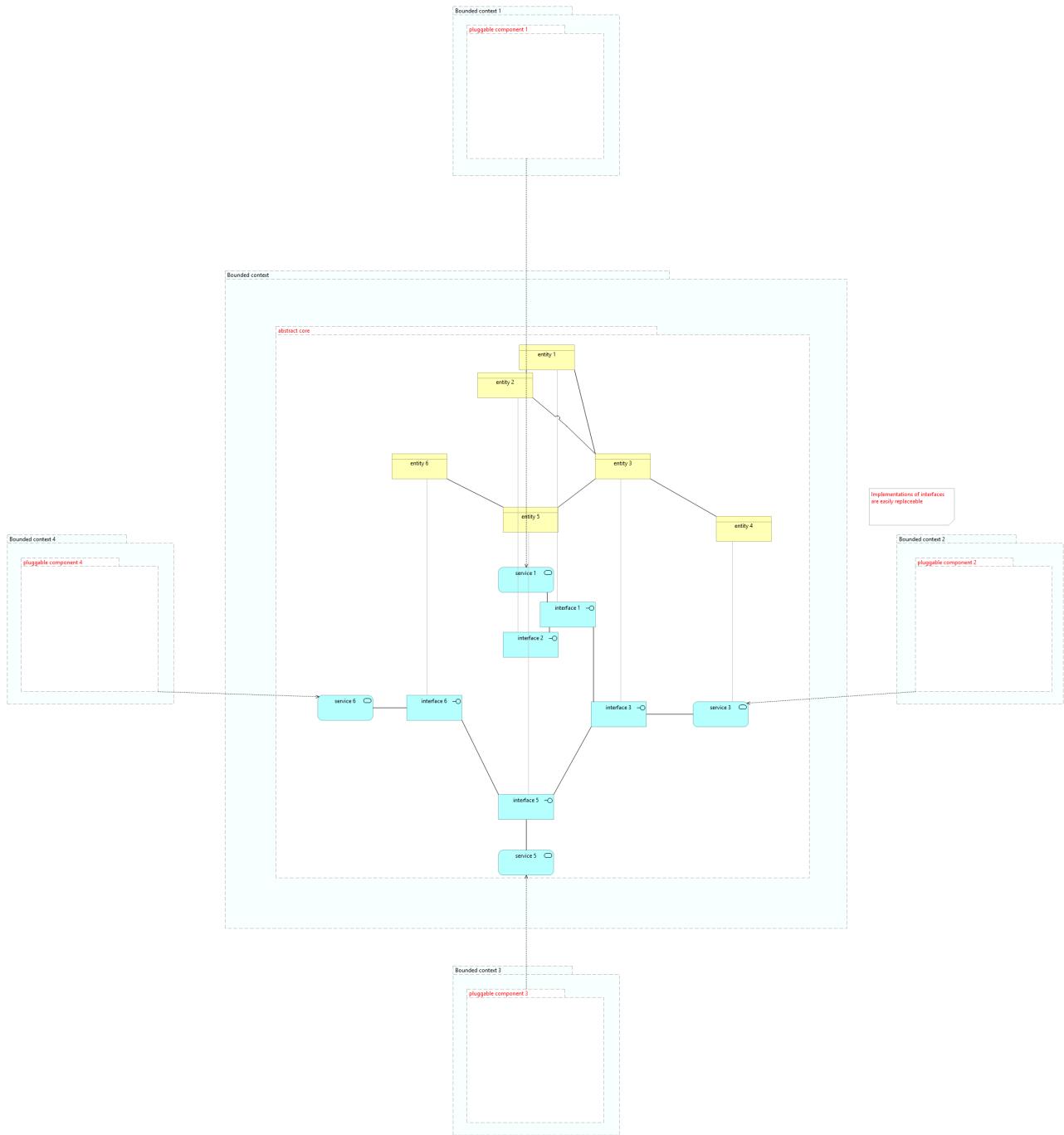
Bounded context 11



KNOWLEDGE LEVEL



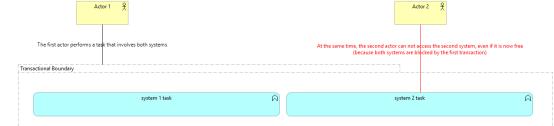
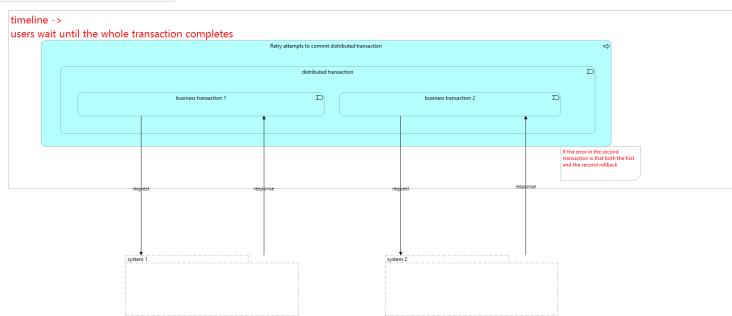
PLUGGABLE COMPONENT FRAMEWORK



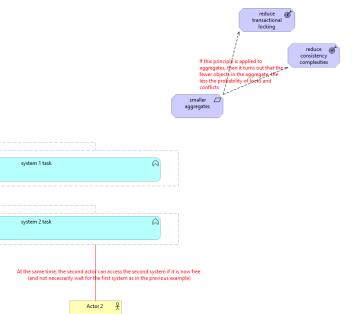
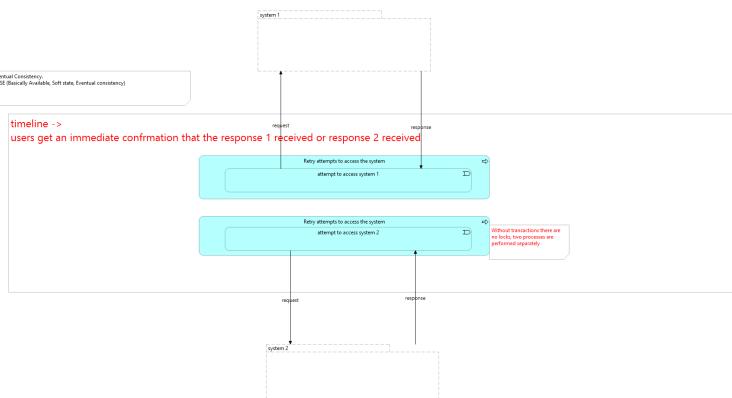
ADDITIONAL PATTERNS

TYPES OF CONSISTENCY

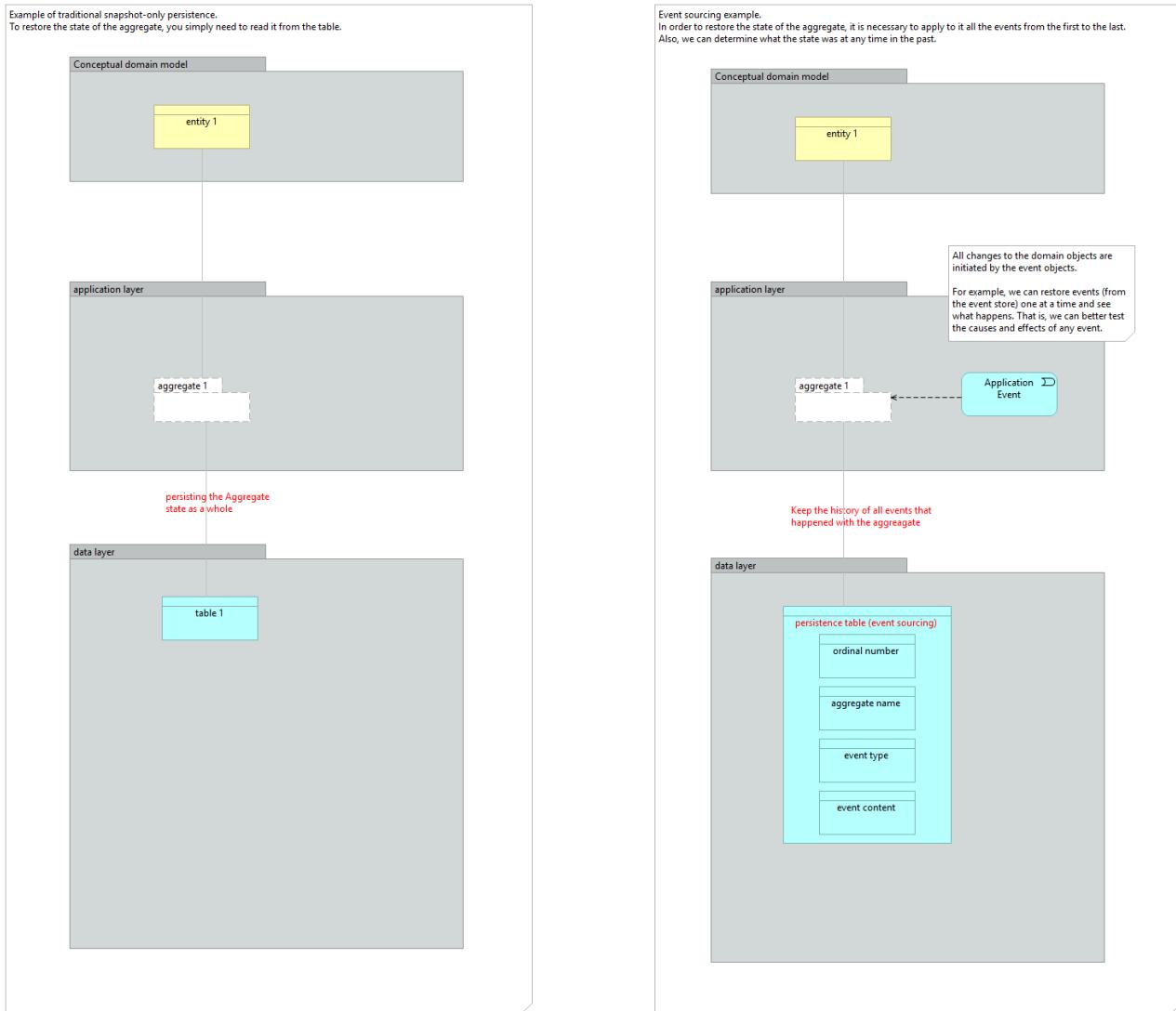
Transactional consistency:
Ensures that the system is in a consistent state.
ACID (Atomicity, Consistency, Isolation, Durability)
The more systems a distributed transaction, the greater the risk of potential locks and conflicts.



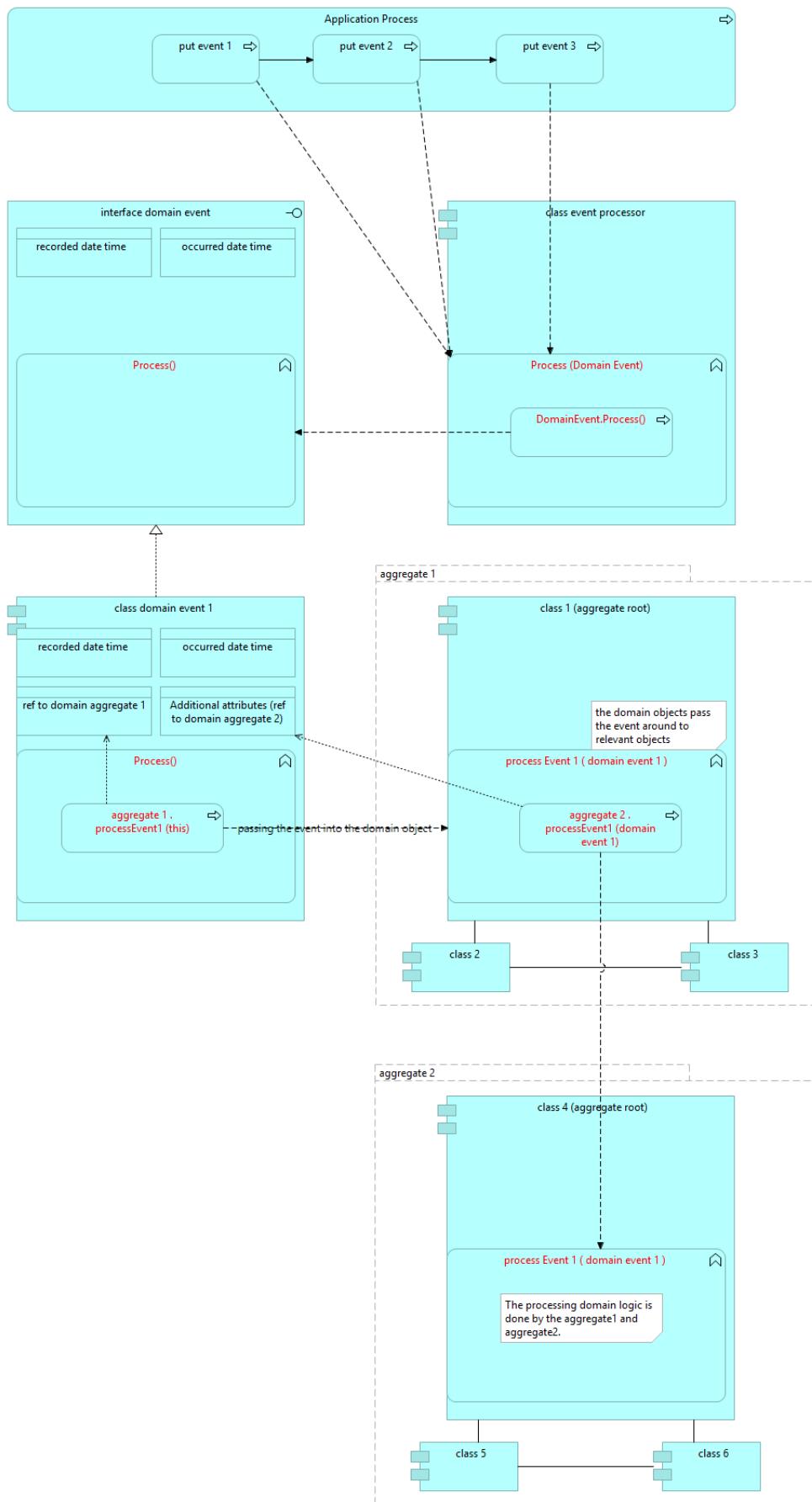
Eventual Consistency:
BASE (Basically Available, Soft state, Eventual consistency)



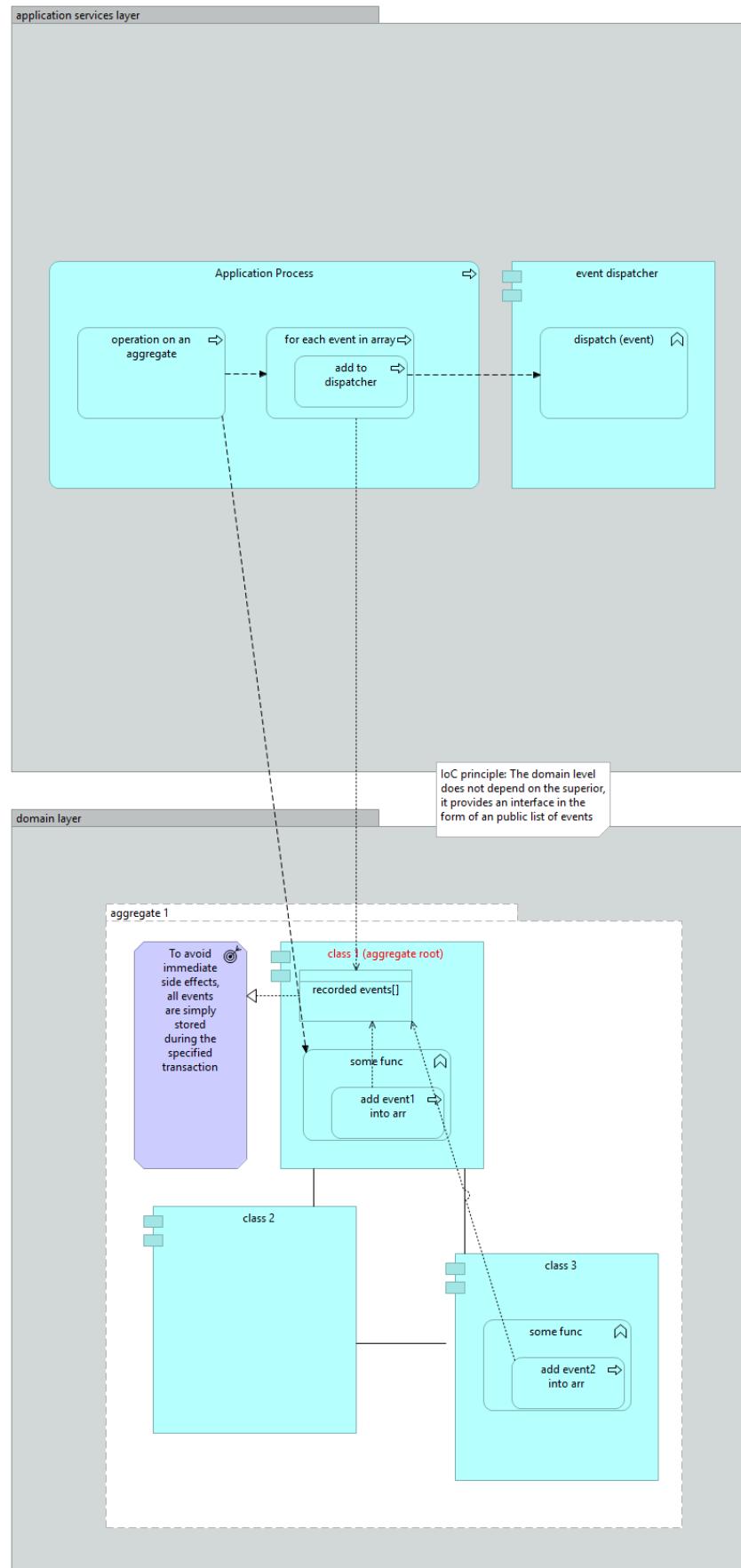
EVENT SOURCING



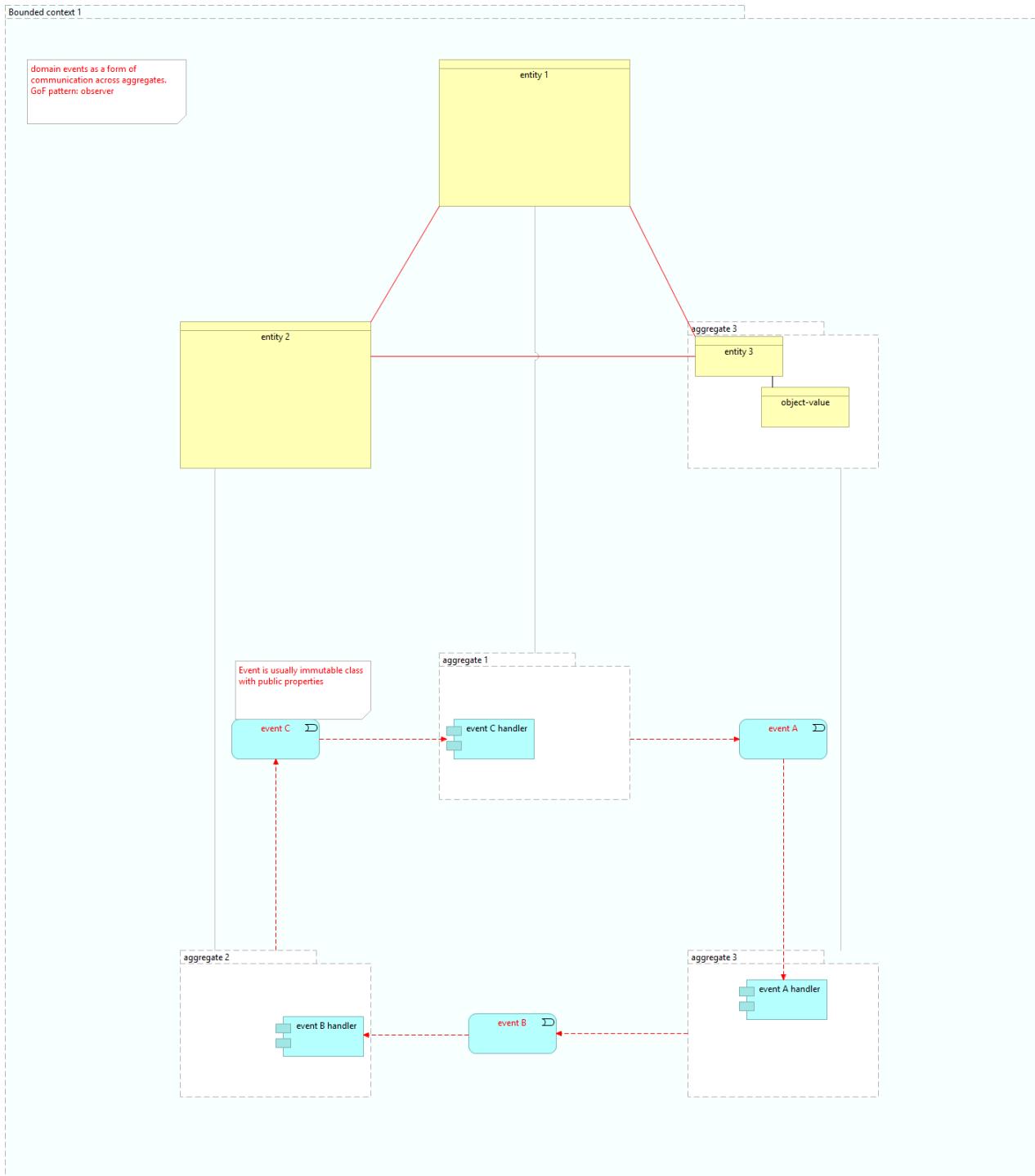
EVENT PROCESSOR



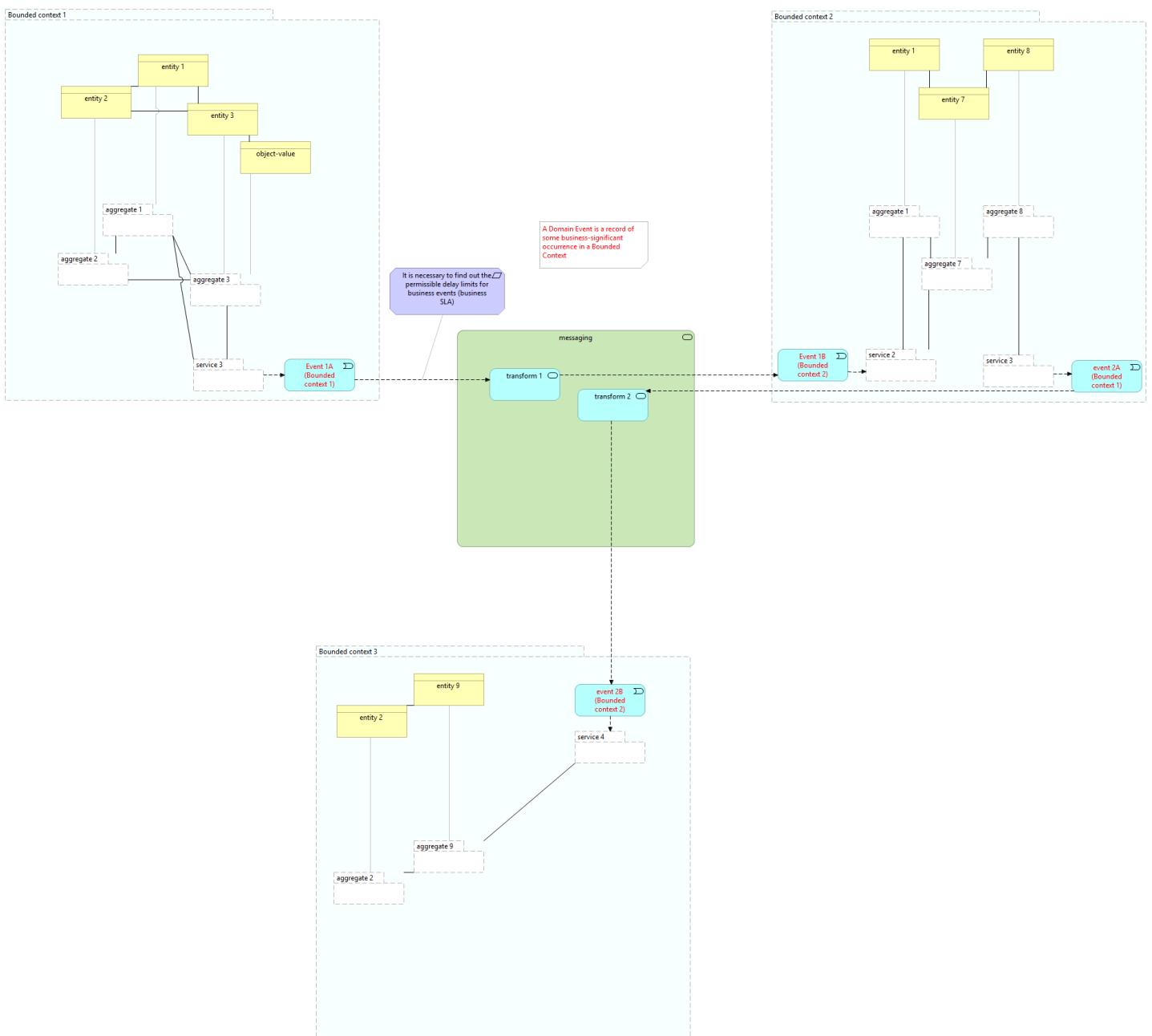
EVENT DISPATCHER



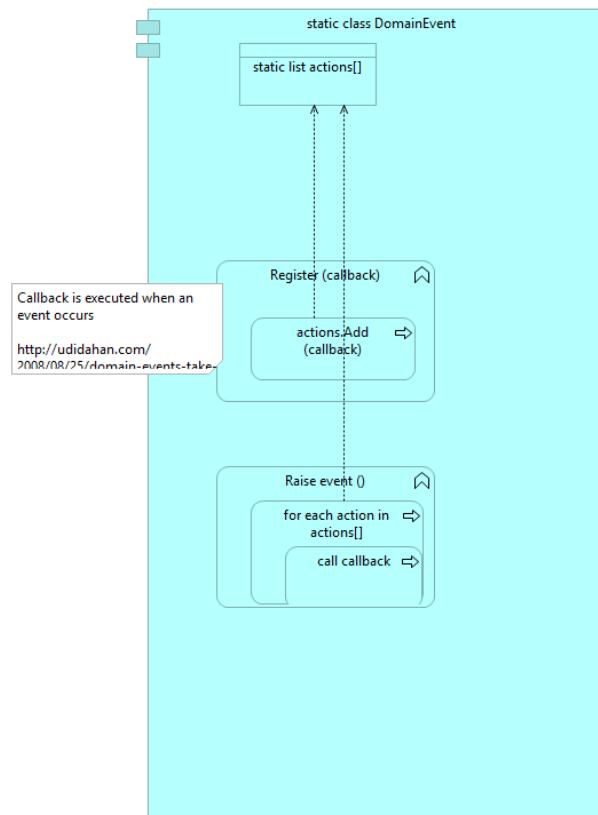
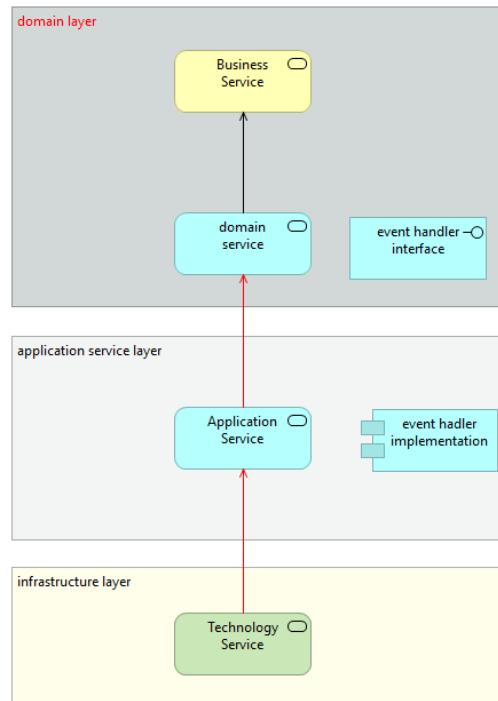
INTERNAL DOMAIN EVENTS



EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS

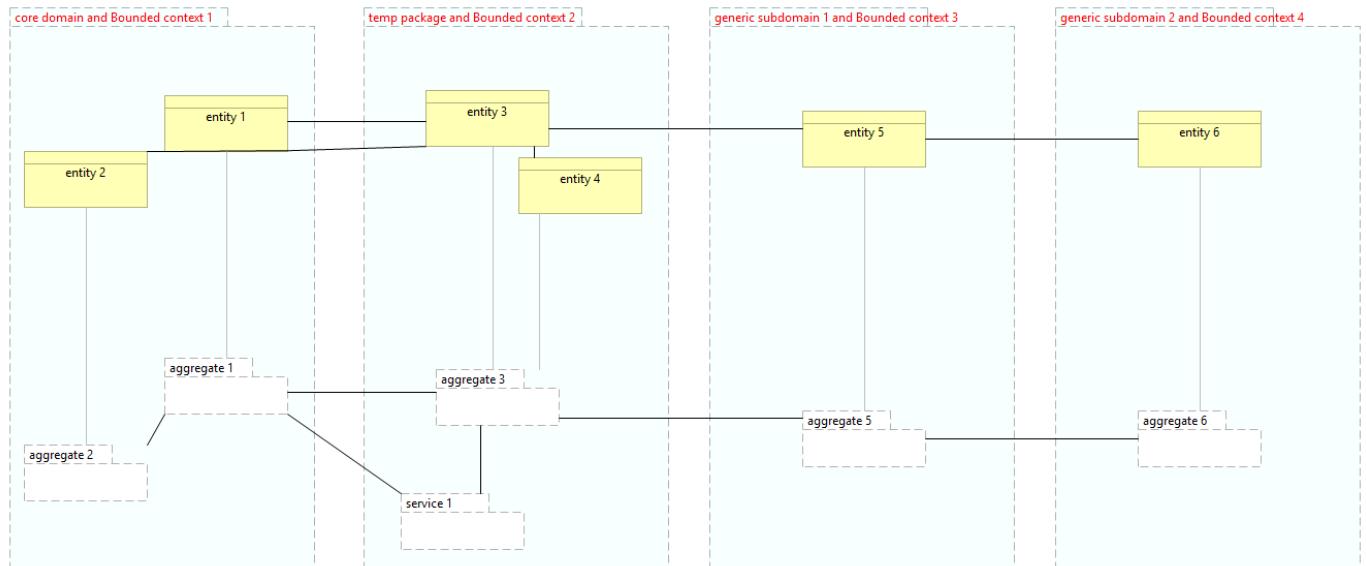


STATIC DOMAIN EVENTS CLASS

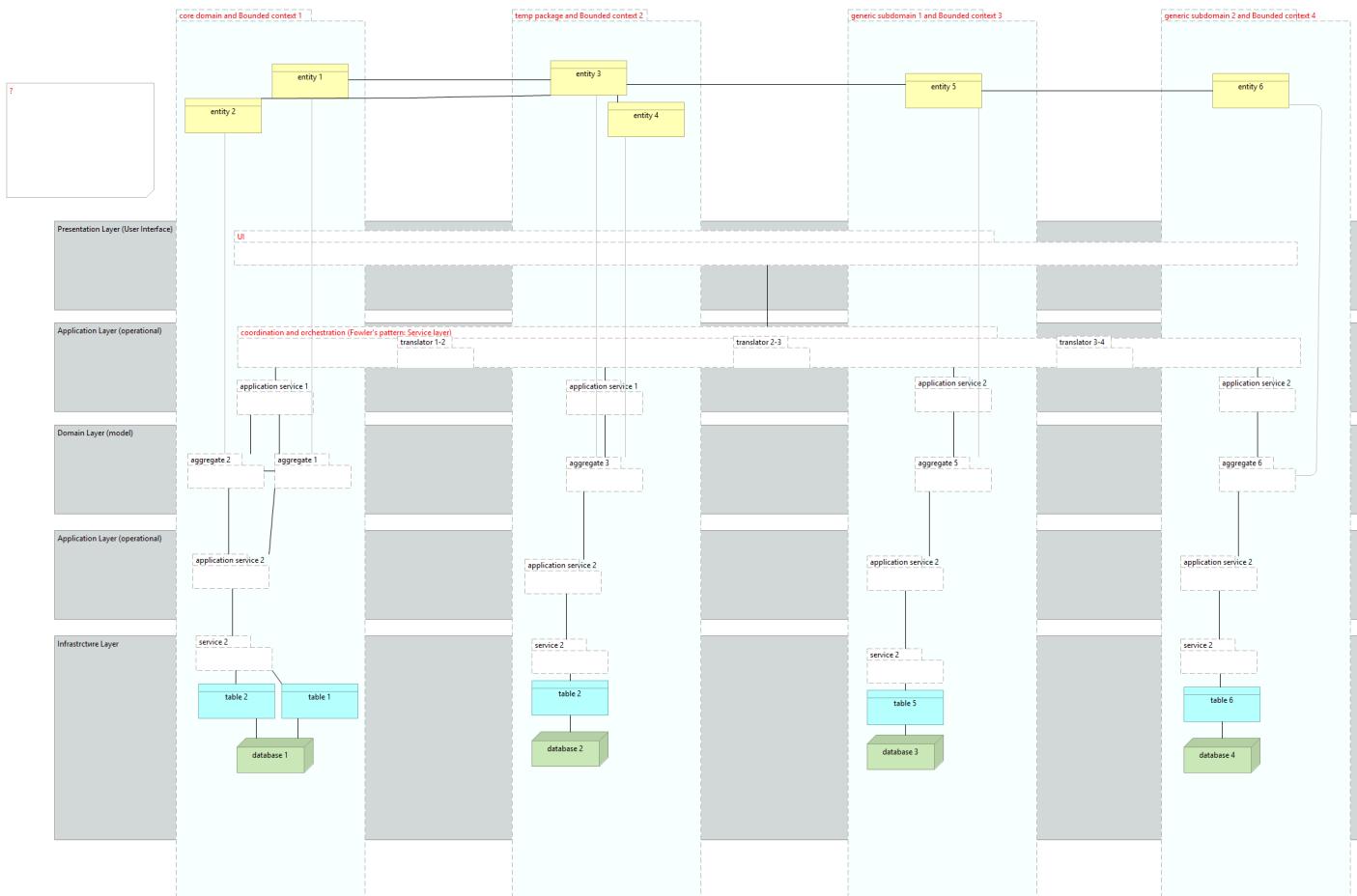


ONE SUBDOMAIN PER BOUNDED CONTEXT

May be multiple Subdomains in one Bounded Context
but most optimal to use one Subdomain per Bounded Context

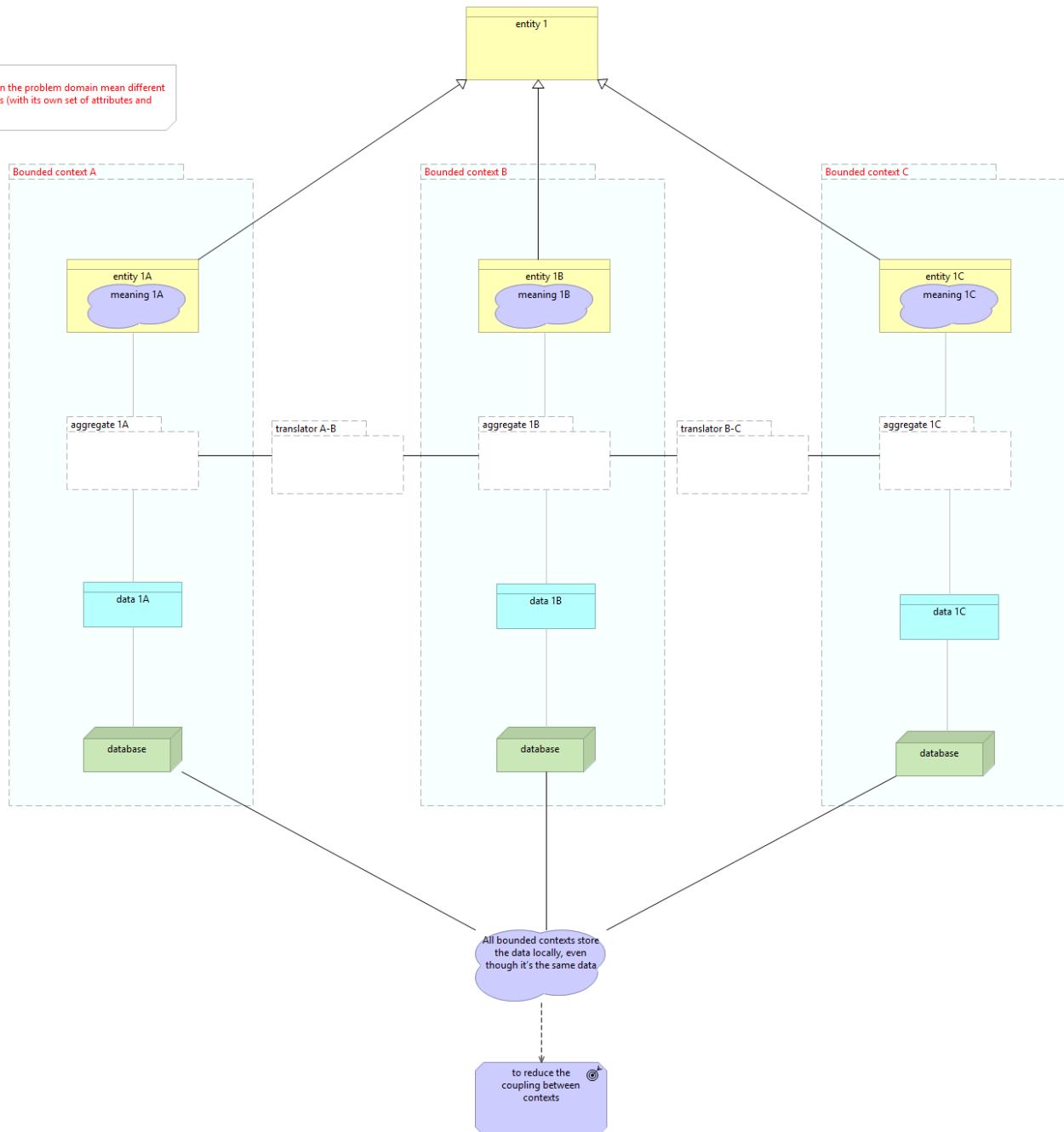


THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS

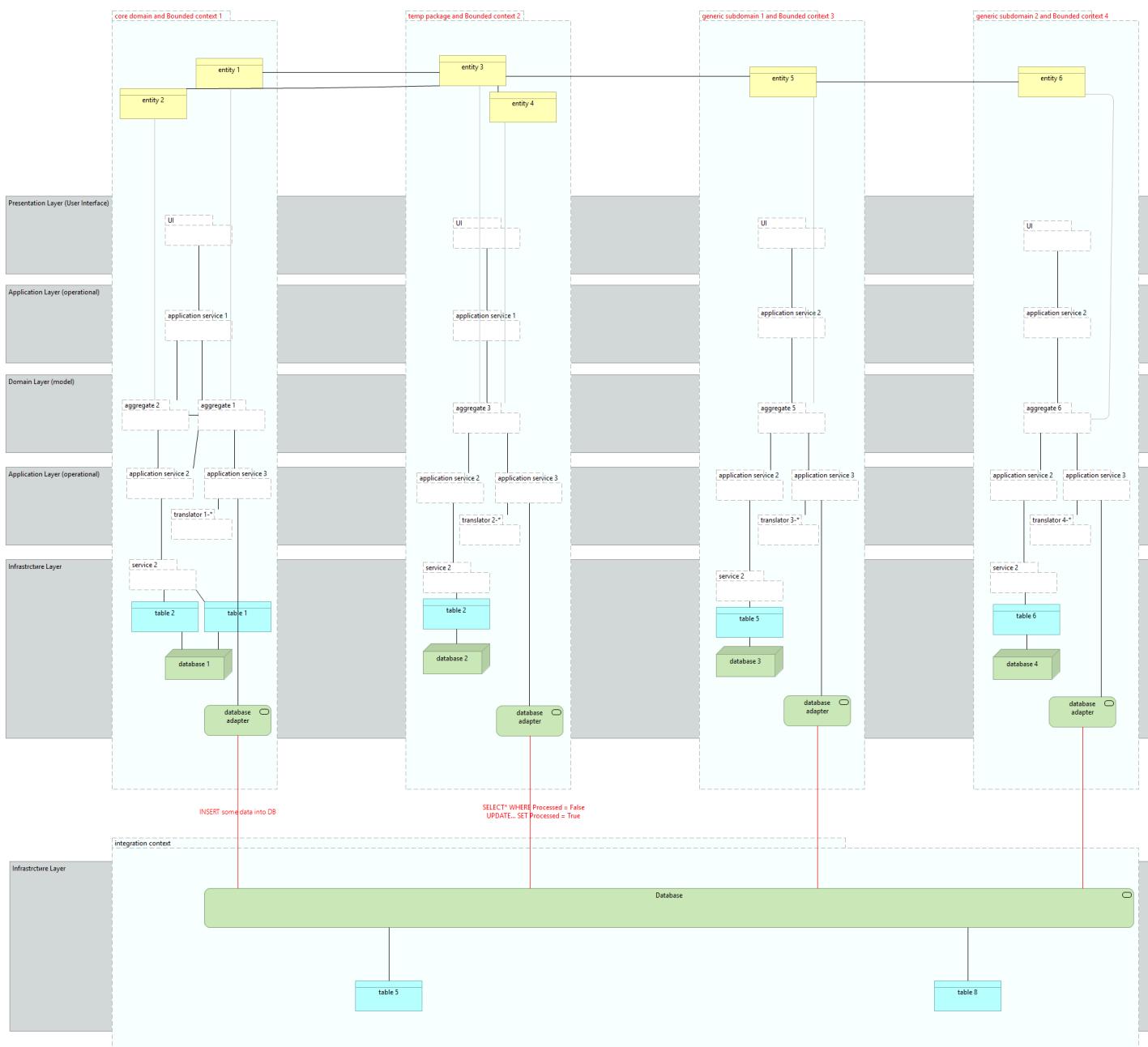


THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS

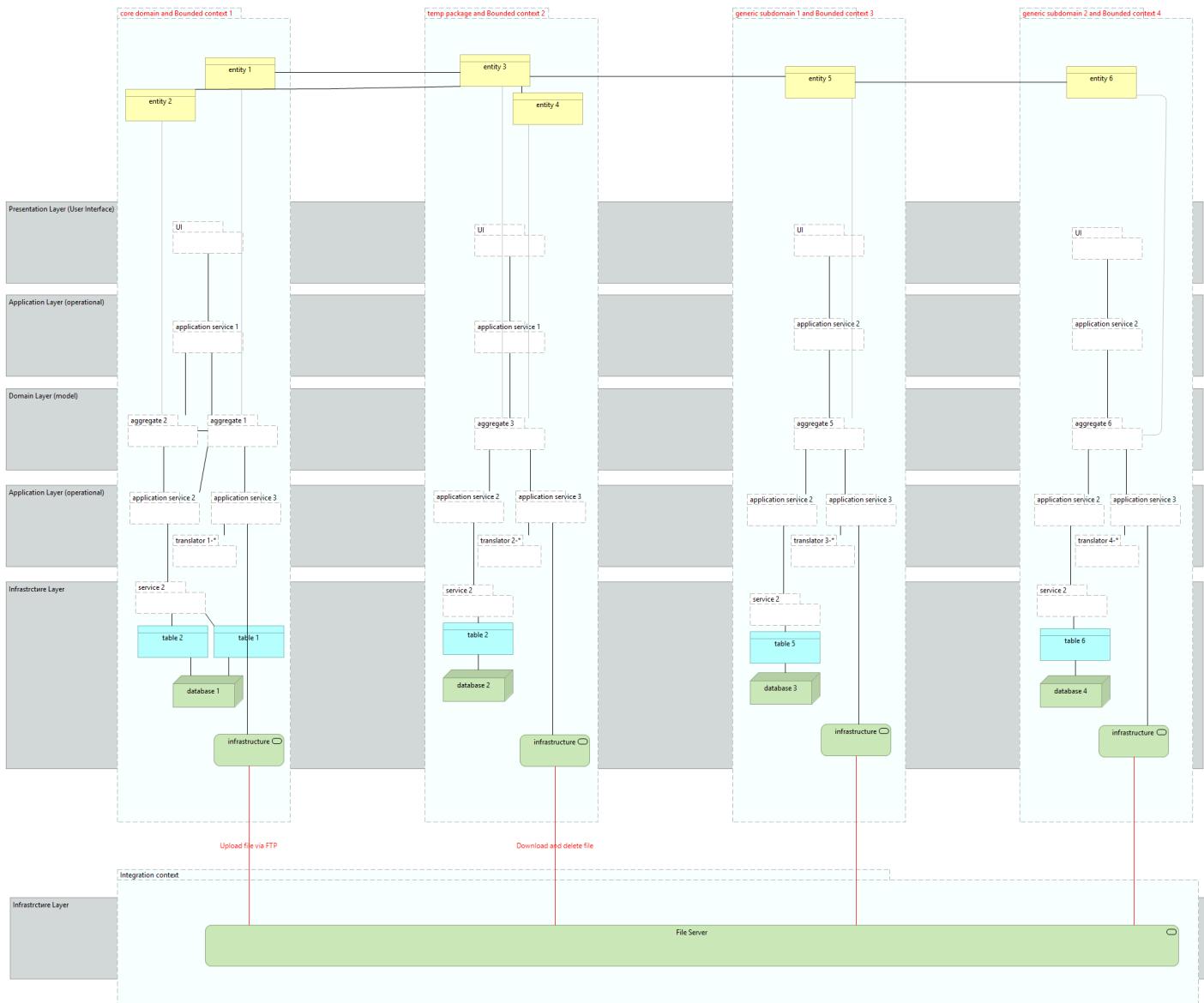
Example
The same physical entity in the problem domain mean different things in different contexts (with its own set of attributes and aspects of behavior).



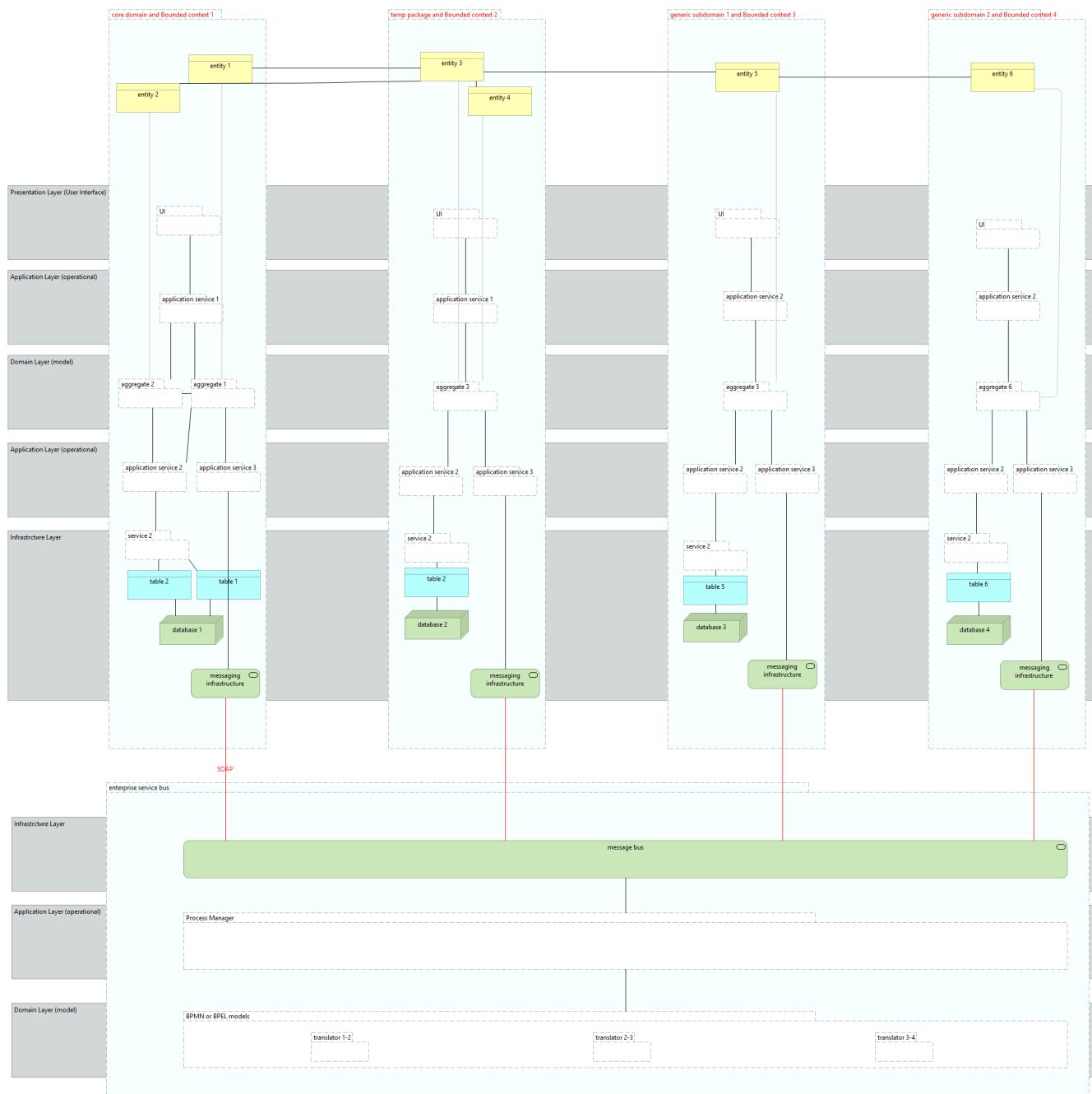
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE



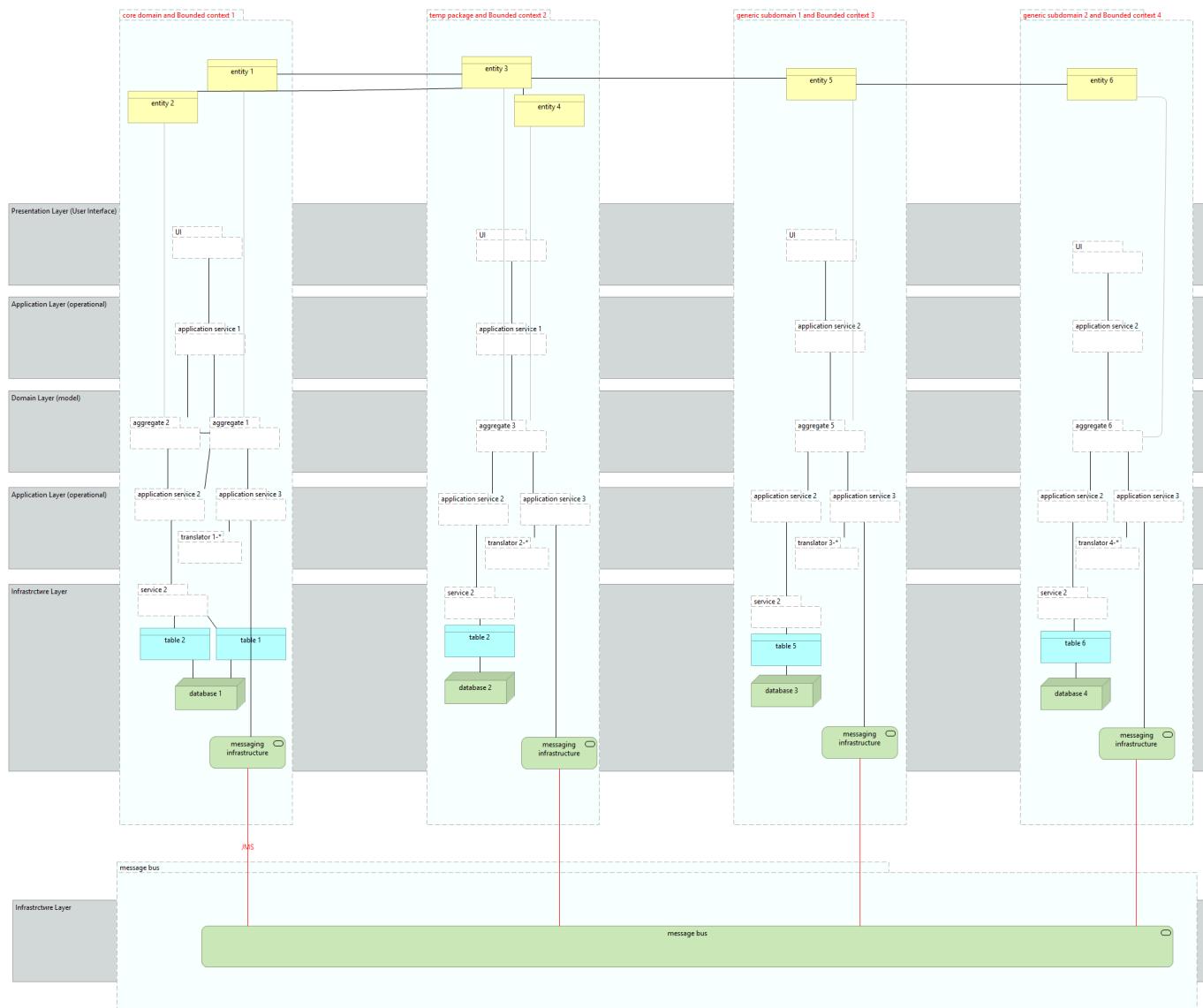
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES



INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS



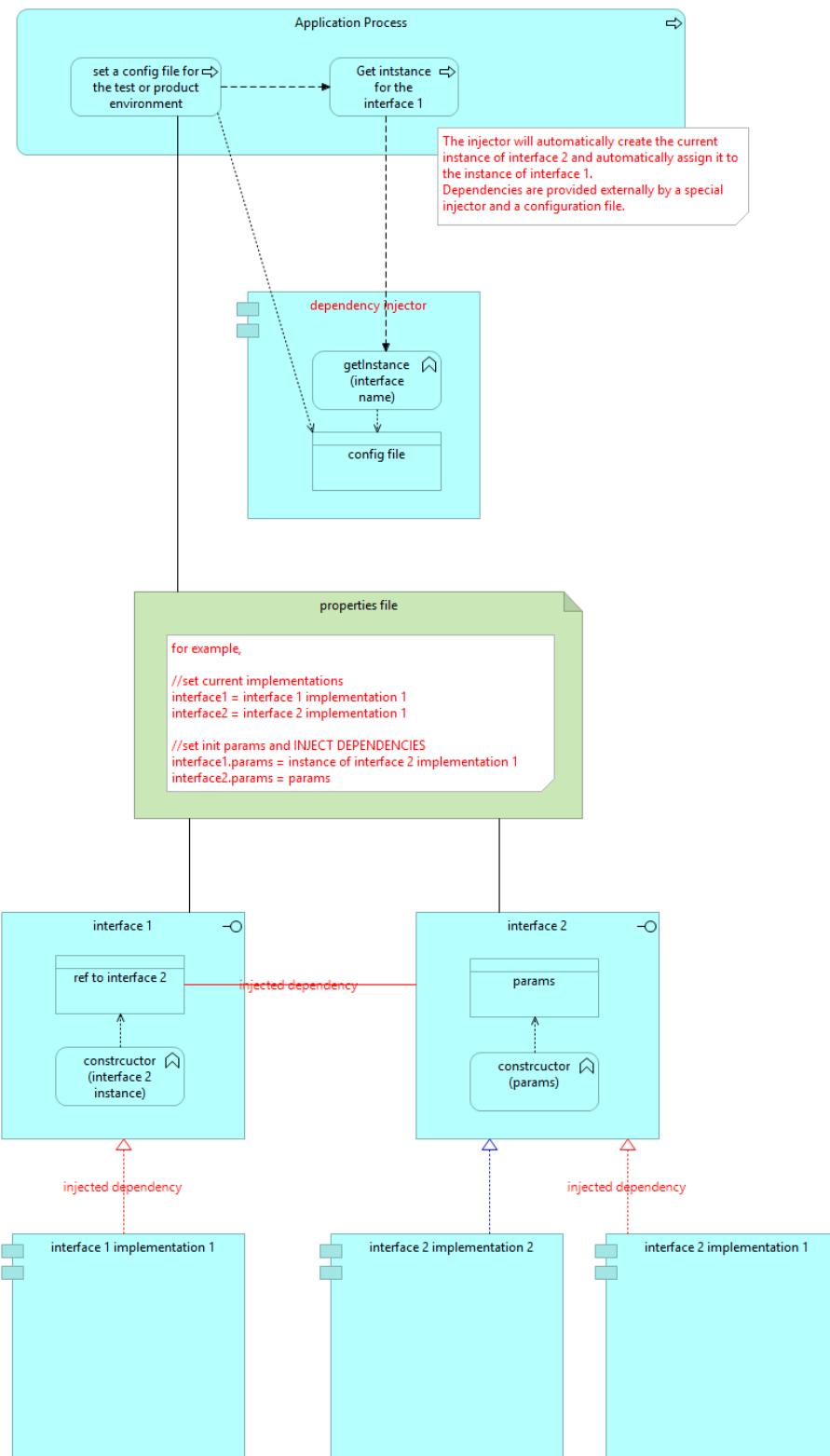
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE



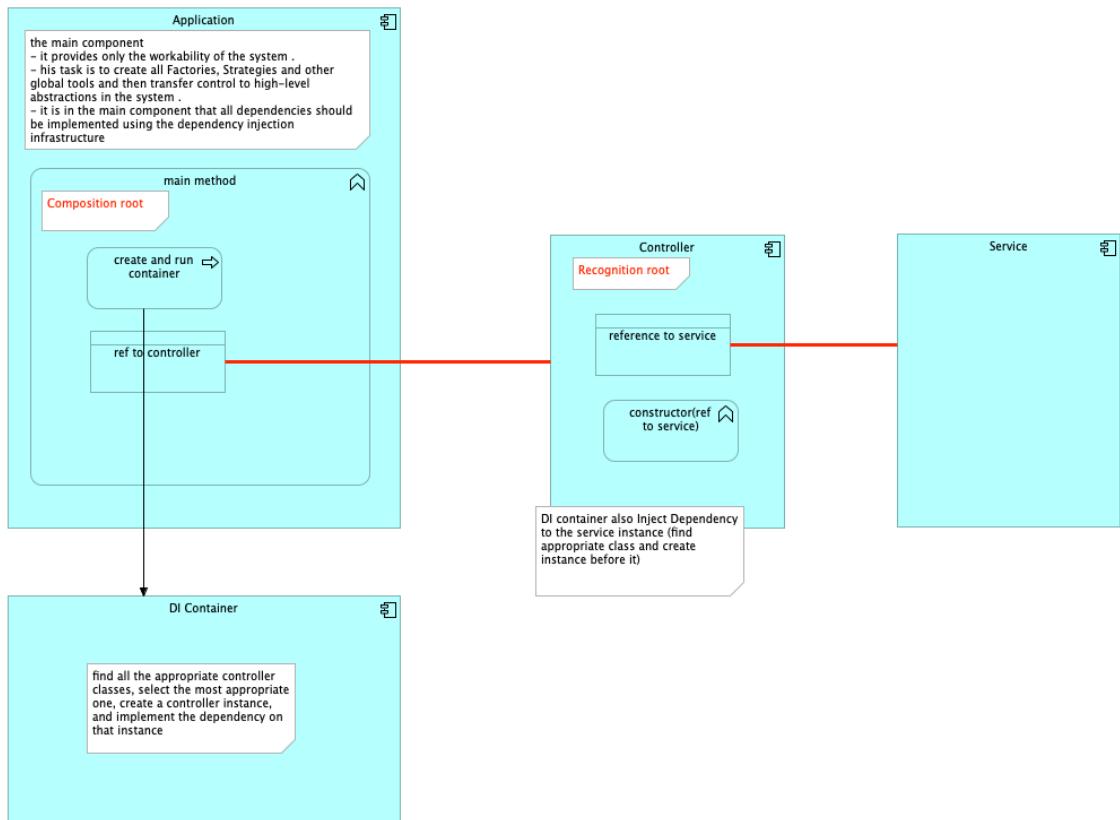
DEPENDENCY INJECTION

principle of separating configuration from use
Fowler's pattern: plugin
Fowler's pattern: Segregated Interface

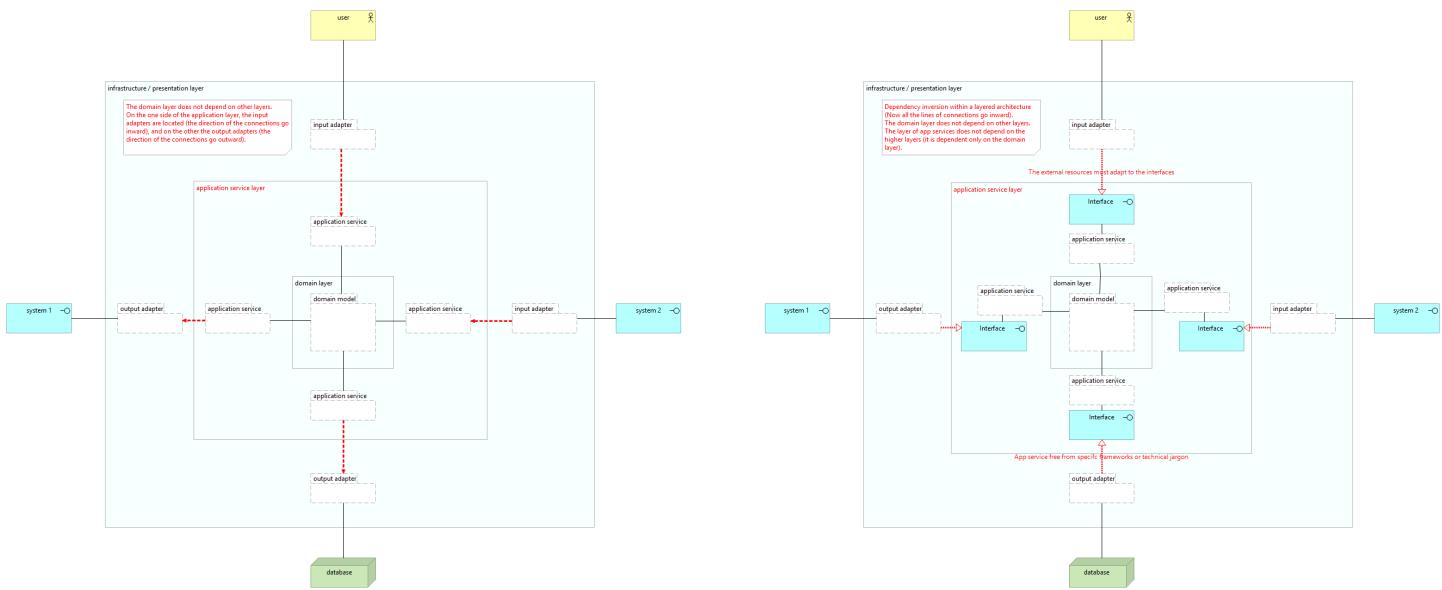
- inject the implementation into the application class
- removing the dependency from the application class to the plugin implementation



the diagram demonstrates the basic concepts of DI:
 - the main component
 - root of composition
 - recognition root – first recognized and injected class
 (the root of the object graph to be recognized)



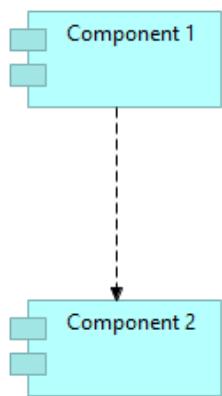
DEPENDENCY INVERSION



INVERSION OF CONTROL

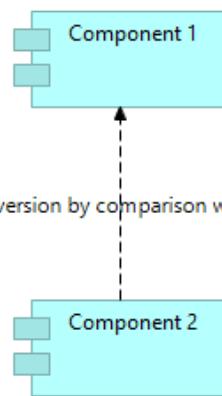
IoC
Inversion of control is about who initiates messages.

Variant 1



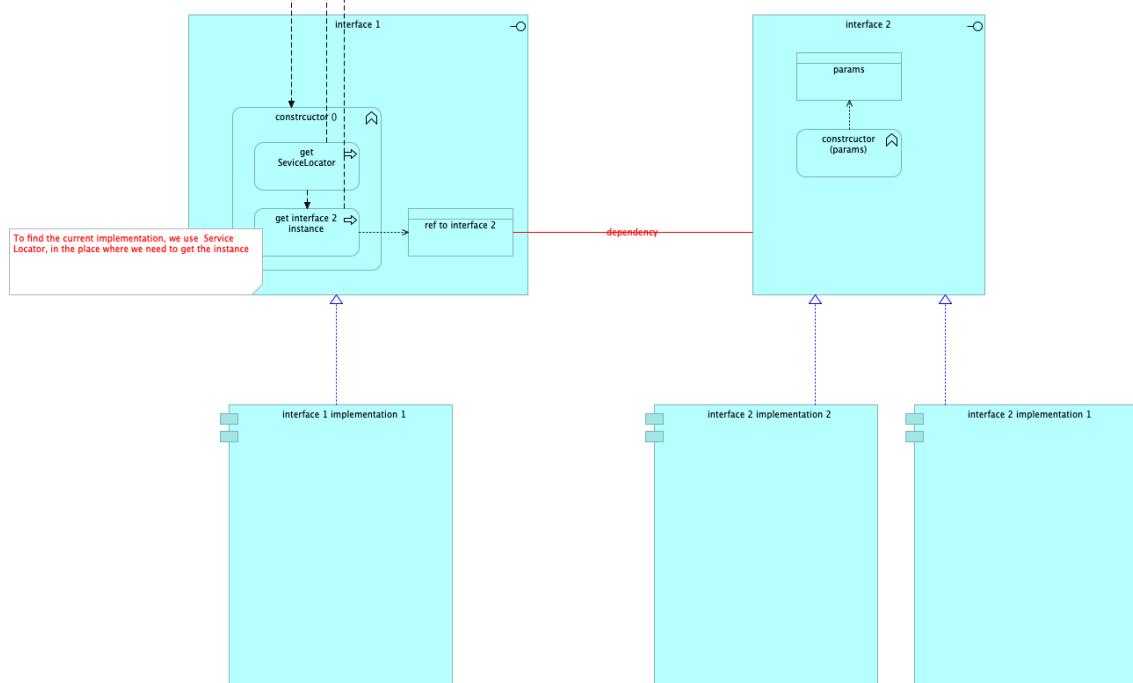
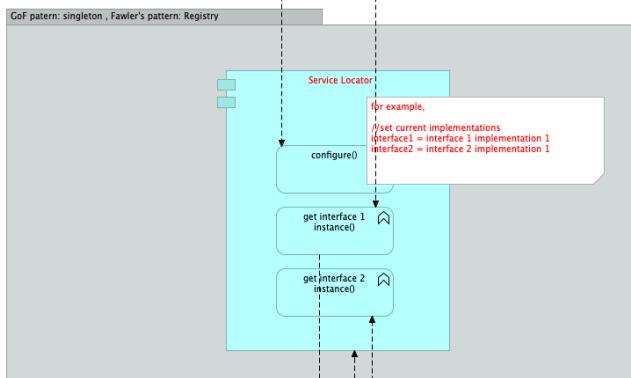
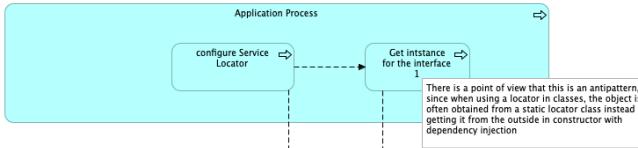
Variant 2

Here the control inversion by comparison with the previous case



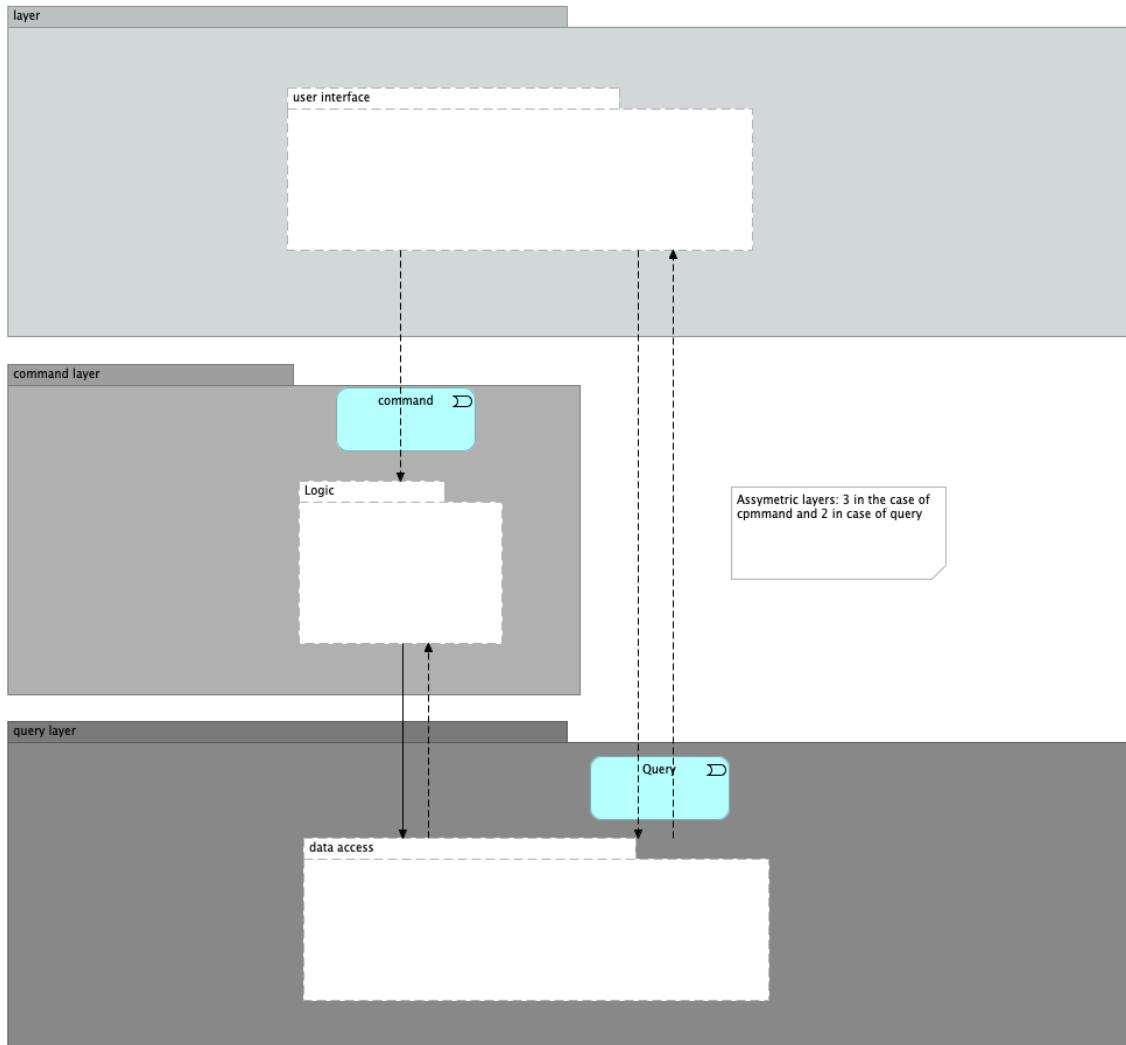
SERVICE LOCATOR

principle of separating configuration from use
 Fowler's pattern: plugin
 Fowler's pattern: Segregated Interface
 Fowler's pattern: registry



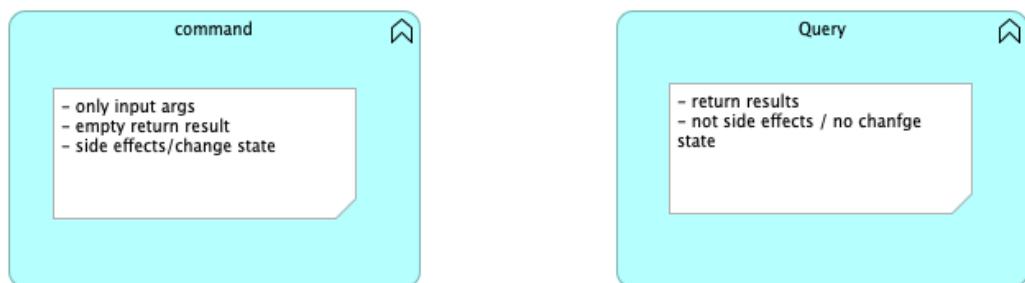
CQRS

Command Query Responsibility Segregation (CQRS)
Different layers for Commands and Queries

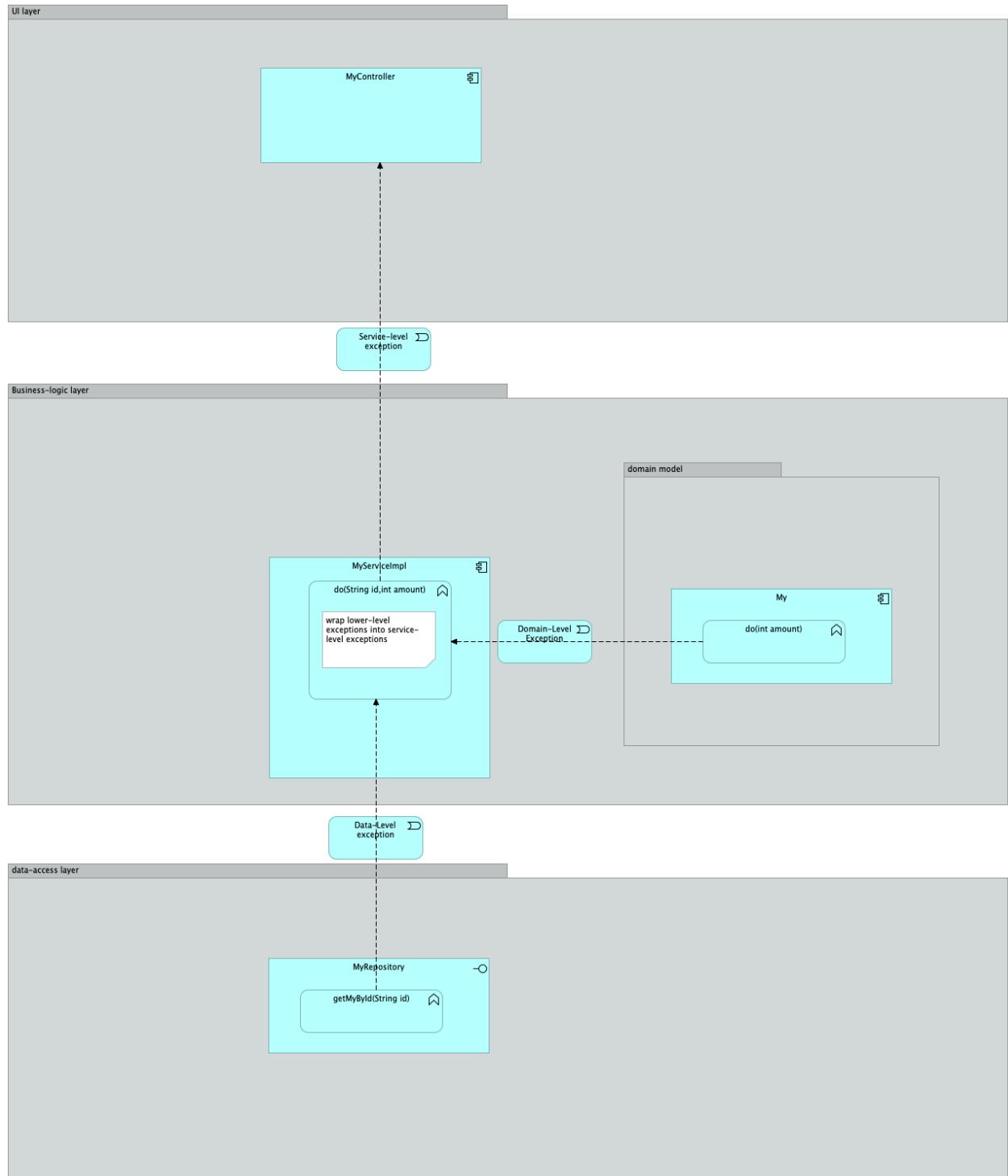


CQS

CQS Command/Query Separation

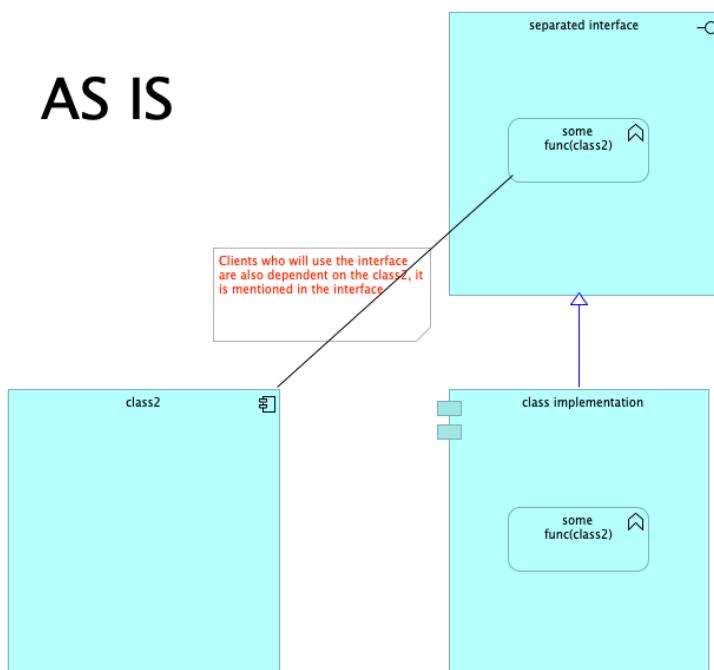


WRAP LOW-LEVEL EXCEPTIONS

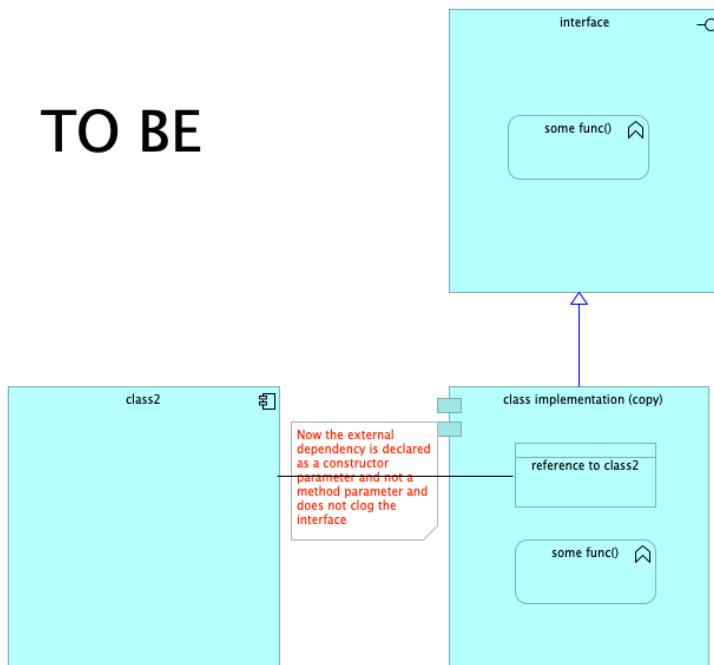


EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR

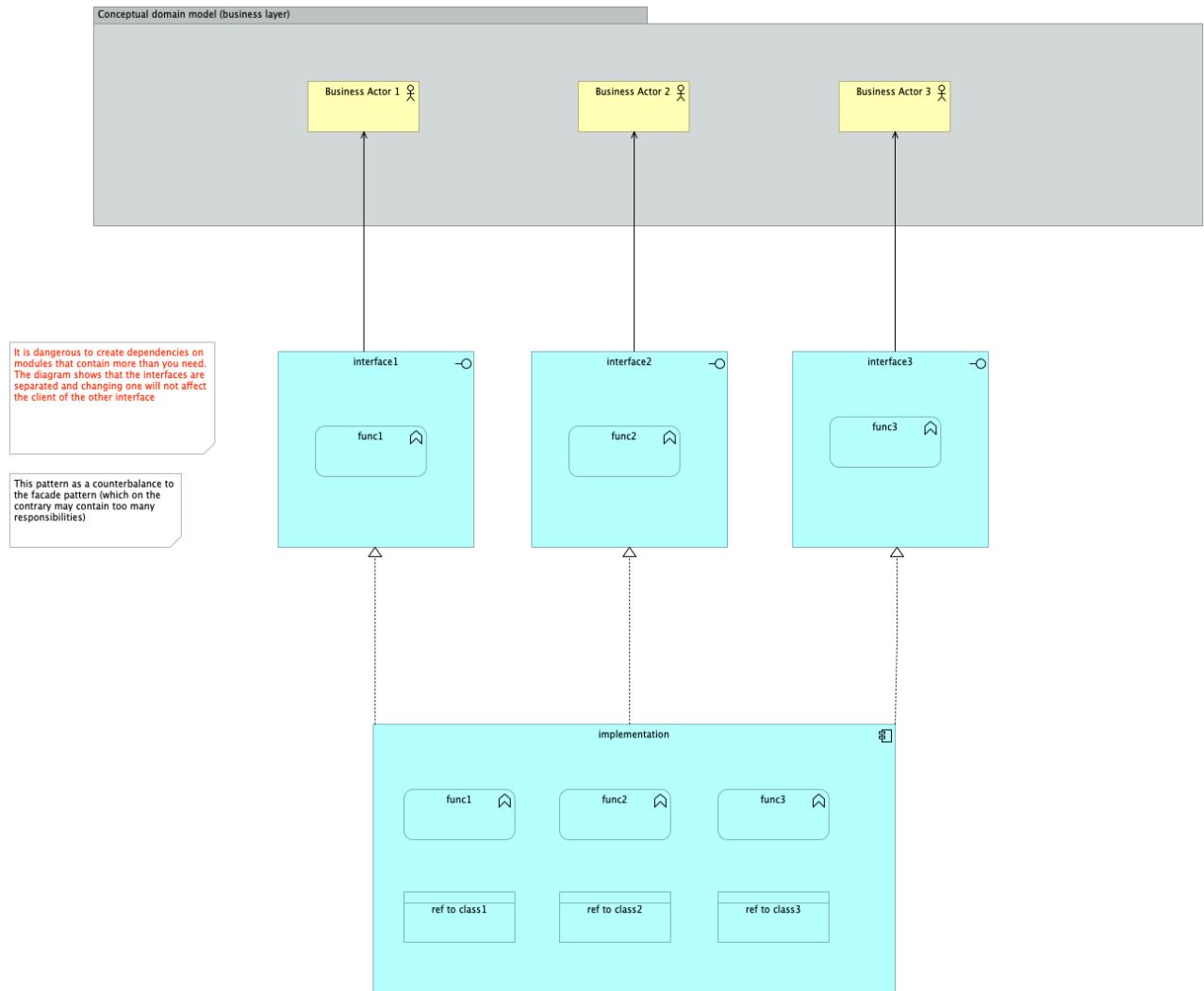
AS IS



TO BE



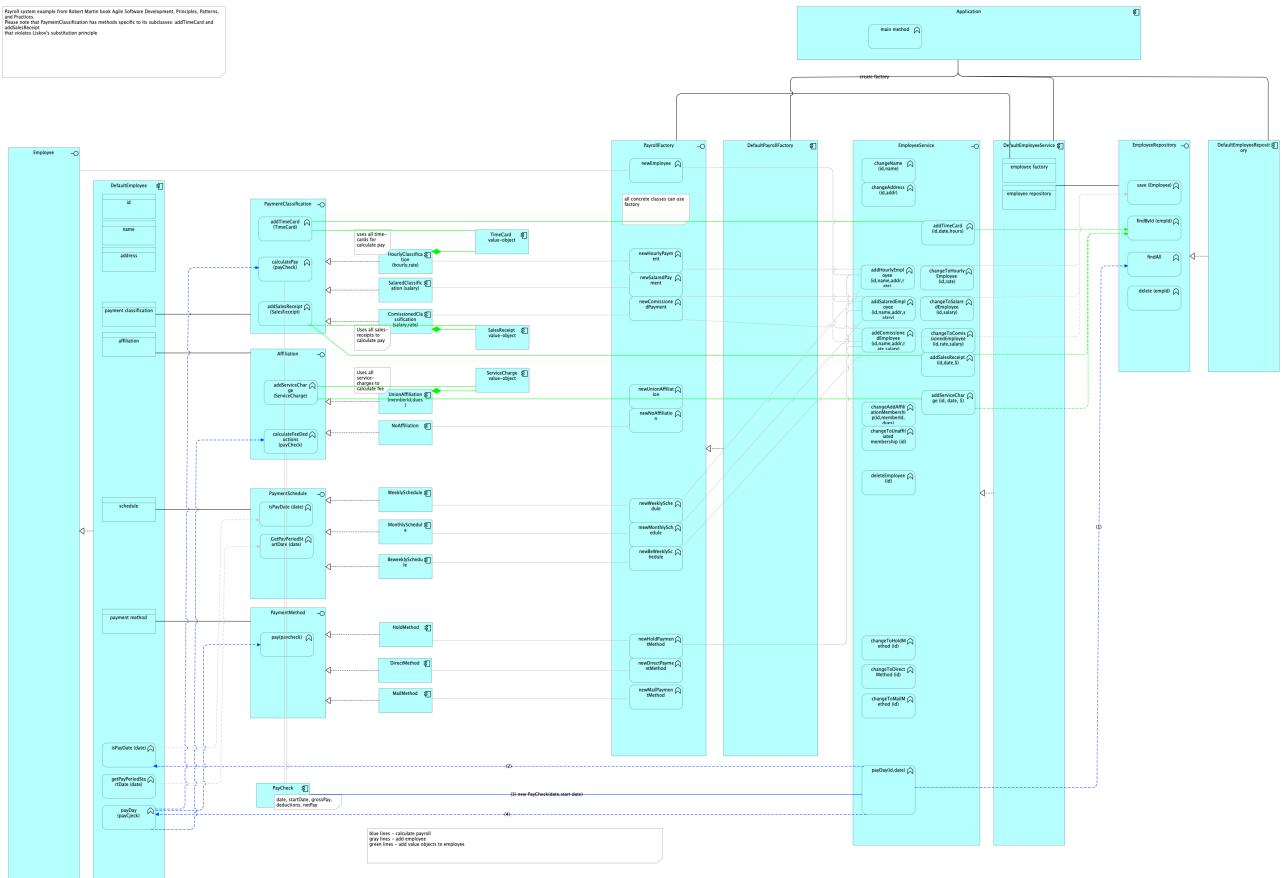
INTERFACE SEGREGATION

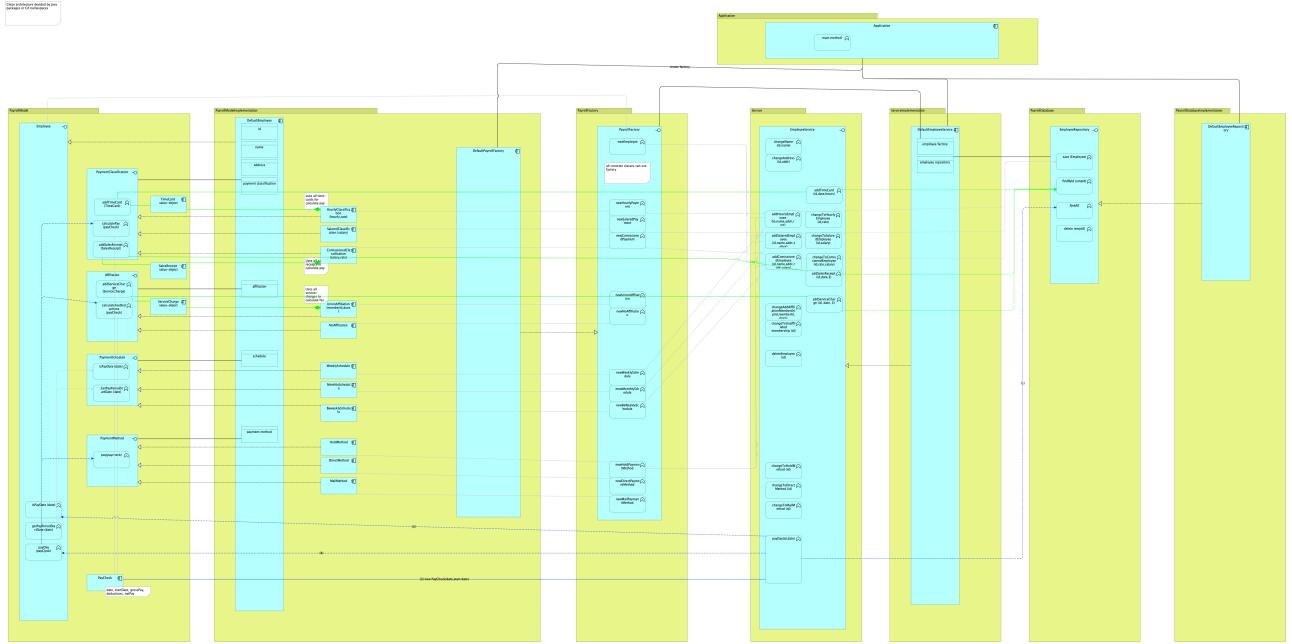


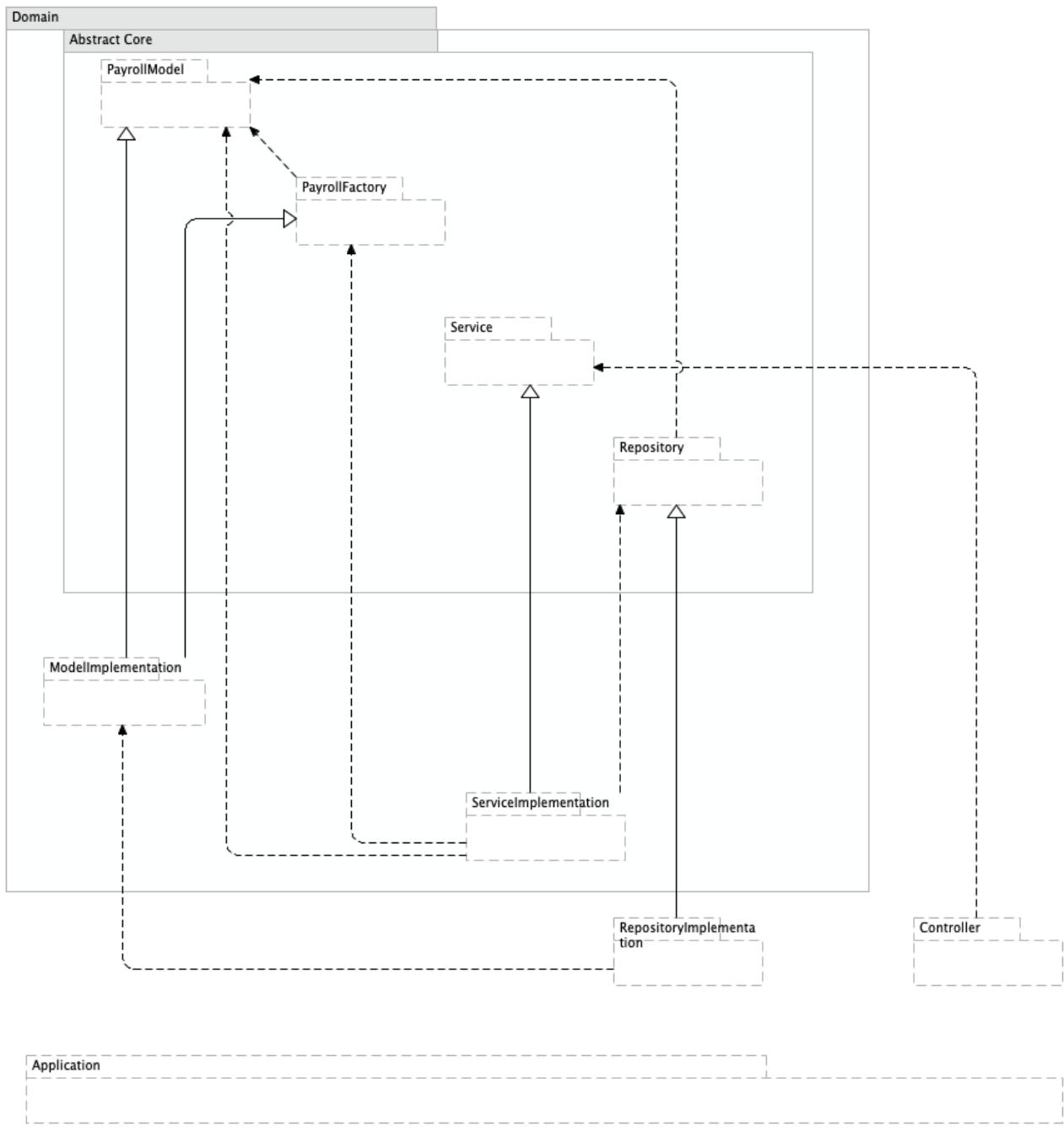
5 Clean Architecture

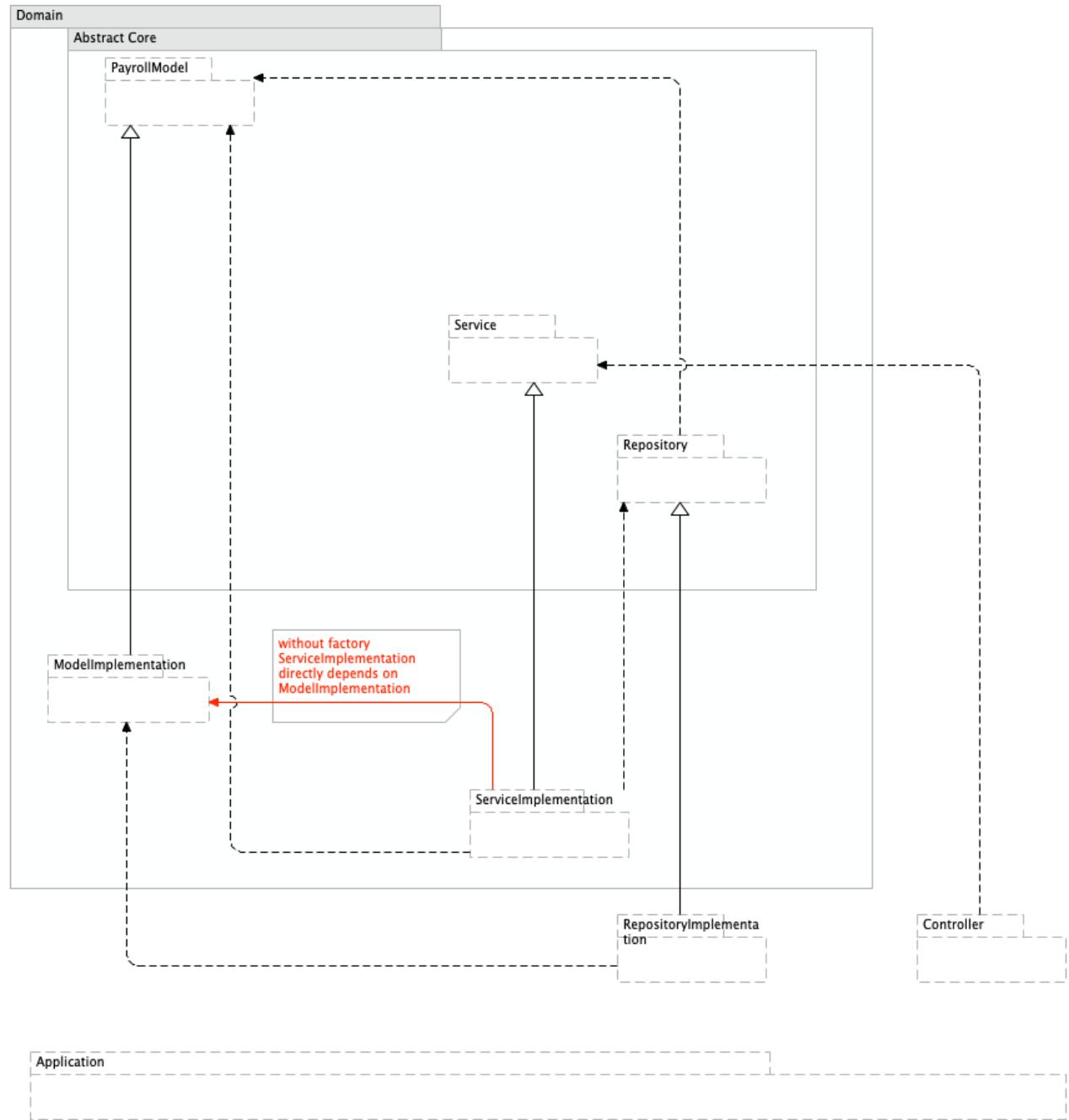
PAYROLL DEMO

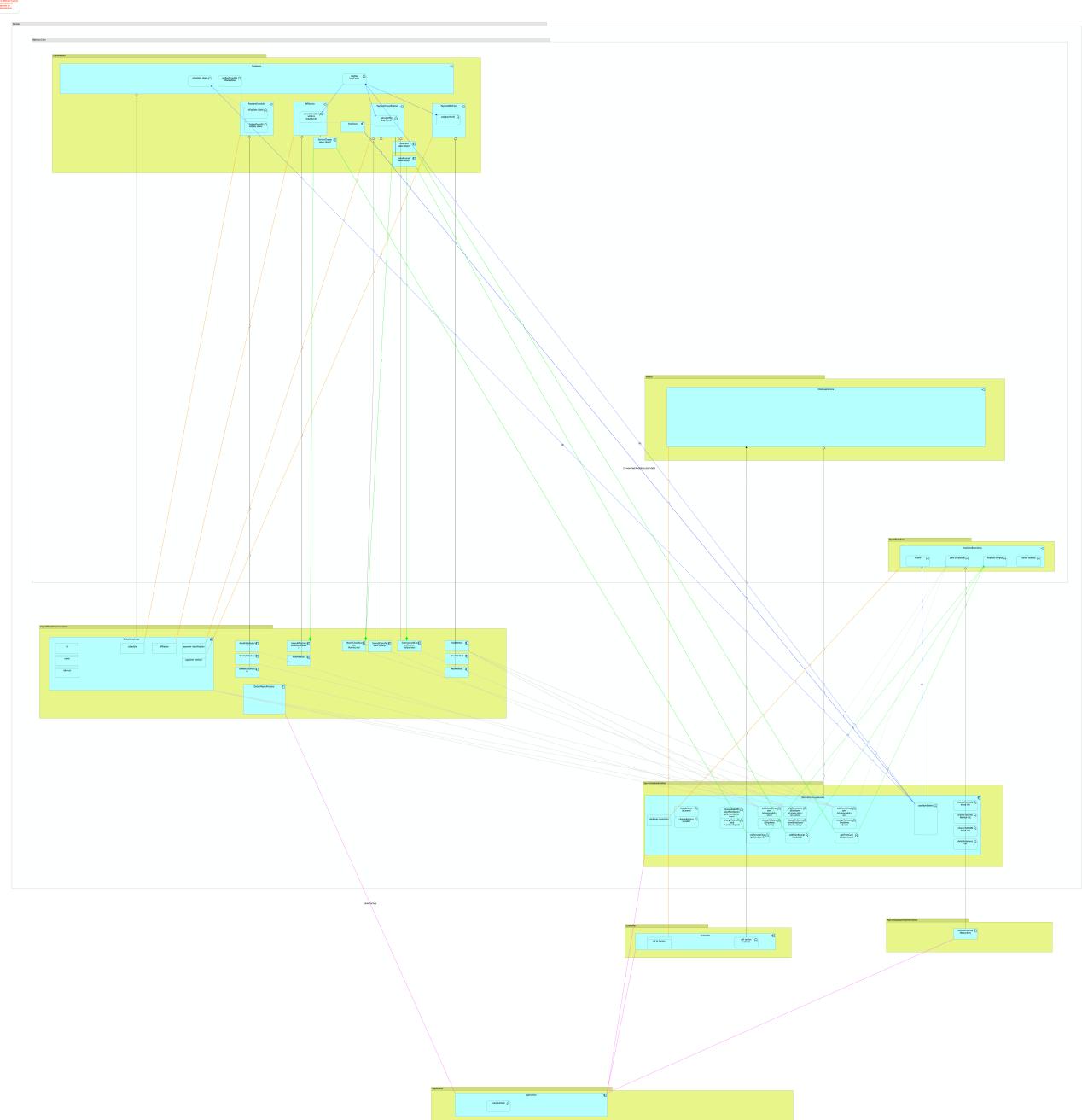
APPLICATION STRUCTURE





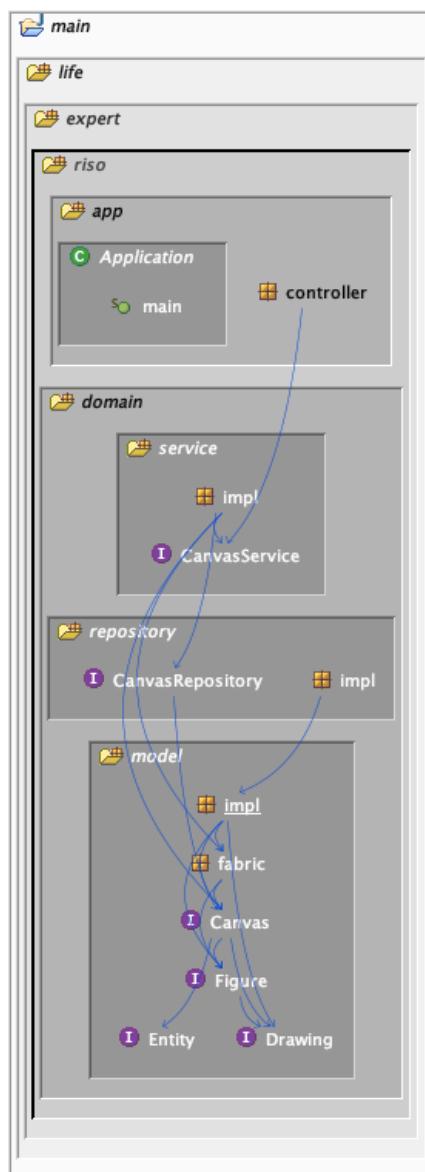


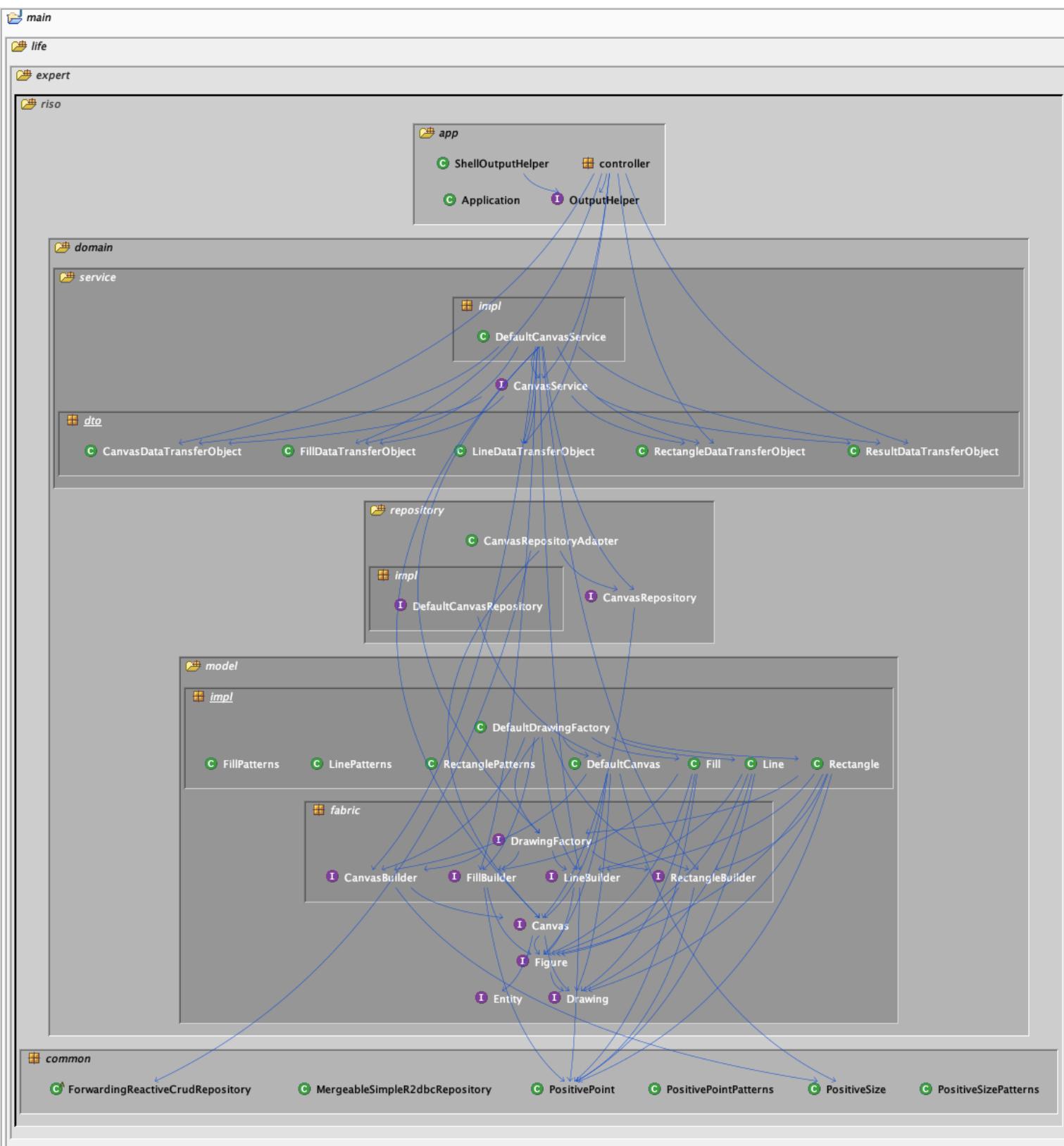


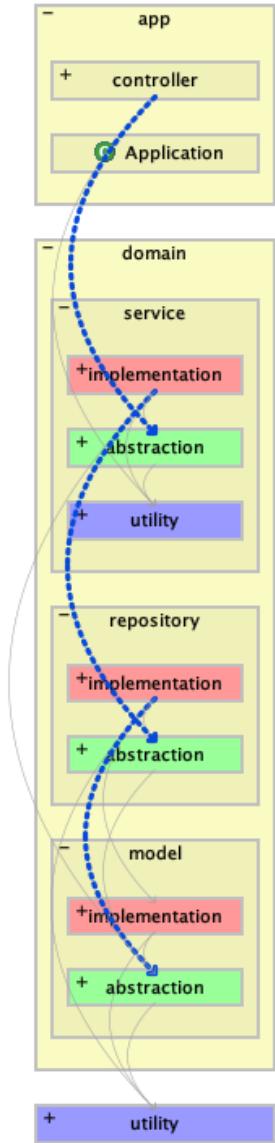


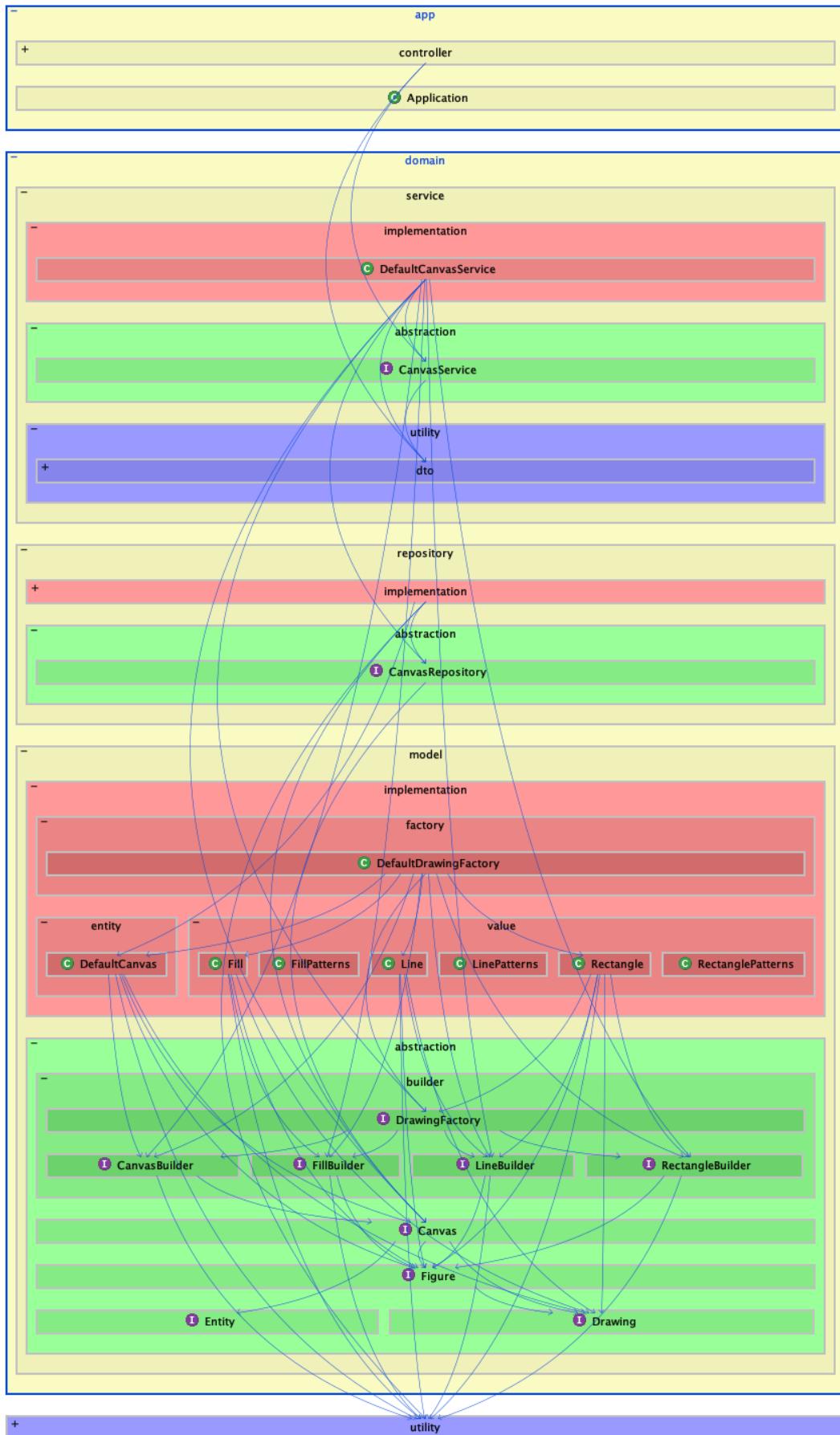
STRUCTURE101

DIAGRAMS

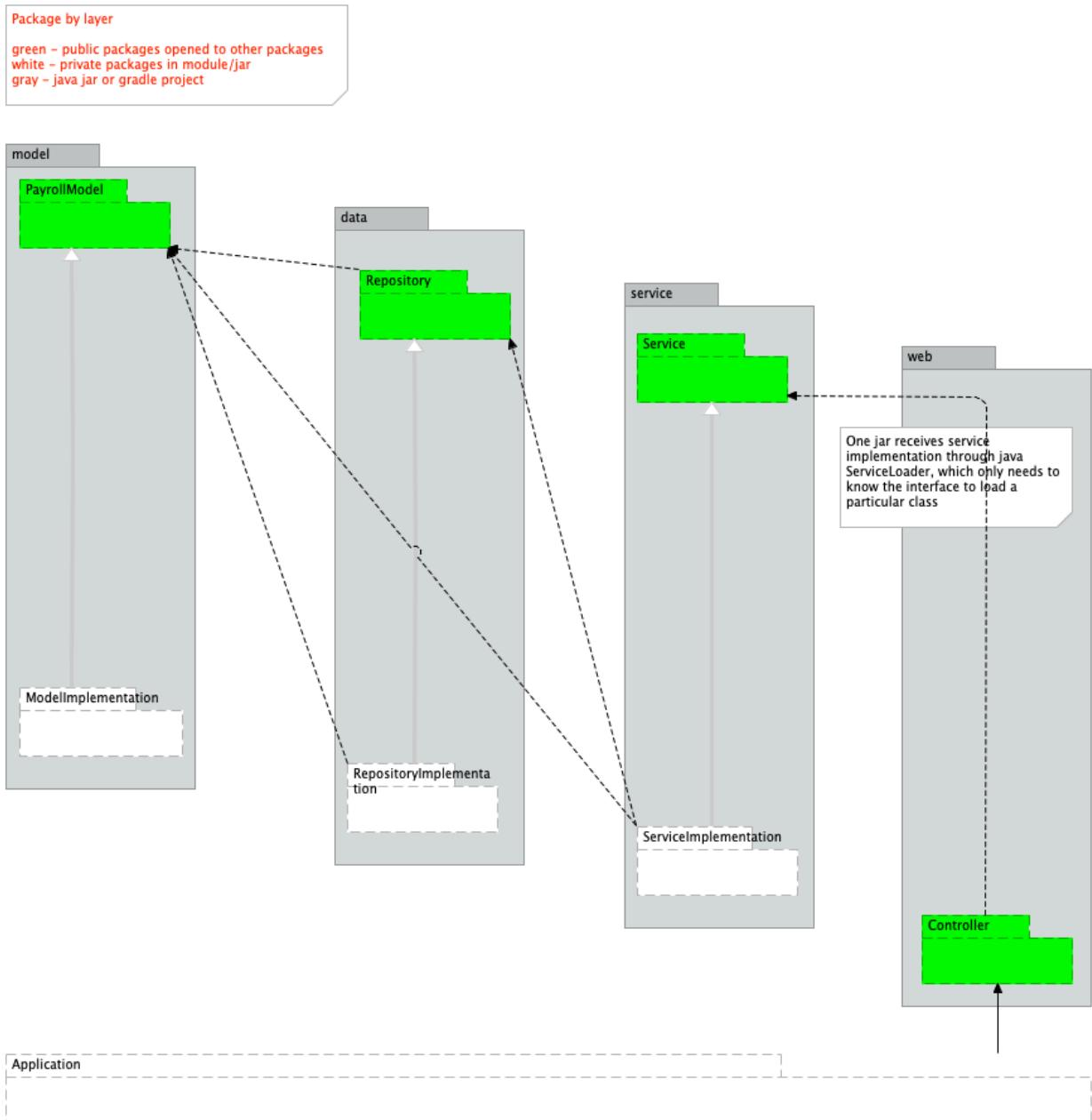




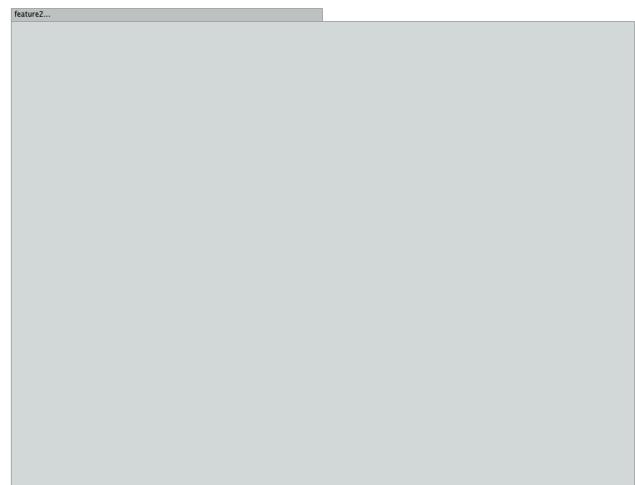
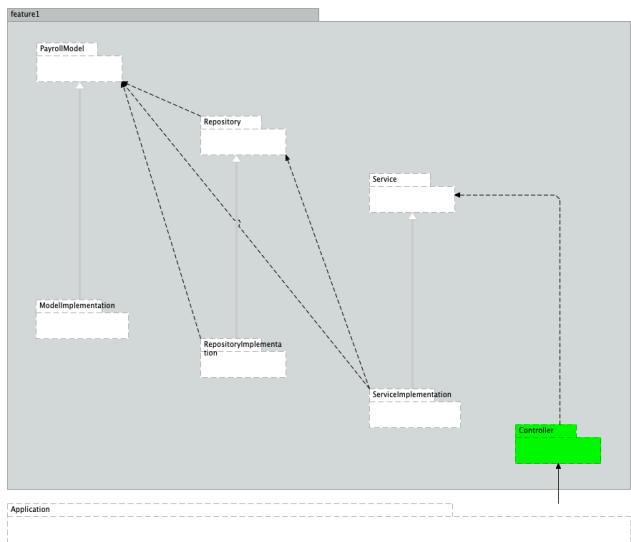




ARCHITECTURAL STYLES

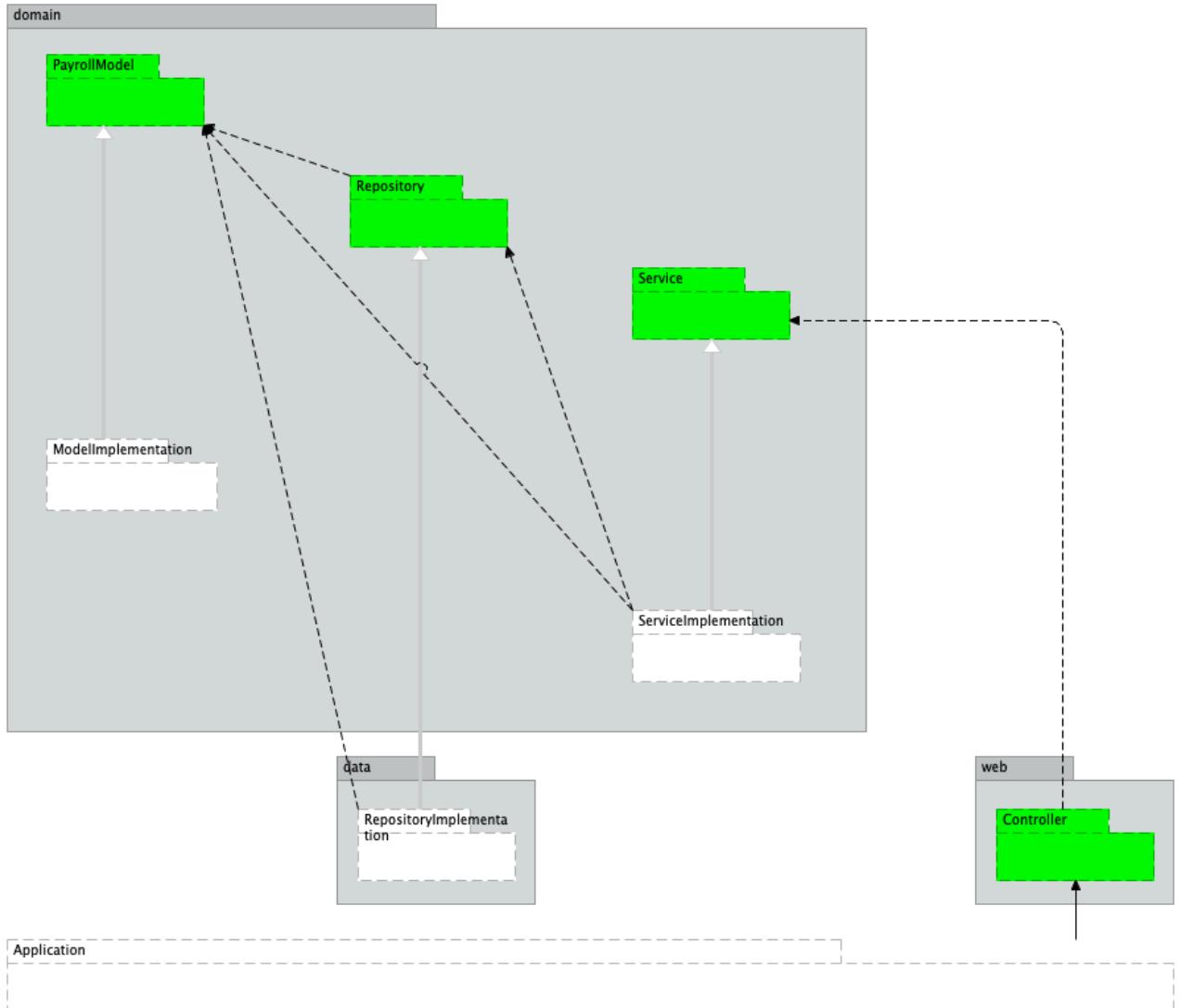


Package by feature
green - public packages in module/jar
white - private packages in module/jar

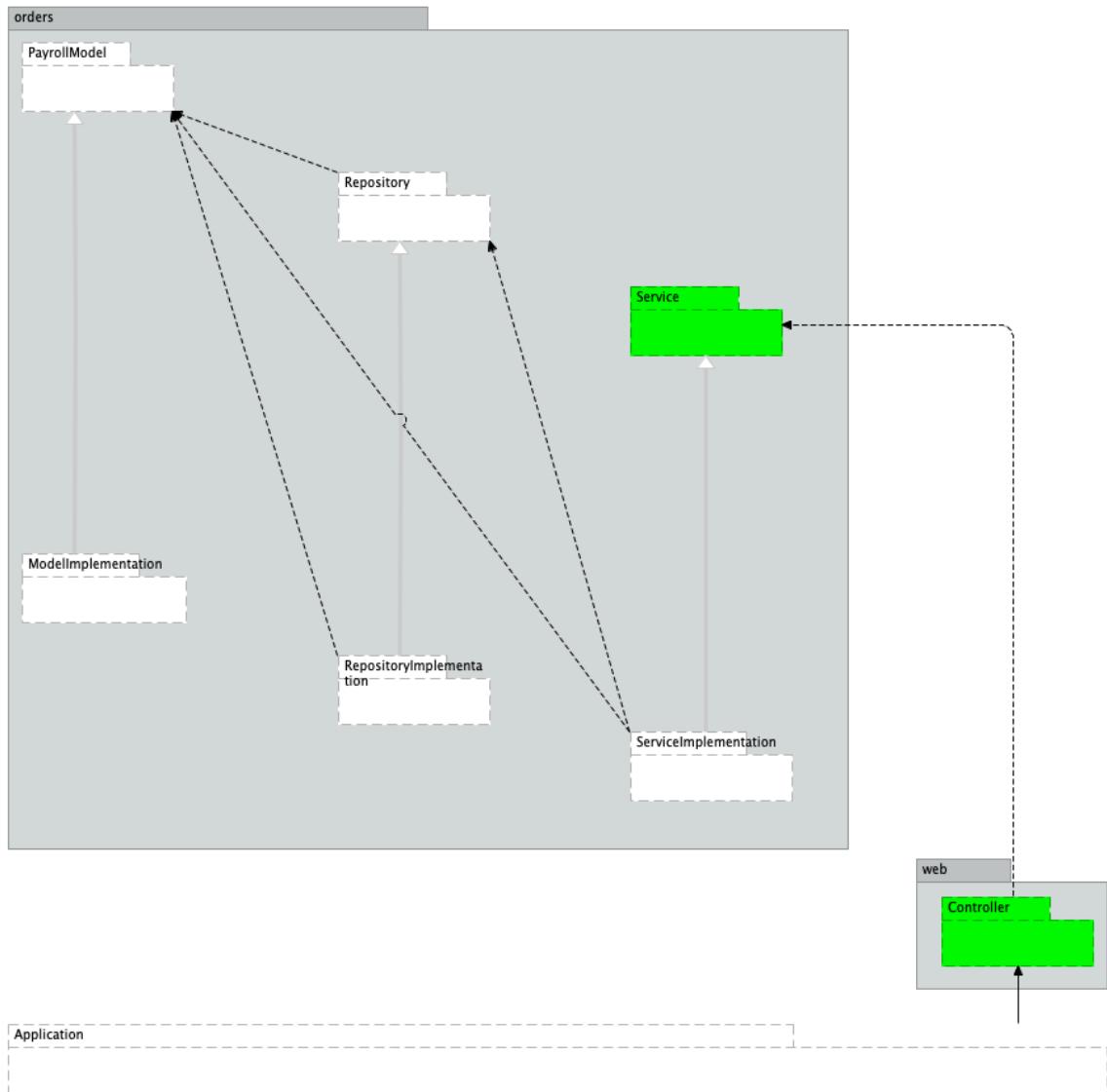


Application

"ports and adapters,
 "the "hexagonal architecture,"
 "boundaries, controllers, entities,"
 green - public packages in module/jar
 white - private packages in module/iar



Package by component
MSA microservices architecture style
green – public packages in module/jar
white – private packages in module/jar





SOME ARCHITECTURAL DIAGRAMS
ARE CONSTRUCTED USING
STRUCTURE101 STUDIO,
COURTESY OF STRUCTURE101

structure101



SOME ARCHITECTURAL DIAGRAMS ARE
CONSTRUCTED USING JARCHITECT,
COURTESY OF CODEGEARS /

COMPLETE CATALOG OF ALL CLASSICAL PATTERNS IN THE
ARCHIMATE LANGUAGE (ARCHITOOL USED) THE VERSION
INCLUDES ALL 155+ PATTERNS COMPLETED (278+ MODELS).
IT'S GREAT OPPORTUNITY TO USE BEST PRACTICES IN YOUR
MICRO SERVICE ARCHITECTURE (ALSO AVIALABLE AT
[HTTPS://GITHUB.COM/WILMERKRISP/PATTERNS](https://github.com/wilmerkrisp/patterns))



KrispWilmer